Undefined 1 (2010) 1–10
IOS Press

The Mastro System for Ontology-based Data Access

Diego Calvanese [a] Giuseppe De Giacomo [b] Domenico Lembo [b] Maurizio Lenzerini [b] Antonella Poggi [b] Mariano Rodriguez-Muro [a] Riccardo Rosati [b] Marco Ruzzi [b] and Domenico Fabio Savo [b]

[a] *Free University of Bozen-Bolzano, Piazza Domenicani 3, I-39100, Bolzano, Italy*
Email: lastname@inf.unibz.it
[b] *Sapienza Universita di Roma,Via Ariosto 25, I-00185, Roma, Italy*
Email: lastname@dis.uniroma1.it

Abstract. In this paper we present Mastro, a java tool for ontology-based data access (OBDA) developed at the University of Rome "La Sapienza" and at the Free University of Bozen-Bolzano. Mastro manages OBDA systems in which the ontology is specified in *DL-Lite*$_{A,id}$, a logic of the *DL-Lite* family of tractable Description Logics specifically tailored to ontology-based data access, and is connected to external JDBC enabled 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 provides a proprietary API, an OWLAPI compatible interface, and a plugin for the Protégé 4 ontology editor. It has been successfully used in several projects carried out in collaboration with important organizations, on which we briefly comment in this paper.

Keywords: Ontology-based data access, Description Logics, Reasoning over ontologies

1. Introduction

In this paper we present Mastro, a tool for ontology-based data access developed at the University of Rome "La Sapienza" and at the Free University of Bozen-Bolzanp. Ontology-based data access (OBDA) refers to a setting in which ontologies are used as a high-level, conceptual view over data repositories, allowing users to access data without the need to know how they are actually organized and where they are stored (cf. Figure 1).

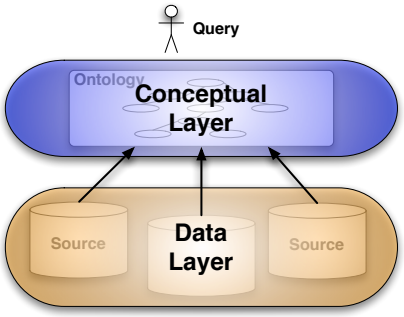

Fig. 1. Ontology-based Data Access

The OBDA approach turns out to be very useful in all those scenarios in which accessing data in a unified and coherent way is difficult, which may happen for several reasons: Databases may have undergone several manipulations during the years, often for optimizing applications using them, and may have lost their original design; they may have been distributed or replicated without a coherent design, so that the information turns out to be dispersed over several independent (maybe heterogeneous) data sources; data in such sources tend

therefore to be redundant and mutually inconsistent; etc.

Through Mastro it is possible to design and manage OBDA systems, i.e., systems in which an ontology is connected to external data sources through mappings. As in data integration systems [24], we use mappings 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 given in terms of a conceptualization of the domain of interest, constructed independently 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 system independent; 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 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 a solid theoretical basis [6,9,8,27, 4]. The ontologies it manages are specified in *DL-Lite*$_{A,id}$, 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 instance level, i.e., the data, largely dominates the intensional level. From the point of view of expressive power, *DL-Lite*$_{A,id}$ captures all basic constructs of the languages for ontologies and for conceptual modeling. Furthermore, it allows for specifying advanced forms of identification constraints [10]. Other general forms of constraints are also expressible, under a particular semantic approximation [8]. We notice that all such constraints turned out to be very useful in practical experiences we conducted with Mastro [1,31], and that such constructs are not part of OWL2, the current W3C standard language for specifying ontologies.

The mapping mechanism adopted by Mastro [27] allows for solving the so-called *impedance mismatch* problem, arising from the fact that, while the data sources store values, the instances of concepts in the ontology are objects. Answering unions of conjunctive queries in OBDA systems managed by Mastro can be done through a very efficient technique (that is in AC^0 with respect to data complexity, i.e., the complexity measured only w.r.t. the extensional level) that reduces this task to standard SQL query evaluation [9,27]. Also, it has been shown that even very slight extensions of the expressive abilities of our system lead beyond this complexity bound [6,5]. However, more expressive queries, essentially all FOL queries expressible over the ontology, can be answered via a similar SQL encoding, under a suitable semantic approximation [8].

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 sources need to be accessed, possibly specified in non-relational form, Mastro can be coupled with an external relational data federation tool[1], which wraps sources and represents them as if they were a single (virtual) relational database.

Mastro comes with its proprietary API, but is equipped also with an OWLAPI compatible interface that has been developed for interaction with OWLAPI compliant applications. In particular, such an interface has been exploited to implement the Mastro plugin for the Protégé 4 ontology editor[2]. Mastro is currently available for download at `http://www.dis.uniroma1.it/~quonto/`.

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 provide an in-depth description of the main modules in which Mastro is organized, briefly describing the procedures and algorithms they realize. In Section 4, we report on three main use cases in which Mastro has been successfully experimented. In Section 5, we comment on other similar approaches and tools. In Section 6, we conclude the paper.

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

[2] `http://protege.stanford.edu/`

2. Ontology-based data access

In OBDA, the aims it 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) [2], which are logics that allow one to represent the domain of interest in terms of *concepts*, denoting sets of objects, *roles*, denoting binary relations between objects, and *attributes*, denoting relationships between objects and values from predefined domains (such as strings, integers, etc.). A DL knowledge base $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is traditionally constituted by a TBox $\mathcal{T}$, representing intensional knowledge, and an ABox $\mathcal{A}$ representing extensional knowledge.

MASTRO is able to deal with DL TBoxes that are expressed in *DL-Lite*$_{A,id}$, a member of the *DL-Lite* family of lightweight DLs [9]. In such DLs, a good tradeoff is achieved between the expressive power of the TBox language 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 stored in databases. We don't specify here the formal syntax and semantics of *DL-Lite*$_{A,id}$, for which we refer to [9,7], but state only that this logic essentially captures standard conceptual modeling formalisms, such as UML Class Diagrams and Entity-Relationship Schemas. Indeed, *DL-Lite*$_{A,id}$ distinguishes at the semantic level between abstract objects and domain values, and allows one to express in a TBox the following kinds of logical assertions: *(i)* inclusion assertions between concepts (that include projections of roles on one of their components), expressing ISA, typing of relations, and mandatory participation to relations or attributes; by using a negated concept one can also express disjointness; *(ii)* inclusion assertions between roles and attributes, to express ISA and disjointness of (binary) relations; and *(iii)* functionality assertions, and complex forms of identification constraints. An ABox instead, contains assertions about specific individuals or values, such as the fact that an individual is an instance of a class, that two individuals are related by a role, or that an attribute relates an individual to some value.

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. Such *mapping assertions* have 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. Intuitively, such 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. 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 form of *global-as-view* (GAV) mapping [24] (cf. also Section 5.2). In order to overcome the so-called *impedance mismatch* between the database, storing values, and the ontology, maintaining abstract objects, the mapping assertions are used 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 constitute the object identifiers for the instances of the ontology [27].

The semantics of DLs is given in terms of standard first-order interpretations. Traditional intensional reasoning tasks w.r.t. a given TBox are class *satisfiability*, i.e., checking whether for some model of the TBox a given concept has a non-empty extension, and class *subsumption*, i.e., checking whether the extension of one class is contained in the extension of another class in every model of the TBox. As for the OBDA setting, let us denote with $\langle \mathcal{T}, \mathcal{M}, \mathcal{D} \rangle$ a TBox $\mathcal{T}$ connected by means of mappings $\mathcal{M}$ to a database $\mathcal{D}$. 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 [27] for the formal details). In this setting, *satisfiability* amounts to checking whether $\langle \mathcal{T}, \mathcal{M}, \mathcal{D} \rangle$ admits at least one model. Queries over the TBox are *unions of conjunctive queries* (UCQs), i.e., unions of select-project-join SQL queries, and in the OBDA setting they are interpreted according to the certain answer semantics [24,27]: the *certain answers* to a query Q over $\langle \mathcal{T}, \mathcal{M}, \mathcal{D} \rangle$ are those tuples that are in the answer to Q for every model of $\langle \mathcal{T}, \mathcal{M}, \mathcal{D} \rangle$. Satisfiability can be reduced easily to query answering, and for the logics of the *DL-Lite* family it has been shown that this latter inference task can be carried out efficiently in the size

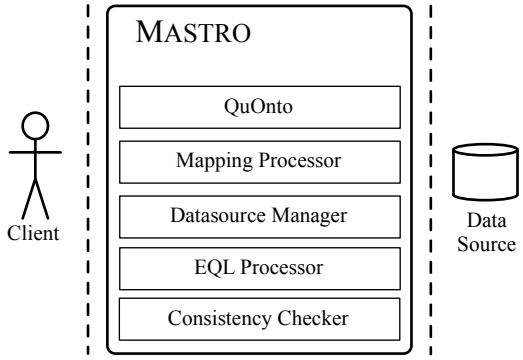

Fig. 2. MASTRO basic architecture

of the data, by relying on query evaluation performed by the underlying relational database. We refer to Section 3.1 for a brief description of this technique as carried out by MASTRO, and to [9] for more formal details.

3. Architecture of MASTRO

In this section, we describe the general architecture of MASTRO, as well as some of the important aspects of the modules that constitute the system, which are shown in Figure 2. Each such module provides basic functionalities that concur to realize the services that MASTRO offers. Among the main services we recall *intensional reasoning support*, *consistency checking*, *conjunctive query answering*, and *epistemic query answering*, all performed over *DL-Lite*$_{A,id}$ ontologies connected through mappings to external data sources. We start our discussion by introducing the *core modules* of MASTRO, i.e., those supporting intensional reasoning and conjunctive query answering. We then continue with a description of the *advanced modules*, i.e., those supporting epistemic query answering and expressive constraint management, as well as checking consistency of the system with respect to such constraints. We conclude the section with a description of the *interfaces* that are available for accessing MASTRO's functionalities.

3.1. The core modules

The core of MASTRO consists of three modules: the QUONTO module, the *Mapping Processor* module, and the *Datasource Manager* module. As for design-time tasks, such modules provide the basic features for ontology and mapping specification and management. In particular *DL-Lite*$_{A,id}$ TBoxes and mappings, both specified according to proprietary XML syntax, are taken as input, parsed, and stored respectively by the QUONTO module and by the mapping processor. As for runtime tasks, focusing on conjunctive query answering, these modules implement the *reformulation*, *unfolding*, and *execution* steps of the query answering procedure, respectively.

QUONTO *module.* QUONTO is a reasoner for *DL-Lite*$_{A,id}$ that provides intentional reasoning services, i.e., class satisfiability, class subsumption, etc., as well as *reformulation* of UCQs. In short, the reformulation process takes as input a UCQ Q expressed over a *DL-Lite*$_{A,id}$ ontology, in fact over a TBox $\mathcal{T}$, and returns a set Q_r of CQs, again expressed over $\mathcal{T}$, that represents the perfect reformulation of Q with respect to $\mathcal{T}$, i.e., the evaluation of Q_r over any *DL-Lite*$_{A,id}$ ABox $\mathcal{A}$ (seen as a database) returns the certain answers to Q with respect to the ontology $\langle \mathcal{T}, \mathcal{A} \rangle$. The reformulation engine implemented in QUONTO is based on the PerfectRef algorithm presented in [9], adapted to natively deal with OWL constructs captured by *DL-Lite*$_{A,id}$ and not explicitly considered in [9], as for example the use of qualified existential restrictions in the right-hand side of concept inclusions (see [15] for details). Also, the reformulation is enhanced with optimizations that allow for reducing the number of queries it generates.

Mapping Processor module. Since MASTRO does not explicitly manage ABoxes, but rather accesses data stored in external systems via mappings, the set of queries Q_r is not evaluated over an ABox, but rather processed according to the mappings to obtain a query that can be evaluated over the data sources. Indeed, Q_r is phrased in terms of classes and properties of the TBox $\mathcal{T}$ and to obtain a query expressed in terms of the alphabet of the sources, it is necessary to perform a further rewriting step dependent on the mapping, which, roughly, substitutes the TBox predicates occurring in Q_r with their definition provided by the mapping assertions. Such a process is called *query unfolding* and is carried out by the Mapping Processor. Generally speaking, query unfolding is a quite straightforward procedure, widely applied in data integration applications. In MASTRO, how-

ever, query unfolding is complicated by the fact that mappings are more complex than those usually adopted in data integration, and by the fact that function symbols occurring in mapping assertions must be taken into account (see Section 2). To deal with such aspects, the Mapping Processor implements the partial evaluation-based unfolding technique from [27], which we briefly summarize in the following: *(i)* we first "split" the mapping assertions in such a way that each assertion has exactly one ontology predicate in the head; *(ii)* we associate an auxiliary predicate to each SQL query in the body of mapping assertions; *(iii)* we transform each mapping assertion into a logic program clause, whose head coincides with the head of the mapping assertion, and the body is an atom expressed in terms of the auxiliary predicate associated to the SQL query in the mapping assertion; *(iv)* we next compute the partial evaluation of Q_r with respect to the logic program gathering all clauses constructed in the previous step, i.e., a new set of conjunctive queries phrased only in terms of the auxiliary predicates[3]; and, last, *(v)* we translate the partial evaluation into an SQL query over the sources, by replacing each auxiliary predicate with the associated SQL view. The use of partial evaluations and auxiliary predicates gives us the flexibility to work on the unfolding process at an abstract level, independently from the type of data sources and the specific form of the views that we associate to the auxiliary predicates. Further details on the technique can be found in [27,31].

Datasource Manager module. This module is responsible for maintaining the connections to the data sources, for coordinating query execution, and for management of database resources such as pointer maintenance and transaction management. The most relevant feature of this module is the ability to parallelize query execution to improve query answering performance. This feature is a key feature that allows for fast query processing even in the case where a very big number of queries is generated by the reformulation-unfolding process. Depending on the system configuration, several execution threads are spawned and queries are assigned to the different execution threads (see Figure 3). Each thread manages a group of queries, which are executed sequentially within the thread itself. When the processing of any of these queries terminates, the result set associated to the answer is forwarded to a *result set wrapper* that is returned to the client and that keeps receiving the results of subsequent queries.

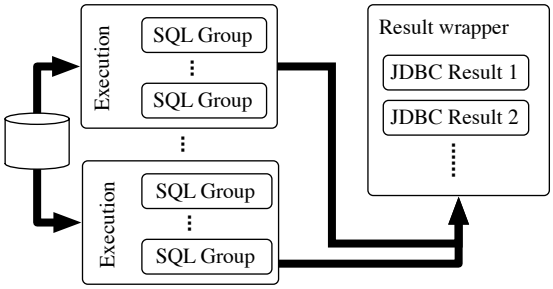

Fig. 3. Parallel query execution management

3.2. Advanced functionalities

Built on top of the core functionalities, the MASTRO system offers access to the features of two *advanced* modules that allow for epistemic query answering and expressive constraint management, and consistency check, respectively.

EQL Processor module. This module provides the ability to specify and execute epistemic queries, specifically so-called EQL queries [8] expressed over *DL-Lite*$_{A,id}$ ontologies. Roughly speaking, an EQL query is an SQL query specified over virtual relations expressed as UCQs over the ontology (called inner queries). Answering an EQL query consists in computing the extension of the virtual relations, each containing the certain answers of the corresponding inner query with respect to the ontology, the mapping and the source data, and evaluating the SQL query over such relations. This actually corresponds to putting each inner query under the scope of an epistemic operator (see [8] for details). Rather than computing the extension of the virtual relations, the EQL Processor exploits the reformulation service offered by the QUONTO module, and the unfolding service provided by the Mapping Processor module, in order to rewrite each inner query into an SQL query over the sources, thus transforming the entire EQL query into an SQL query over the sources. Such a query is then sent to the Datasource Manager, which is in charge of evaluating it. EQL queries

[3] The term *partial evaluation* is due to the connection of this technique with the optimization technique from the logic programming literature that carries the same name.

are extremely useful when the expressive power of CQs is not enough. For example, they allow for expressing negation and comparison in queries. Also, through EQL queries it is possible to specify powerful integrity constraints over the ontology (as shown in [14]), which go beyond the expressivity of the DLs of the *DL-Lite* family.

Consistency Checker module. This module allows Mastro to verify whether an OBDA system it manages is satisfiable, i.e., it admits at least one model. By virtue of the characteristics of *DL-Lite*$_{A,id}$, such a check can be reduced to answering suitable queries posed over the ontology, each one associated to a TBox assertion or to an integrity constraint that can be violated by data at the sources. The Consistency Checker therefore relies on the services for query answering provided by the core modules and the EQL Processor we described above. Indeed, apart from basic features, which enable for checking the violation of functionalities or exclusion dependencies (i.e., inclusions with negation in their right-hand side), the Consistency Checker allows one to verify consistency of very expressive constraints (i.e., identification and EQL constraints), which is reduced to answering EQL queries over the ontology. The Consistency Checker can also localize data that violate TBox assertions and/or constraints over the ontology. It can indeed generate those queries (UCQs or EQL) whose answers return data that give rise to inconsistencies.

3.3. Interfaces

Mastro's functionalities can be accessed in three ways: through a proprietary API, through an OWLAPI compatible interface, and by means of a plugin for the Protégé 4 ontology editor. In particular, Mastro's proprietary API is the one that is used to integrate all the modules that compose the system. This API is also used for the implementation of specific procedures required during the deployment of the tool in application scenarios, as the ones described in Section 4. The API also provides parser facilities for loading ontologies with mappings using Mastro's own XML syntax and allows for accessing all the functionalities offered by the system.

The OWLAPI compatible interface is built on top of Mastro's API. This public interface allows for a straightforward integration of Mastro with OWLAPI[4]-OBDALib[5] applications. The main access point of this API is the so-called MastroOWLReasoner, an implementation of the OWLReasoner interface from the OWLAPI, and of the OBDAReasoner interface, which is part of the OBDALib. Through functionalities provided by the OWLReasoner interface, clients have access to Mastro's services associated with traditional OWL reasoners, i.e., concept subsumption, satisfiability, etc. Through the OBDAReasoner interface, clients can access Mastro's OBDA related functionalities, i.e., specification of ontology with mappings, conjunctive query answering, etc.

Fig. 4. Mastro interfaces

In order to provide its functionalities, the OWLAPI compatibility layer relies on an OWL to *DL-Lite*$_{A,id}$ translator module which is able to process OWL 2 ontologies represented by means of OWLAPI objects and produce *DL-Lite*$_{A,id}$ ontologies represented by Mastro's internal API objects. OWL 2 is an expressive language, hence it may happen that some OWL 2 axioms cannot be translated into corresponding *DL-Lite*$_{A,id}$ expressions. In such cases the module produces a syntactic approximation in *DL-Lite*$_{A,id}$.

Last but not least, Mastro can also be accessed by means of a Protégé 4 plugin. This plugin is built on top of the MastroOWLReasoner (cf. Figure 4) and exposes all functionalities of Mastro, allowing for the use of the facilities offered by Protégé for ontology editing and by the OBDA Plugin for Protégé [28] for editing of mappings towards external data sources. The Mastro plugin extends Protégé with new features to express assertions that are not part of the OWL 2 language but that are supported by Mastro, such as identification and EQL constraints.

[4] `http://owlapi.sourceforge.net/`
[5] `http://obda.inf.unibz.it/`

4. The system at work: experiences on real cases

In order to demonstrate the usefulness of OBDA and the feasibility and efficiency of the MASTRO system, we are experimenting the system in three real world applications, on which we report now.

SELEX Sistemi Integrati. We first report on the case study conduced with MASTRO at SELEX Sistemi Integrati (SELEX-SI), a Finmeccanica Company that is world leader in the provision of integrated defence and air traffic critical systems. In such a case study we considered Configuration and Data Management (C&DM) in SELEX-SI and focused on a significative portion of the data manipulated in this context [1].

C&DM is a technical management model that is central to all SELEX-SI activities since it governs the entire products' life cycle. We mainly focused on the data concerning the design and the production of components that are used to realize complex systems, physical deployment of such components, and analysis of their obsolescence. Currently, such data are stored in various, partially overlapping sources, and managed by five different systems under diverse data models (relational, XML-based, etc.). We used MASTRO to integrate such data, in such a way that relevant queries connected to important C&DM informational needs could be automatically processed by MASTRO. We used the external data federation tool Websphere Federation Server[6] to present to MASTRO all the data sources as if they were a single relational database. At the end of the federation process, we produced a relational schema with around 50 relational tables, with an average of 15 attributes each, managed by Websphere.

After the federation step described above, we designed the C&DM domain in terms of a *DL-Lite*$_{A,id}$ TBox. On the basis of our analysis, we produced an ontology that models concepts concerning the design and the production of components that are used to realize complex systems, and several aspects concerning the physical deployment of such components and the analysis of their obsolescence, considering also possible substitutions. In some relevant cases, according to the requirements of the domain experts, it was necessary to model at the ontology level information on the provenance of the data. Then, we defined specific ontology elements to represent the data source from which the information is retrieved. This allowed for the specification of queries comparing information stored in different sources. The overall *DL-Lite*$_{A,id}$ ontology for C&DM contains around 40 concepts, 30 roles, and 50 attributes.

We also defined around 100 mapping assertions connecting the source and the ontology. Finally, we tested a set of significative queries for C&DM at SELEX-SI, and demonstrated the usefulness of MASTRO to easily access information in all cases where a user would query separately each data source, and manually combine the single answers.

Monte dei Paschi di Siena. The usefulness of the MASTRO system goes beyond data integration applications, and embraces data quality management, as we demonstrated through the experimentation of OBDA carried out in a joint project with Banca Monte dei Paschi di Siena (MPS)[7], Free University of Bozen-Bolzano, and SAPIENZA Università di Roma. In such a project we used MASTRO for accessing a set of data sources from the actual MPS data repository by means of an ontology and focused on querying such an ontology to find out inconsistent data [31].

In our case study 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 data sources are managed by a specific application. The application is in charge of guaranteeing data integrity (in fact, the underlying database does not force constraints on data). Not only this application performs various updates, but an automatic procedure is executed on a daily basis to examine the data collected in the database so as to identify connections between customers that are relevant for the credit rating calculus. By both looking at these sources, and interviewing domain experts, we designed an ontology representing the conceptual model of the domain, and the mapping between the ontology and

[6] `http://www-306.ibm.com/software/data/integration/federation_server/`.

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

the sources. The resulting OBDA system is expressed in terms of approximately 600 *DL-Lite*$_{A,id}$ axioms over 79 concepts and 33 roles, and 200 mapping assertions.

The experimentations confirmed the importance of several distinguished features of our system, namely, the possibility to specify identification constraints and epistemic constraints, which have been used extensively to model important business rules. Checking that such rules are satisfied by data retrieved from the sources through mappings has been the main objective of the project. With respect to this, we highlight two kinds of data quality problems that we were able to detect, one related to unexpected incompletenesses in the data sources, and the other one related to inconsistencies in the data.

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.

Network inventory systems. Finally, we briefly mention an ongoing experimentation we are carrying out in the telecommunication context, and specifically on the domain of network inventory systems The OBDA system we have realized in this case study is formed by a *DL-Lite*$_{A,id}$ TBox constructed over 112 concepts, 84 roles and containing around 100 identification constraints and EQL constraints. The data source is a 3 million tuple database, stored in 45 relational tables managed by the Oracle 10g RDBMS. The mapping layer is formed by 348 mapping assertions.

According to the idea that the quality of the data stored in the sources can be measured in terms of the amount of data respecting the constraints implied by the domain description offered by the ontology, we laid down the basis of a methodology for using MASTRO for data quality management. The last two experiences we have mentioned, have revealed how the same ontology can be used both for querying data and for checking the quality of data sources. The different objectives in the two cases only partially influence the design of the mapping, whereas the ontology design turns out to be an independent task. Another important lesson concerns the mapping generation: according to our experiences, due to the complexity of extracting the right semantics of the source tables, the bulk of the work in mapping specification has to be essentially carried out manually.

5. Comparison with other approaches and tools

To the best of our knowledge, MASTRO is the only system currently available for ontology-based data access. In particular it is the only system allowing both for specification and reasoning on an ontology, and for mapping it to external data sources. It follows that, from a general perspective, it is not possible to compare MASTRO with other tools providing exactly the same functionalities. A comparison with other systems, however, can be carried out by considering separately the main features provided by MASTRO. In particular we will consider *(i)* the MASTRO features for ontology definition and reasoning, and compare the reasoning engine of MASTRO with other ontology reasoners, and *(ii)* the MASTRO features for data source access by means of mappings, and compare MASTRO with tools for data integration.

5.1. Tools for ontology definition and reasoning

As we explained before, the core reasoning engine of MASTRO is based on QUONTO, a reasoner for *DL-Lite*$_{A,id}$. Differently from QUONTO, other well-known DL reasoners such as Racer [19], Pellet [32], and Fact++ [20] are essentially focused on standard ontology reasoning services (instance checking, ontology satisfiability, concept subsumption, etc.). Although some features for ABox query processing have been provided by such tools, and several optimization have been carried out, such systems are not able to deal with ontologies with very large ABoxes (e.g., with several millions of membership assertions) as the ones we considered in our experimentations. This is mainly due to the inherent computational complexity of answering queries in the expressive DL languages supported by the above mentioned systems.

We mention some other *DL-Lite* based approaches and reasoners. In [21] an approach to query answering alternative to the one of QUONTO is presented. Besides a (less complex) query rewriting step, such an approach requires to also extend the ABox (stored in a database managed by a RDBMS) with the aim of reducing the amount of rewritten queries produced by the reformulation step. Indeed, in QUONTO the size of the rewriting may be exponential with respect to the size of the original query. However, the drawback of such an approach combining query rewriting with ABox extension, is that it cannot be applied to data sources that are outside the control of the system, thus preventing its use in many practical cases. In such cases one might adopt an approach that requires to materialize the external data sources in a local repository, over which data manipulations can be performed. This, however, has the drawback of having to reflect the data updates occurring at the sources on the local copy. Nevertheless, the idea in [21] is interesting and results given by a first experimentation are encouraging.

Another approach, purely intensional as the one provided by QUONTO, aiming at optimizing the reformulation step, has been implemented in the REQUIEM prototype reasoner [26]. REQUIEM makes use of a rewriting algorithm different from the one used in QUONTO, which reduces the number of queries in the final reformulation, and is able to deal natively with the qualified existential construct[8]. However, it currently supports neither any of the advanced features, such as identification or epistemic constraint management, nor mappings to external databases.

The OWLGres prototype [33], 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. More details on the comparison between OWLGres and QUONTO can be found in [15].

[8] This feature has been recently implemented in QUONTO and is currently under testing.

5.2. Tools for data integration

None of the above mentioned systems provides mechanism for connecting an ontology to external independent data sources with powerful mappings. With respect to this feature, the approaches and tools that are closer to MASTRO are those developed in the context of data integration. Several major software vendors (Oracle, IBM, Microsoft, etc.) provide nowadays products for information integration: such tools can be seen as a collection of wrappers allowing the users to access a variety of data sources and to see such sources as if they were a single database. However, no real semantic integration is carried out, and such systems have to be considered as data federation tools rather than semantic data integration tools. From the research point of view, semantic data integration [24] has been studied deeply in the last two decades, producing a number of interesting results. In particular, the approaches can be classified according to the form they adopt for the mapping that connects the global reconciled view to the data sources. Two main formalisms have been proposed for the mappings representation [24]: the *global-as-view (GAV)* approach, in which entities of the global schema are defined by means of queries over the source schema, and the *local-as-view (LAV)* approach, in which source entities are defned by means of queries over the global schema. We notice that currently MASTRO adopts a particular forma of GAV mapping. Examples of GAV systems are TSIMMIS (The Stanford-IBM Manager of Multiple Information Sources) [12], and Garlic [34]. Their architecture is essentially a hierarchy of data source wrappers and mediators. The latter are modules whose basic tasks consists in putting together the data returned by the wrappers into the final answers to users' queries. The mapping mechanisms provided by such systems can be considered a first form of GAV mapping implemented in a purely procedural way. LAV proposals, instead, have been based on a more declarative approach. Information Manifold [25], INFOMASTER [17], and Picsel [18] are notable examples. Such systems implement query answering by means of query rewriting algorithms, such as the inverse rules, the bucket, or the MiniCon [29] algorithms.

Despite the intensive research described so far, only few efforts have been dedicated to the study

of data integration through ontologies. Indeed, all the systems mentioned above suffer from some weaknesses from the modeling perspective, mainly due to the limited expressive expressive power of the languages provided to model the global schema of the integration system. In this direction, MASTRO aims at overcoming this limitation by providing the best expressive power allowed while preserving tractability of reasoning and of the integration tasks.

6. Conclusions

In this paper we presented MASTRO, a system for ontology-based data access, which provides a comprehensive solution to such a problem by offering features both for specifying and reasoning on an ontology, and for mapping external data sources to it. Efficient algorithms for advanced forms of query answering are implemented, which enable effective data access. Experiences on real cases yielded very encouraging results, showing the applicability of the MASTRO approach to real-world problems.

MASTRO can be extended in several directions. In particular, we plan to extend the system with respect to the following aspects:

(i) Enriching the ontology representation and reasoning layer with *inconsistency tolerant* capabilities: indeed, when integrating different data sources under the same ontology, it may happen that the reconciled data do not satisfy the ontology. In such cases, repairing the data could be inconvenient, or not possible at all. However, it is possible and important to exploit techniques for consistent query answering [3,13] in order to make MASTRO able to support meaningful query answering even in the presence of inconsistent data. Theoretical results at the basis of the approach we want to implement can be found in [22,23].

(ii) Allowing for more expressive forms of mappings: LAV mappings could be adopted for those settings in which source data may be incomplete with respect to the ontology used to access the sources underlying the system.

(iii) Implementing "write-also" capabilities: most of the studies carried out in information integration, and the systems proposed to solve the integration problem are mainly oriented towards a read-only approach. This means that the data flows from the sources to the global ontology only. However, several studies have been carried out on the *update* problem [16,11,35], attempting to reflect over the sources an update expressed in terms of the global ontology. We already carried out some experiments in this direction and plan to extend MASTRO in order to support such features.

(iv) finally, we are currently working to optimize the reformulation step in QUONTO, following the line of research of [30,26], in order to reduce the number of queries produced: this aspect may have a crucial impact on the performance of the whole system.

References

[1] A. Amoroso, G. Esposito, D. Lembo, P. Urbano, and R. Vertucci. Ontology-based data integration with MASTRO-I for configuration and data management at SELEX Sistemi Integrati. In *Proc. of the 16th Ital. Conf. on Database Systems (SEBD 2008)*, pages 81–92, 2008.

[2] 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.

[3] A. Calì, D. Lembo, and R. Rosati. On the decidability and complexity of query answering over inconsistent and incomplete databases. In *Proc. of the 22nd ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2003)*, pages 260–271, 2003.

[4] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, M. Rodríguez-Muro, and R. Rosati. Ontologies and databases: The *DL-Lite* approach. In *Semantic Technologies for Informations Systems – 5th Int. Reasoning Web Summer School (RW 2009)*, volume 5689 of *LNCS*, pages 255–356. Springer, 2009.

[5] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, R. Rosati, and M. Ruzzi. Data integration through *DL-Lite*$_{\mathcal{A}}$ ontologies. In K.-D. Schewe and B. Thalheim, editors, *Revised Selected Papers of the 3rd Int. Workshop on Semantics in Data and Knowledge Bases (SDKB 2008)*, volume 4925 of *LNCS*, pages 26–47. Springer, 2008.

[6] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Data complexity of query answering in description logics. In *Proc. of the 10th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2006)*, pages 260–270, 2006.

[7] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Can OWL model football leagues? In *Proc. of the 3rd Int. Workshop on OWL: Experiences and Directions (OWLED 2007)*, volume 258 of *CEUR,* `http://ceur-ws.org/`, 2007.

[8] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. EQL-Lite: Effective first-order query processing in description logics. In *Proc. of the*

20th Int. Joint Conf. on Artificial Intelligence (IJCAI 2007), pages 274–279, 2007.

[9] 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.

[10] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Path-based identification constraints in description logics. In *Proc. of the 11th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2008)*, pages 231–241, 2008.

[11] D. Calvanese, E. Kharlamov, W. Nutt, and D. Zheleznyakov. Updating ABoxes in *DL-Lite*. In *Proc. of the 4th Alberto Mendelzon Int. Workshop on Foundations of Data Management (AMW 2010)*, volume 619 of *CEUR,* `http://ceur-ws.org/`, pages 3.1–3.12, 2010.

[12] S. S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. D. Ullman, and J. Widom. The TSIMMIS project: Integration of heterogeneous information sources. In *Proc. of the 10th Meeting of the Information Processing Society of Japan (IPSJ'94)*, pages 7–18, 1994.

[13] J. Chomicki. Consistent query answering: Five easy pieces. In *Proc. of the 11th Int. Conf. on Database Theory (ICDT 2007)*, volume 4353 of *LNCS*, pages 1–17. Springer, 2007.

[14] C. Corona, E. Di Pasquale, A. Poggi, M. Ruzzi, and D. F. Savo. When OWL met *DL-Lite* .... In *Proc. of the 5th Workshop on Semantic Web Applications and Perspectives (SWAP 2008)*, 2008.

[15] C. Corona, M. Ruzzi, and D. F. Savo. Filling the gap between OWL 2 QL and QuOnto: ROWLKit. In *Proc. of the 22nd Int. Workshop on Description Logic (DL 2009)*, volume 477 of *CEUR,* `http://ceur-ws.org/`, 2009.

[16] G. De Giacomo, M. Lenzerini, A. Poggi, and R. Rosati. On the update of description logic ontologies at the instance level. In *Proc. of the 21st Nat. Conf. on Artificial Intelligence (AAAI 2006)*, pages 1271–1276, 2006.

[17] M. R. Genereseth, A. M. Keller, and O. M. Duschka. Infomaster: An information integration system. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pages 539–542, 1997.

[18] F. Goasdoue, V. Lattes, and M.-C. Rousset. The use of CARIN language and algorithms for information integration: The Picsel system. *Int. J. of Cooperative Information Systems*, 9(4):383–401, 2000.

[19] V. Haarslev and R. Möller. RACER system description. In *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2001)*, volume 2083 of *LNAI*, pages 701–705. Springer, 2001.

[20] I. Horrocks. The FaCT system. In *Proc. of the 7th Int. Conf. on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX'98)*, volume 1397 of *LNAI*, pages 307–312. Springer, 1998.

[21] R. Kontchakov, C. Lutz, D. Toman, F. Wolter, and M. Zakharyaschev. The combined approach to query answering in *DL-Lite*. In *Proc. of the 12th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2010)*, 2010.

[22] D. Lembo, M. Lenzerini, R. Rosati, M. Ruzzi, and D. F. Savo. Inconsistency-tolerant semantics for description logics. In *Proc. of the 4th Int. Conf. on Web Reasoning and Rule Systems (RR 2010)*, 2010.

[23] D. Lembo and M. Ruzzi. Consistent query answering over description logic ontologies. In *Proc. of the 1st Int. Conf. on Web Reasoning and Rule Systems (RR 2007)*, 2007.

[24] M. Lenzerini. Data integration: A theoretical perspective. In *Proc. of the 21st ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2002)*, pages 233–246, 2002.

[25] A. Y. Levy, A. Rajaraman, and J. J. Ordille. Querying heterogenous information sources using source descriptions. In *Proc. of the 22nd Int. Conf. on Very Large Data Bases (VLDB'96)*, 1996.

[26] H. Pérez-Urbina, B. Motik, and I. Horrocks. A comparison of query rewriting techniques for *DL-lite*. In *Proc. of the 22nd Int. Workshop on Description Logic (DL 2009)*, volume 477 of *CEUR,* `http://ceur-ws.org/`, 2009.

[27] 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.

[28] A. Poggi, M. Rodríguez-Muro, and M. Ruzzi. Ontology-based database access with DIG-Mastro and the OBDA Plugin for Protégé. In *Proc. of the 4th Int. Workshop on OWL: Experiences and Directions (OWLED 2008 DC)*, 2008.

[29] R. Pottinger and A. Y. Halevy. MiniCon: A scalable algorithm for answering queries using views. *Very Large Database J.*, 10(2–3):182–198, 2001.

[30] R. Rosati and A. Almatelli. Improving query answering over *DL-Lite* ontologies. In *Proc. of the 12th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2010)*, 2010.

[31] 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 the 23rd Int. Workshop on Description Logic (DL 2010)*, volume 573 of *CEUR,* `http://ceur-ws.org/`, pages 20–31, 2010.

[32] E. Sirin and B. Parsia. Pellet: An OWL DL reasoner. In *Proc. of the 17th Int. Workshop on Description Logic (DL 2004)*, volume 104 of *CEUR,* `http://ceur-ws.org/`, 2004.

[33] M. Stocker and M. Smith. Owlgres: A scalable OWL reasoner. In *Proc. of the 5th Int. Workshop on OWL: Experiences and Directions (OWLED 2008)*, volume 432 of *CEUR,* `http://ceur-ws.org/`, 2008.

[34] M. Tork Roth, M. Arya, L. M. Haas, M. J. Carey, W. F. Cody, R. Fagin, P. M. Schwarz, J. T. II, and E. L. Wimmers. The Garlic project. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, page 557, 1996.

[35] D. Zheleznyakov, D. Calvanese, E. Kharlamov, and W. Nutt. Updating TBoxes in *DL-Lite*. In *Proc. of the 23rd Int. Workshop on Description Logic (DL 2010)*, volume 573 of *CEUR,* `http://ceur-ws.org/`, pages 102–113, 2010.