Using ontologies for semantic data integration

Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini,
Antonella Poggi, Riccardo Rosati

Dipartimento di Ingegneria Informatica, Automatica e Gestionale
Sapienza Università di Roma
⟨degiacomo,lembo,lenzerini,poggi,rosati⟩@diag.uniroma1.it

Abstract. While big data analytics is considered as one of the most important paths to competitive advantage of today's enterprises, data scientists spend a comparatively large amount of time in the data preparation and data integration phase of a big data project. This shows that data integration is still a major challenge in IT applications. Over the past two decades, the idea of using semantics for data integration has become increasingly crucial, and has received much attention in the AI, database, web, and data mining communities. Here, we focus on a specific paradigm for semantic data integration, called Ontology-Based Data Access (OBDA). The goal of this paper is to provide an overview of OBDA, pointing out both the techniques that are at the basis of the paradigm, and the main challenges that remain to be addressed.

1 Introduction

Big data analytics is considered as one of the most important paths to competitive advantage of today's enterprises. However, after years of focus on technologies for big data storing and processing, many observers are pointing out that making sense of big data cannot be done without suitable tools for conceptualizing, preparing, and integrating data[1]. Indeed, a common misconception about big data is that it is a black box: you load data and magically gain insight. This is not the case: loading a big data platform with quality data with enough structure to deliver value is a lot of work.

Thus, it is not surprising that data scientists spend a comparatively large amount of time in the data preparation phase of a project. Whether you call it data wrangling, data munging, or data integration, it is estimated that 50%-80% of a data scientists' time is spent on preparing data for analysis. If we consider that in any IT (information Technology) organization, data governance is also essential for tasks other than data analytics, we can conclude that the challenge of identifying, collecting, retaining, and providing access to all relevant data for the business at an acceptable cost, is huge.

Data integration is considered as one of the old problems in data management, and the above observations show that it is a major challenge today. Formal approaches to data integration started in the 90's [36,17,47,33]. Since then, research both in academia and in industry has addressed a huge variety of aspects of the general problem. Among them, we want to focus on the idea of using semantics for making data integration more

[1] `http://www.dbta.com/`

powerful. Using semantics here means conceiving data integration systems where the semantics of data is explicitly specified, and is taken into account for devising all the functionalities of the system. Over the past two decades, this idea has become increasingly crucial to a wide variety of information-processing applications, and has received much attention in the AI, database, web, and data mining communities [40]. In this paper we focus on a specific paradigm for semantic data integration. Indeed, about a decade ago, a new paradigm for modeling and interacting with a data integration systems, called "Ontology-Based Data Access" (OBDA), was proposed [14,42]. According to such paradigm, the client of the information system is freed from being aware of how data and processes are structured in concrete resources (databases, software programs, services, etc.), and interacts with the system by expressing her queries and goals in terms of a conceptual representation of the domain of interest, called ontology.

OBDA aims at a radical solution to some of the major challenges that the complex information systems software is throwing at us at its current stage of maturity: (i) the increasing complexity of the solutions required, and the impossibility of formally verifying them; (ii) the fact that the lifespan of software technologies (not to talk about hardware ones) is much shorter that the lifespan of the solutions and the applications that use them; (iii) the obvious observation that legacy systems, or simply legacy components, are everywhere, and realistically they will remain everywhere for a long time. For all the above reasons, while the amount of data stored in current information systems and the processes making use of such data continuously grow, turning these data into information, and governing both data and processes are still tremendously challenging tasks for Information Technology. The problem is complicated by the proliferation of data sources both within a single organization, and in cooperating environments. The following factors explain why such a proliferation constitutes a major problem with respect to the governance goal:

- Despite the fact that the initial design of a collection of data sources is adequate, corrective maintenance actions tend to re-shape them into a form that often diverges from the original conceptual structure.
- It is common practice to change a data source (e.g., a database) so as to adapt it to specific application-dependent needs, so that it often becomes a data structure coupled to a specific application, rather than an application-independent database.
- The data stored in different sources and the processes operating over them tend to be redundant, and mutually inconsistent, mainly because of the lack of central, coherent and unified data management tasks. This poses great difficulties with respect to the goal of accessing data in a unified and coherent way.

A system realizing the vision of OBDA is constituted by three components:

- The *ontology*, whose goal is to provide a formal, clean and high level representation of the domain of interest, and constitutes the component with which the clients of the information system (both humans and software programs) interact.
- The *data source* layer, representing the existing data sources in the information system, which are managed by the processes and services operating on their data.
- The *mapping* between the two layers, which is an explicit representation of the relationship between the data sources and the ontology, and is used to translate the

operations on the ontology (e.g., query answering) in terms of concrete actions on the data sources.

Thus, OBDA is an advanced approach to semantic data integration, in which the global schema is given in terms of an ontology, i.e., a formal and conceptual view of the application domain, rather than simply a unified view of the data at the sources.

The goal of this paper is to provide an overview of the OBDA paradigm, pointing out both the techniques that are at the basis of the paradigm, and the main challenges that remain to be addressed. The paper is organized as follows. In Section 2 we illustrate the general, formal framework underlying the paradigm. In Section 3 and Section 4 we deal with the main computational problem that has been studied so far in OBDA, namely query answering. Section 5 concludes the paper with a discussion on various aspects that are already the subject of current study, and will be increasingly important in the near future.

2 Framework for Ontology-based Data Access

In this section we first provide a general framework for OBDA, and then we focus on a notable framework instantiation that allows for practical application of the OBDA paradigm.

Before proceeding further, we point out that here we abstract from the problem of dealing with multiple and heterogeneous sources, by assuming that we have access to a single relational database through an SQL interface. In practice, such a database might be obtained through the use of off-the-shelf data federation tools which allow seeing a set of data sources as if they were a single relational database. Note that this relational database does not represent the integrated view of the various sources, but simply a replication of the source schemas expressed in terms of a unique format.

A general framework for OBDA. An *OBDA specification* $\mathcal{J}$ is as a triple $\langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle$, where $\mathcal{O}$ is an ontology, $\mathcal{S}$ is a relational schema, called source schema, and $\mathcal{M}$ is a mapping from $\mathcal{S}$ to $\mathcal{O}$. More precisely, $\mathcal{O}$ represents intensional knowledge about the domain, expressed in some logical language. Typically, $\mathcal{O}$ is a lightweight Description Logic (DL) TBox [7], i.e., it is expressed in a language ensuring both semantic richness and efficiency of reasoning, and in particular of query answering. The mapping $\mathcal{M}$ is a set of mapping assertions, each one relating a query over the source schema to a query over the ontology.

An *OBDA system* is a pair $(\mathcal{J}, D)$ where $\mathcal{J}$ is an OBDA specification and D is a database for the source schema $\mathcal{S}$, called source database for $\mathcal{J}$. The semantics of $(\mathcal{J}, D)$ is given in terms of the logical interpretations that are models of $\mathcal{O}$ (i.e., satisfy all axioms of $\mathcal{O}$, and satisfy $\mathcal{M}$ with respect to D). The notion of mapping satisfaction depends on the semantic interpretation adopted on mapping assertions. Commonly, such assertions are assumed to be *sound*, which intuitively means that the results returned by the source queries occurring in the mapping are a subset of the data that instantiate the ontology. The set of models of $\mathcal{J}$ with respect to D is denoted with $Mod_D(\mathcal{J})$.

In OBDA systems, the main service of interest is *query answering*, i.e., computing the answers to user queries, which are queries posed over the ontology. It amounts to return the so-called *certain answers*, i.e., the tuples that satisfy the user query in all the interpretations in $Mod_D(\mathcal{J})$. Query answering in OBDA is thus a form of reasoning under incomplete information, and is much more challenging than classical query evaluation over a database instance.

From the computational perspective, query answering depends on *(1)* the language used for the ontology; *(2)* the language used for user queries; and *(3)* the language used to specify the queries in the mapping. In the following, we consider a particular instantiation of the OBDA framework, in which we choose each such language in such a way that query answering is guaranteed to be tractable w.r.t. the size of the data. We remark that the configuration we get is to some extent "maximal", i.e., as soon as we go beyond the expressiveness of the chosen languages, we lose this nice computational behaviour (cf. Section 3).

A tractable OBDA framework. From the general framework we obtain a tractable one by choosing appropriate languages as follows:

- the ontology language is *DL-Lite*$_A$ or its subset *DL-Lite*$_R$;
- the mapping language follows the *global-as-view* (GAV) approach [33];
- the user queries are unions of conjunctive queries.

Ontology language

DL-Lite$_A$[42] is essentially the maximally expressive member of the *DL-Lite* family of lightweight DLs [14]. In particular, its subset *DL-Lite*$_R$ has been adopted as the basis of the OWL 2 QL profile of the W3C standard OWL (Ontology Web Language) [38]. As usual in DLs, *DL-Lite*$_A$ allows for representing the domain of interest in terms of *concepts*, denoting sets of objects, and *roles*, denoting binary relations between objects. In fact, *DL-Lite*$_A$ considers also *attributes*, which denote binary relations between objects and values (such as strings or integers), but for simplicity we do not consider them in this paper. From the expressiveness point of view, *DL-Lite*$_A$ is able to capture essentially all the features of Entity-Relationship diagrams and UML Class Diagrams, except for completeness of hierarchies. In particular, it allows for specifying ISA and disjointness between either concepts or roles, mandatory participations of concepts into roles, the typing of roles. Formally, a *DL-Lite*$_A$ TBox is a set of assertions obeying the following syntax:

$$\begin{array}{lll} B_1 \sqsubseteq B_2 \quad B_1 \sqsubseteq \neg B_2 & & \text{(concept inclusions)} \\ R_1 \sqsubseteq R_2 \quad R_1 \sqsubseteq \neg R_2 & & \text{(role inclusions)} \\ (\mathsf{funct}\ R) & & \text{(role functionalities)} \end{array}$$

where B_1 and B_2 are basic concepts, i.e., expressions of the form A, $\exists P$, or $\exists P^-$, and R, R_1, and R_2 are a basic roles, i.e., expressions of the form P, or P^-. A and P denote an *atomic concept* and an *atomic role*, respectively, i.e., a unary and binary predicate from the ontology alphabet, respectively. P^- is the *inverse* of an atomic role P, i.e., the role obtained by switching the first and second components of P, and $\exists P$ (resp. $\exists P^-$), called existential unqualified restriction, denotes the projection of the role P on its first (resp. second) component. Finally $\neg B_2$ (resp. $\neg R_2$) denotes the negation

of a basic concept (resp. role). Assertions in the left-hand side (resp. right-hand side) of the first two rows are called positive (resp. negative) inclusions. Assertions of the form $(\mathsf{funct}\ R)$ are called role functionalities and specify that an atomic role, or its inverse, is functional. *DL-Lite*$_A$ poses some limitations on the way in which positive role inclusions and role functionalities interact. More precisely, in a *DL-Lite*$_A$ TBox an atomic role that is either functional or inverse functional cannot be specialized, i.e., if $(\mathsf{funct}\ P)$ or $(\mathsf{funct}\ P^-)$ are in the TBox, no inclusion of the form $R \sqsubseteq P$ or $R \sqsubseteq P^-$ can occur in the TBox. *DL-Lite*$_R$ is the subset of *DL-Lite*$_A$ obtained by removing role functionalities altogether.

A *DL-Lite*$_A$ *interpretation* $\mathcal{I} = (\Delta^I, \cdot^I)$ consists of a non-empty *interpretation domain* Δ^I and an *interpretation function* $\cdot^I$ that assigns to each atomic concept A a subset A^I of Δ^I, and to each atomic role P a binary relation P^I over Δ^I. In particular, for the constructs of *DL-Lite*$_A$ we have:

$$\begin{array}{ll} A^I \subseteq \Delta^I & (\exists R)^I = \{o \mid \exists o' . (o, o') \in R^I\} \\ P^I \subseteq \Delta^I \times \Delta^I & (\neg B)^I = \Delta^I \setminus B^I \\ (P^-)^I = \{(o_2, o_1) \mid (o_1, o_2) \in P^I\} & (\neg R)^I = (\Delta^I \times \Delta^I) \setminus R^I \end{array}$$

Let C be either a basic concept B or its negation $\neg B$. An interpretation $\mathcal{I}$ satisfies a concept inclusion $B \sqsubseteq C$ if $B^I \subseteq C^I$. Similarly for role inclusions. Also, $\mathcal{I}$ satisfies a role functionality $(\mathsf{funct}\ R)$ if the binary relation R^I is a function, i.e., $(o, o_1) \in R^I$ and $(o, o_2) \in R^I$ implies $o_1 = o_2$.

Mapping language

The mapping language in the tractable framework allows mapping assertions of the following the forms,

$$\phi(\boldsymbol{x}) \rightsquigarrow A(f(\boldsymbol{x})) \qquad \phi(\boldsymbol{x}) \rightsquigarrow P(f_1(\boldsymbol{x}_1), f_2(\boldsymbol{x}_2)) \tag{1}$$

where $\phi(\boldsymbol{x})$ is a domain independent first-order query (i.e., an SQL query) over $\mathcal{S}$, with free variables $\boldsymbol{x}$, A and P are as before, variables in $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ also occur in $\boldsymbol{x}$, and f, possibly with subscripts, is a function. Intuitively, the mapping assertion in the left-hand side, called concept mapping assertion, specifies that individuals that are instances of the atomic concept A are constructed through the use of the function f from the tuples retrieved by the query $\phi(\boldsymbol{x})$. Similarly for the mapping assertion in the right-hand side of (1), called role mapping assertion. Each assertion is of type GAV, i.e., it associates a view over the source (represented by $\phi(\boldsymbol{x})$) to an element of the global schema (in this case the ontology). However, differently from traditional GAV mappings, the use of functions is crucial here, since we are considering the typical scenario in which data sources do not store the identifiers of the individuals that instantiate the ontology, but only maintain values. Thus, functions are used to address the semantic mismatch existing between the extensional level of $\mathcal{S}$ and $\mathcal{O}$ [42]. We notice that a mapping using assertions of the form (1) is indeed expressible in R2RML, the W3C recommendation for specifying mappings from relational database to RDF datasets [20]. Formally, we say that an interpretation $\mathcal{I}$ satisfies a mapping assertion $\phi(\boldsymbol{x}) \rightsquigarrow A(f(\boldsymbol{x}))$ with respect to a source database D, if for each tuple of constants $\boldsymbol{t}$ in the evaluation of $\phi(\boldsymbol{x})$ on D, $(f(\boldsymbol{t}))^I \in A^I$, where $(f(\boldsymbol{t}))^I \in \Delta^I$ is the interpretation of $f(\boldsymbol{t})$ in $\mathcal{I}$, that is, $f(\boldsymbol{t})$ acts

simply as a constant denoting an object[2]. Satisfaction of assertions of the form $\phi(\boldsymbol{x}) \rightsquigarrow P(f_1(\boldsymbol{x}_1), f_2(\boldsymbol{x}_2))$ is defined analogously. We also point out that *DL-Lite*$_A$ adopts the Unique Name Assumption (UNA), that is, different constants denote different objects, and thus different ground terms of the form $f(\boldsymbol{t})$ are interpreted with different elements in Δ^I.[3]

User queries

In our tractable framework for OBDA, user queries are conjunctive queries (CQs) [2], or unions thereof. With $q(\boldsymbol{x})$ we denote a CQ with free variables $\boldsymbol{x}$. A Boolean CQ is a CQ without free variables. Given an OBDA system $(\mathcal{J}, D)$ and a Boolean CQ q over $\mathcal{J}$, i.e., over the TBox of $\mathcal{J}$, we say that q is *entailed by* $(\mathcal{J}, D)$, denoted with $(\mathcal{J}, D) \models q$, if q evaluates to true in every $\mathcal{I} \in Mod_D(\mathcal{J})$. When the user query $q(\boldsymbol{x})$ is non-Boolean, we denote with $cert_D(q(\boldsymbol{x}), \mathcal{J})$ the *certain answers* to q with respect to $(\mathcal{J}, D)$, i.e., the set of tuples $\boldsymbol{t}$ such that $(\mathcal{J}, D) \models q(\boldsymbol{t})$, where $q(\boldsymbol{t})$ is the Boolean CQ obtained from $q(\boldsymbol{x})$ by substituting $\boldsymbol{x}$ with $\boldsymbol{t}$.

Query answering

Although query answering in the general framework may become soon intractable or even undecidable, depending on the expressive power of the various languages involved, the tractable framework has been designed to ensure tractability of query answering. We end this section by illustrating the basic idea for achieving tractability.

In the tractable OBDA framework previously described, one can think of a simple chase procedure [2] for query answering, which first retrieves an initial set of concept and role instances from the data source through the mapping, and then, using the ontology axioms, "expands" such a set of instances deriving and materializing all the logically entailed concept and role assertions; finally, queries can be evaluated on such an expanded set of instances. Unfortunately, in *DL-Lite*$_A$ (and in *DL-Lite*$_R$ already) the instance materialization step of the above technique is not feasible in general, because the set of entailed instance assertions starting from even very simple OBDA specifications and small data sources may be infinite.

As an alternative to the above materialization strategy, most of the approaches to query answering in OBDA are based on query rewriting, where the aim is to first compute the perfect rewriting q' of a query q w.r.t. an OBDA specification $\mathcal{J}$, and then evaluate q' over the source database. Actually, the above described OBDA framework allows for modularizing query rewriting. Indeed, the current techniques for OBDA consist of two phases: a phase of *query rewriting w.r.t. the ontology* followed by a phase of *query rewriting w.r.t. the mapping*. In the first phase, the initial query q is rewritten with respect to the ontology, producing a new query q_1, still over the ontology signature: intuitively, q_1 "encodes" the knowledge expressed by the ontology that is relevant for answering the query q. In the second phase, the query q_1 is rewritten with respect to the mapping $\mathcal{M}$, thus obtaining a query q_2 to be evaluated over the source data. Thus, the

[2] As usual, we assume to deal with pre-interpreted data types, and thus we can harmlessly use $\boldsymbol{t}$ to denote both the values and the constants representing them.

[3] In fact, if we restrict to *DL-Lite*$_R$, then UNA becomes immaterial and can be dropped, as done in OWL 2 QL.

mapping assertions are used for reformulating the query into a new one expressed over the source schema signature.

In the following two sections we delve into the details of the two phases constituting the rewriting-based query answering algorithm described above.

3 Query rewriting with respect to the ontology

In order to isolate the properties of ontology rewriting, in this section we assume that all relevant data in the sources have been stored, using the mapping $\mathcal{M}$, in a database whose schema coincides with the ontology signature (in DL jargon, such database is called ABox database). Note that the ABox database can be constructed by computing, for each mapping assertion, the tuples returned by the query on the left-hand side of the assertion, and then inserting into the ABox databases all the facts as sanctioned by the right-hand side of the assertion. According to this scenario, the phase of rewriting a query q with respect to the ontology aims at deriving a query q_1 still expressed over the signature of the ontology such that evaluating q_1 over the ABox database returns the set of certain answers to q with respect to the whole specification. The goal of this section is to discuss the techniques for the ontology rewriting phase, including their computational complexity.

Most of the proposed techniques [14,41,18] start from a CQ or a UCQ (i.e., a set of CQs), and end up producing a UCQ that is an expansion of the initial query. They are based on variants of clausal resolution [30]: every rewriting step essentially corresponds to the application of clausal resolution between a CQ among the ones already generated and a concept or role inclusion axiom of the ontology. Each such step produces a new conjunctive query that is added to the resulting UCQ. The rewriting process terminates when a fix-point is reached, i.e., no new CQ can be generated.

A potential bottleneck of the rewriting approach is caused by the size of the rewritten query, and several research works aim at optimization techniques addressing this issue. For example, the first algorithm for query rewriting w.r.t. a *DL-Lite* ontology [14] has been improved in [41,18] by refining and optimizing the way in which term unification is handled by the above resolution step. Notice that the sentences corresponding to the ontology axioms may be Skolemized (e.g., due to the presence of existentially quantified variables in the right-hand side of a concept inclusion): to compute perfect rewritings, the unification of Skolem terms during resolution can actually be constrained in various ways with respect to standard resolution.

Some recent proposals for optimizing query rewriting w.r.t. the ontology (e.g., [46,18,25]) are based on the use of Datalog queries besides CQs and UCQs, to express either intermediate results or the final rewritten query. The same idea has also been used to extend query rewriting to more expressive, not necessarily first-order rewritable (see below) ontology languages [41,18,24,11]. Other approaches take a more radical view, and propose strategies based on partial materialization of instance assertions [29].

The results in [14,42] show that, following the technique illustrated above, query answering is *first-order rewritable*, i.e., for each union of CQ q over $\mathcal{J}$, it is possible to compute a first-order query q_r such that, for each ABox database D, $\boldsymbol{t} \in cert_D(q, \mathcal{J})$ iff $\boldsymbol{t}$ is in the evaluation of q_r over D. Since q_r can be effectively expressed as an

SQL query, this property is actually saying that CQ answering can be reduced to query evaluation over a relational database (thus, it is in AC^0, a subclass of LOGSPACE), for which we can rely on standard relational DBMSs. The above property also implies that CQ answering is in AC^0 in ABox complexity, which is the complexity of evaluating a first-order query over a relational database. Indeed, this is an immediate consequence of the fact that the complexity of the above phase of query rewriting is independent of the data source, and that the final rewritten query is an SQL expression. It can also be shown that conjunctive query answering in the OBDA setting is NP-complete w.r.t. combined complexity, i.e., the complexity of the problem with respect to the size of the whole input (data source, OBDA specification, and query). This is the same as the combined complexity of SQL query answering over the data source.

Finally, an important question is whether we can further extend the ontology specification language of OBDA without losing the above nice computational property of the query rewriting phase. In [15] it is shown that adding any of the main concept constructors considered in Description Logics and missing in *DL-Lite*$_A$ (e.g., negation, disjunction, qualified existential restriction, range restriction) causes a jump of the data complexity of conjunctive query answering in OBDA, which goes beyond the class AC^0. This issue has been further investigated in [6].

As for the query language, we note that going beyond unions of CQs is problematic from the point of view of tractability, or even decidability. For instance, adding negation to CQs causes query answering to become undecidable [27].

4 Query rewriting with respect to the mapping

Next we discuss the second phase of query rewriting in OBDA; namely the phase of rewriting with respect to the mapping. It is well-known by the studies on data integration [33] that rewriting a query w.r.t. a GAV mapping boils down to a simple unfolding strategy, which essentially means substituting every predicate of the input query with the queries that the mapping associates to that predicate [33].

In OBDA, however, query rewriting w.r.t. mappings is complicated by the following two issues: *(i)* OBDA mappings allow for constructing objects that are instances of the ontology predicates from the values stored in the data source, in order to deal with the mentioned impedance mismatch problem; *(ii)* the source queries in the mapping are expressed using the full expressive power of SQL, which is needed to bridge the large cognitive distance that may exist between the ontology and the source schema.

Solutions to issue (*i*) depend on the strategy adopted to construct objects from values. When functors applied to values are used, as in the OBDA framework instantiation we presented above, logic terms constructed through such functors can be treated in the standard way in the unifications at the basis of the unfolding procedure: see, e.g., the algorithm proposed in [42], which relies on techniques from partial evaluation of logic programs. In the R2RML standard [20], functors are realized through templates that construct W3C compliant URIs for objects from the values returned by the SQL query in the mapping assertion.

Instead, issue (*ii*) above heavily affects the performance of query answering. Indeed, current SQL engines have hard times in optimizing the execution of queries expressed

over virtual views, like those introduced by the unfolding, that use complex SQL features such as union, nesting, or aggregation. Performance problems are of course amplified when there are several SQL queries mapping the same ontology predicate. Due to the above mentioned limitations, it is not realistic to group all such queries within a single mapping assertion for each predicate. However, without such grouping, the mapping associates several queries to the same predicate, and therefore the size of the query obtained by rewriting w.r.t. the mapping may be exponential in the size of the input query. Indeed, in real-world applications, it may very well happen that the size of the produced rewriting is too large to be handled by current SQL engines. Techniques to avoid or mitigate these issues are currently under investigation (see Section 5).

We observe that, beside sound mapping assertions, the literature on data integration has also considered complete or exact mapping assertions, in order to deal with the cases in which the data that satisfy the global schema or ontology are respectively either a superset of or the same set as the data returned by the source queries. However, both such assumptions soon lead to intractable query answering, as shown in [1,16].

Also, as for the query language to be adopted in the mapping, we notice that while our framework already allows for very expressive queries over the source schema, queries over the ontology are less expressive. Hence, in the rest of this section, we consider the impact of enabling GLAV mapping assertions [33] in our framework, where a GLAV mapping allows CQs (with existentially quantified variables) on the right-hand side of the assertions. At a first glance, a mapping specified through assertions of the form (1) might be considered a pure GAV mapping [33], since no existentially quantified variables occur in the right-hand side of mapping assertions. In the following we show that, in fact, the presence of object terms of the form $f(\boldsymbol{x})$ allows the above mapping to be more expressive, in the sense that, under certain conditions, OBDA specifications with GLAV mappings can be transformed into specifications having mappings of the form (1) shown in Section 2, which have an analogous behaviour with respect to query answering.

First of all, we formally define a GLAV mapping from a source schema $\mathcal{S}$ to a TBox $\mathcal{O}$ as a set of assertions of the form

$$\phi(\boldsymbol{x}) \rightsquigarrow \psi(\boldsymbol{x}, \boldsymbol{y}) \tag{2}$$

such that $\phi(\boldsymbol{x})$ is as before, i.e., a domain independent first-order query over $\mathcal{S}$, and $\exists \boldsymbol{y}.\psi(\boldsymbol{x}, \boldsymbol{y})$ is a CQ over $\mathcal{O}$ that may contain terms of the form $f(\boldsymbol{x'})$, where variables $\boldsymbol{x'}$ also occur in $\boldsymbol{x}$. Given a database instance D for $\mathcal{S}$, an interpretation $\mathcal{I}$ satisfies a GLAV mapping assertion if for each tuple $\boldsymbol{t}$ in the evaluation of $\phi(\boldsymbol{x})$ over D, the Boolean CQ $\exists \boldsymbol{y}.\psi(\boldsymbol{t}, \boldsymbol{y})$ evaluates to true in $\mathcal{I}$.

Given a mapping assertion m of form (2), with $\boldsymbol{y} = \{y_1, \ldots, y_n\}$, we compute from m a set $\mathcal{M}_m$ of mapping assertions of the form (1) as follows:

1. substitute each y_i with the term $f_i(\boldsymbol{x})$, such that $f_i \neq f_j$ for each $i, j \in \{1, \ldots, n\}$;
2. for each atom α occurring in m after the above substitution, add to $\mathcal{M}_m$ the mapping assertion $\phi(\boldsymbol{x}) \rightsquigarrow \alpha$.

For example, if m is the GLAV mapping assertion

$$T(x) \rightsquigarrow A_1(f(x)), P(f(x), y), A_2(y)$$

the above procedure returns the three mapping assertions

$$T(x) \rightsquigarrow A_1(f(x)) \qquad T(x) \rightsquigarrow P(f(x), g(x)) \qquad T(x) \rightsquigarrow A_2(g(x))$$

We denote with $\tau(m)$ the set of mapping assertions of the form (1) obtained from m through the above procedure. Given a GLAV mapping $\mathcal{M}_L$, we define $\tau(\mathcal{M}_L) = \{\tau(m) \mid m \in \mathcal{M}_L\}$[4].

Theorem 1. *Let $\mathcal{J} = \langle \mathcal{O}, \mathcal{S}, \mathcal{M}_L \rangle$ be an OBDA specification, where $\mathcal{O}$ is a DL-Lite$_R$ TBox and $\mathcal{M}_L$ is a GLAV mapping, let $\mathcal{J}_\tau = \langle \mathcal{O}, \mathcal{S}, \tau(\mathcal{M}_L) \rangle$, and q a Boolean CQ over $\mathcal{O}$. For each source database D for $\mathcal{J}$ we have that $(\mathcal{J}, D) \models q$ iff $(\mathcal{J}_\tau, D) \models q$.*

Proof. Given a database instance D, it is easy to see that $Mod_D(\mathcal{J}_\tau) \subseteq Mod_D(\mathcal{J})$. Indeed, to satisfy mapping assertions in $\mathcal{M}_L$, different existential variables can be assigned with the same object of the interpretation domain, whereas this is not possible in $\tau(\mathcal{M}_L)$, due to the fact that different existential variables are denoted in $\tau(\mathcal{M}_L)$ by terms using different functions, and due to the UNA. Thus, if $\mathcal{J} \models q$, then obviously $\mathcal{J}_\tau \models q$. For the other way round, note that for each model $\mathcal{I} \in Mod_D(\mathcal{J})$ there exists a model $\mathcal{I}_\tau \in Mod_D(\mathcal{J}_\tau)$ such that $\mathcal{I}$ satisfies all joins satisfied by $\mathcal{I}_\tau$. Therefore, since each $\mathcal{I}_\tau$ satisfies q then each $\mathcal{I}$ satisfies q, thus showing the thesis. □

For non-Boolean queries, further work is needed. For example, consider the same GLAV mapping assertion as before, an empty TBox $\mathcal{O}$, the non-Boolean CQ $q(x, y) \leftarrow A(x) \wedge P(x, y)$, and the source database $D = \{T(d)\}$. Then, $cert_D(\langle \mathcal{O}, \mathcal{S}, \{m\} \rangle) = \emptyset$, whereas $cert_D(\langle \mathcal{O}, \mathcal{S}, \tau(m) \rangle) = \{\langle f(d), g(d) \rangle\}$. For non-Boolean queries, however, it is easy to see that $cert_D(\langle \mathcal{O}, \mathcal{S}, \mathcal{M}_L \rangle) \subseteq cert_D(\langle \mathcal{O}, \mathcal{S}, \tau(\mathcal{M}_L) \rangle)$, for any GLAV mapping $\mathcal{M}_L$. More precisely, for this setting it is possible to show that in each $\boldsymbol{t} \in cert_D(\langle \mathcal{O}, \mathcal{S}, \tau(\mathcal{M}_L) \rangle)$ such that $\boldsymbol{t} \notin cert_D(\langle \mathcal{O}, \mathcal{S}, \mathcal{M}_L \rangle)$ some function symbol introduced by the transformation τ occurs. Thus, one might think to filter out all such tuples from the set $cert_D(\langle \mathcal{O}, \mathcal{S}, \tau(\mathcal{M}_L) \rangle)$ to obtain the certain answers to the query over the original GLAV system. In our example, this means dropping the tuple $\langle f(d), g(d) \rangle$. These results indeed say that query answering over an OBDA system having GLAV mapping can be reduced to query answering over an OBDA system using mapping of the form (1), where no existential variables occur. This means that systems like Mastro [23] or Ontop [12], which manage mapping assertions of the form (1), can be used (with minimal adaptations) to answer queries in the presence of GLAV mappings.

The above result has been shown for OBDA where the ontology is expressed in *DL-Lite$_R$*. Unfortunately it cannot be extended to full *DL-Lite$_A$* which includes functional roles. Indeed, it is not hard to see that the result stated in Theorem 1 does no longer hold for full *DL-Lite$_A$* ontologies. Consider for example the GLAV mapping $\mathcal{M}_L$ from a source schema $\mathcal{S}$ that contains the assertions

$$T_1(x) \rightsquigarrow P(f(x), y) \quad \text{and} \quad T_2(x) \rightsquigarrow P(f(x), y).$$

In this case, $\tau(\mathcal{M}_L)$ contains the assertions

$$T_1(x) \rightsquigarrow P(f(x), g_1(x)) \quad \text{and} \quad T_2(x) \rightsquigarrow P(f(x), g_2(x)).$$

[4] In $\mathcal{M}_L$, the sets of fresh function symbols introduced in each $\tau(m)$ are pairwise disjoint.

Assume now to have an ontology $\mathcal{O}$ that contains the axiom (funct P), and a source database $D = \{T_1(d), T_2(d)\}$. Then $Mod_D(\langle \mathcal{O}, \mathcal{S}, \tau(\mathcal{M}_L)\rangle) = \emptyset$, since the individuals $g_1(d)$ and $g_2(d)$ have to be interpreted with different objects, due to the UNA, and this leads to the violation of (funct P). On the other hand, it is not difficult to see that instead $Mod_D(\langle \mathcal{O}, \mathcal{S}, \mathcal{M}_L\rangle) \neq \emptyset$. Thus, in this case the transformation τ causes a consistent system to become inconsistent, and, as a consequence, Theorem 1 is invalidated. Of course, one might think to renounce to the UNA on the fresh skolem terms introduced by τ. This however would cause query answering to be no longer first-order rewritable [13], which, as said, is a crucial requirement for practical applicability of the OBDA approach. Indeed, [13] has shown that query answering for OBDA systems with a *DL-Lite*$_A$ ontology and GLAV mapping is NLOGSPACE-hard in data complexity.

5 Current challenges

In this section we discuss the main challenges related to OBDA that currently deserve investigation.

Query rewriting optimization. As already mentioned in Section 3, despite the theoretical low complexity of query answering within the tractable OBDA framework described Section 2, experiments carried in real-world scenarios show that the behaviour of current relational DBMS is extremely disappointing when the queries to be executed at the source database are too complex, which turns out to be the case, especially, when the cognitive distance between the ontology and the data is large. Thus, a few works focused on optimizing query rewriting with respect to the mappings in the tractable framework of Section 2. Specifically, in [45], the authors propose an optimization based on the idea of compiling the ontology into the mappings. This has the advantage of allowing ignoring redundant rewritings. In [23], the authors propose to introduce *view predicates* over the data sources, to split the mappings into *low-level mappings*, relating view predicates to SQL queries over the source database, and *high-level mappings*, relating the ontology elements to conjunctive queries over the view predicates. Hence, the rewriting process is split into two rewriting phases, one producing a union of conjunctive queries over the views, and another producing an SQL query over the data source. This allows reducing the size of the final rewriting, by optimizing the size of each conjunctive query over the views without reasoning about SQL expressions and, by adding inclusions between views, to eliminate redundant conjunctive queries within the union. As a further optimization, the authors introduce so-called *perfect mappings*, which are assertions logically entailed by the OBDA specification, allowing for handling whole subqueries as single atoms both in the ontology rewriting and in the mapping rewriting process.

Finally, in [10], the authors propose a query rewriting optimization strategy based on searching within a set of alternative equivalent rewritings, one with minimal evaluation cost when evaluated through a relational DBMS, where the cost depends both on properties of the data itself (e.g., value distribution), on the storage model (e.g., the presence of indexes), and on the DBMS optimizer's algorithm. Despite all the above mentioned efforts, the evaluation of the final rewriting by the DBMS still seems the most critical bottleneck of query answering within practical real-world OBDA scenarios.

Metamodeling and metaquerying. Recent papers point out the need of enriching conceptual ontology languages with *metamodeling* and *metaquerying* features, i.e., features for specifying and reasoning about metaclasses (also called metaconcepts) and metarelations (also called metaproperties) [4,9]. Roughly speaking, a metaclass is a class whose instances can be themselves classes, a metarelation is a relationship (or, property) between metaclasses, and a metaquery is a query possibly using metaclasses and metarelations, and whose variables may be bound to predicates. In OBDA scenarios, metamodeling and metaquerying are essential both for correctly capturing complex domains, and for performing interesting analyses, such as, for example, "find all data sources that contribute, through mappings, to the instances of C, or any of its superclasses". Recently, a new semantics, called *HOS*, has been proposed for allowing metamodeling and metaquerying over OWL 2 QL ontologies [35,34], which is both semantically adequate and exhibits nice computational properties, being answering unions of conjunctive metaqueries still AC^0 in data complexity, under certain realistic restrictions for the queried ontologies. Hence, a new challenge is now ready to be tackled, namely the problem of investigating how HOS can be combined with previous work on OBDA with dynamic ontologies [22], where also the TBox is determined by suitable mappings to a set of (relational) data sources.

Non-relational data sources. Most of the research on OBDA has focused on mappings between ontologies and data sources that either are natively relational, or can be wrapped by means of any data federation or virtualization tool, able to provide access to their contents through an SQL engine. Nevertheless, one may wonder whether the use of such a federation intermediate layer affects query answering performances and it is worth investigating query answering techniques within OBDA settings, where queries in the left-hand side of the mapping assertions are expressed in the native language of a non-relational data source. Among the wide variety of databases used within modern applications, particularly popular are the so-called NoSQL (not only SQL) databases, which are non-relational databases usually adopting one of four main data models, namely the column-family, key-value, document, and graph data models. Thus, in [8], after defining a uniform generalized framework for the access to arbitrary databases, the authors propose a query rewriting algorithm by adapting the technique used for relational databases and using relational algebra as an intermediate representation of the queries. Also, they experiment their results by implementing an OBDA system to access MongoDB [5] document databases.

OBDA methodology and tools. Devising an OBDA specification is likely to be a hard and time-consuming task. Several different competencies are required to collaborate, and in order for the collaboration to be successful, it is crucial that a well-defined methodology and appropriate tools are devised, for developing both the ontology and the mappings. As for the ontology, several methodologies have been devised (see e.g. [26]). Also, when it comes to OWL 2 ontologies, a visual graphical language, called *Graphol* [31], has been proposed and experimented in practice [5], that drastically supports the ontology construction. Thus, besides the *Protégé* ontology editor [39], ontology developers can currently use *Eddy* [32], an editor for Graphol ontolo-

[5] https://docs.mongodb.com/manual/

gies. On the contrary, the problem of devising methodologies and tools for developing mappings for OBDA is largely unexplored. Indeed, while we are not aware of any study aiming at defining a methodology for developing mappings in data integration or OBDA scenarios, schema mappings tools have been proposed (e.g., [43,37]) for supporting the specification of mappings within data integration and data exchange systems. However, none of such work is ready to be used in the OBDA scenario.

OBDA evolution. An OBDA specification is usually considered as a static piece of information. However, it is certainly crucial to investigate how to face changes over the TBox and/or the source schema. A natural assumption is to repair the mapping in such a way that the semantics of the overall system changes "as little as possible". While many approaches exist for both *ontology evolution* [48] and *database schema evolution* [44], to the best of our knowledge, no previous study has analyzed evolution in the presence of a mapping connecting an ontology to a relational data source.

Beyond data access. OBDA is the problem of accessing data through an ontology. However, the theoretical framework presented in Section 2 may offer many other challenging capabilities, among which we mention data quality assessment, instance-level update, and open data publishing. As for data quality assessment, in [19] the authors define a general framework for data consistency in OBDA, and present algorithms and complexity analysis for several relevant tasks related to the problem of checking data quality under the consistency dimension. The (instance-level) update over an OBDA framework is the capability of the framework to react to the addition, removal or change of logically implied assertions about ontology instances (aka *individuals*). Instance-level update was tackled for DL knowledge bases (see e.g., [21,28]) and, recently, a rewriting technique was proposed for `SPARQL` updates over (extended) `RDFS` knowledge bases [3]. However, to the best of our knowledge, no work has focused yet on the problem of updating the extensional level of the ontology, within the theoretical framework proposed for OBDA. Finally, a natural use of OBDA is for publishing open data. However, this requires to define a well-founded semantics of the "right open data set" to be exported, which is far from being clear at the moment.

References

1. S. Abiteboul and O. Duschka. Complexity of answering queries using materialized views. In *Proc. of PODS*, pages 254–265, 1998.
2. S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison Wesley Publ. Co., 1995.
3. A. Ahmeti, D. Calvanese, A. Polleres, and V. Savenkov. Handling inconsistencies due to class disjointness in sparql updates. In *Proc. of ESWC*, volume 9678 of *Lecture Notes in Computer Science*, pages 387–404. Springer, 2016.
4. D. Allemang and J. Hendler. *Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL*. Elsevier, 2011.
5. N. Antonioli, F. Castanò, S. Coletta, S. Grossi, D. Lembo, M. Lenzerini, A. Poggi, E. Virardi, and P. Castracane. Ontology-based data management for the italian public debt. In *Proc. of FOIS*, pages 372–385, 2014.

6. A. Artale, D. Calvanese, R. Kontchakov, and M. Zakharyaschev. The *DL-Lite* family and relations. *JAIR*, 36:1–69, 2009.
7. 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, 2nd edition, 2007.
8. E. Botoeva, D. Calvanese, B. Cogrel, M. Rezk, and G. Xiao. Obda beyond relational dbs: A study for mongodb. In *Proc. of DL*, volume 1577 of *CEUR,* `ceur-ws.org`, 2016.
9. F. Brasileiro, J. P. A. Almeida, V. A. Carvalho, and G. Guizzardi. Expressive multi-level modeling for the semantic web. In *The Semantic Web - ISWC 2016 - 15th International Semantic Web Conference, Kobe, Japan, October 17-21, 2016, Proceedings, Part I*, pages 53–69, 2016.
10. D. Bursztyn, F. Goasdoué, and I. Manolescu. Teaching an RDBMS about ontological constraints. *PVLDB*, 9(12):1161–1172, 2016.
11. A. Calì, G. Gottlob, and T. Lukasiewicz. A general Datalog-based framework for tractable query answering over ontologies. *J. of Web Semantics*, 14:57–83, 2012.
12. D. Calvanese, B. Cogrel, S. Komla-Ebri, R. Kontchakov, D. Lanti, M. Rezk, M. Rodriguez-Muro, and G. Xiao. Ontop: Answering SPARQL queries over relational databases. *Semantic Web J.*, 8(3):471–487, 2017.
13. 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.
14. 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. *JAR*, 39(3):385–429, 2007.
15. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Data complexity of query answering in description logics. *AIJ*, 195:335–360, 2013.
16. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, R. Rosati, and M. Ruzzi. Using OWL in data integration. In *Semantic Web Information Management - A Model-Based Perspective*, pages 397–424. Springer, 2009.
17. D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Description logic framework for information integration. In *Proc. of KR*, pages 2–13, 1998.
18. A. Chortaras, D. Trivela, and G. B. Stamou. Optimized query rewriting for OWL 2 QL. In *Proc. of CADE*, pages 192–206, 2011.
19. M. Console and M. Lenzerini. Data quality in ontology-based data access: The case of consistency. In *Proc. of AAAI*, pages 1020–1026, 2014.
20. S. Das, S. Sundara, and R. Cyganiak. R2RML: RDB to RDF mapping language. W3C Recommendation, W3C, Sept. 2012. Available at `http://www.w3.org/TR/r2rml/`.
21. G. De Giacomo, M. Lenzerini, A. Poggi, and R. Rosati. On instance-level update and erasure in description logic ontologies. *JLC, Special Issue on Ontology Dynamics*, 19(5):745–770, 2009.
22. F. Di Pinto, G. De Giacomo, M. Lenzerini, and R. Rosati. Ontology-based data access with dynamic TBoxes in *DL-Lite*. In *Proc. of AAAI*, 2012.
23. F. Di Pinto, D. Lembo, M. Lenzerini, R. Mancini, A. Poggi, R. Rosati, M. Ruzzi, and D. F. Savo. Optimizing query rewriting in ontology-based data access. In *Proc. of EDBT*, pages 561–572. ACM Press, 2013.
24. T. Eiter, M. Ortiz, M. Simkus, T.-K. Tran, and G. Xiao. Query rewriting for Horn-SHIQ plus rules. In *Proc. of AAAI*. AAAI Press, 2012.
25. G. Gottlob, S. Kikot, R. Kontchakov, V. V. Podolskii, T. Schwentick, and M. Zakharyaschev. The price of query rewriting in ontology-based data access. *AIJ*, 213:42–59, 2014.

26. M. Grüninger. Guide to the ontology of the Process Specification Language. In S. Staab and R. Studer, editors, *Handbook of Ontologies*, pages 575–592. Springer, 2003.
27. V. Gutiérrez-Basulto, Y. A. Ibáñez-García, R. Kontchakov, and E. V. Kostylev. Queries with negation and inequalities over lightweight ontologies. *J. of Web Semantics*, 35:184–202, 2015.
28. E. Kharlamov, D. Zheleznyakov, and D. Calvanese. Capturing model-based ontology evolution at the instance level: The case of *DL-Lite*. *JCSS*, 79(6):835–872, 2013.
29. R. Kontchakov, C. Lutz, D. Toman, F. Wolter, and M. Zakharyaschev. The combined approach to ontology-based data access. In *Proc. of IJCAI*, pages 2656–2661, 2011.
30. A. Leitsch. *The Resolution Calculus*. Springer, 1997.
31. D. Lembo, D. Pantaleone, V. Santarelli, and D. F. Savo. Easy OWL drawing with the graphol visual ontology language. In *Proc. of KR*, pages 573–576, 2016.
32. D. Lembo, D. Pantaleone, V. Santarelli, and D. F. Savo. Eddy: A graphical editor for OWL 2 ontologies. In *Proc. of IJCAI*, pages 4252–4253, 2016.
33. M. Lenzerini. Data integration: A theoretical perspective. In *Proc. of PODS*, pages 233–246, 2002.
34. M. Lenzerini, L. Lepore, and A. Poggi. Answering metaqueries over hi (OWL 2 QL) ontologies. In *Proc. of IJCAI*, pages 1174–1180, 2016.
35. M. Lenzerini, L. Lepore, and A. Poggi. A higher-order semantics for metaquerying in OWL 2 QL. In *Proc. of KR*, pages 577–580, 2016.
36. A. Y. Levy, A. O. Mendelzon, Y. Sagiv, and D. Srivastava. Answering queries using views. In *Proc. of PODS*, pages 95–104, 1995.
37. B. Marnette, G. Mecca, P. Papotti, S. Raunich, and D. Santoro. ++spicy: an opensource tool for second-generation schema mapping and data exchange. *PVLDB*, 4(12):1438–1441, 2011.
38. B. Motik, A. Fokoue, I. Horrocks, Z. Wu, C. Lutz, and B. Cuenca Grau. OWL Web Ontology Language profiles. W3C Recommendation, W3C, Oct. 2009. Available at `http://www.w3.org/TR/owl-profiles/`.
39. M. A. Musen. The Protégé project: a look back and a look forward. *AI Matters*, 1(4):4–12, 2015.
40. N. F. Noy, A. Doan, and A. Y. Halevy. Semantic integration (editorial). *AI Magazine*, 26(1):7, 2005.
41. H. Pérez-Urbina, I. Horrocks, and B. Motik. Efficient query answering for OWL 2. In *Proc. of ISWC*, volume 5823 of *LNCS*, pages 489–504. Springer, 2009.
42. 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.
43. L. Popa, Y. Velegrakis, R. J. Miller, M. A. Hernández, and R. Fagin. Translating web data. In *Proc. of VLDB*, pages 598–609, 2002.
44. E. Rahm and P. A. Bernstein. An online bibliography on schema evolution. *SIGMOD Record*, 35(4):30–31, 2006.
45. M. Rodríguez-Muro and D. Calvanese. Dependencies: Making ontology based data access work in practice. In *Proc. of AMW*, volume 749 of *CEUR,* `ceur-ws.org`, 2011.
46. R. Rosati and A. Almatelli. Improving query answering over *DL-Lite* ontologies. In *Proc. of KR*, pages 290–300, 2010.
47. J. D. Ullman. Information integration using logical views. In *Proc. of ICDT*, volume 1186 of *LNCS*, pages 19–40. Springer, 1997.
48. F. Zablith, G. Antoniou, M. D'Aquin, G. Flouris, H. Kondylakis, E. Motta, D. Plexousakis, and M. Sabou. Ontology evolution: a process-centric survey. *Knowledge Eng. Review*, 30(1):45–75, 2015.