Mapping Data to Higher-Order Description Logic Knowledge Bases

Floriana Di Pinto, Giuseppe De Giacomo, Maurizio Lenzerini, Riccardo Rosati

Dipartimento di Informatica e Sistemistica Antonio Ruberti
Sapienza Università di Roma
lastname@dis.uniroma1.it

Abstract. In this paper we introduce the notion of *mapping-based knowledge base* (MKB), to formalize those ontology-based data access (OBDA) scenarios where both the extensional and the intensional level of the ontology are determined by suitable mapping assertions involving the data sources. We study reasoning over MKBs in the context of $Hi(\textit{DL-Lite}_{\mathcal{R}})$, a higher-order version of the DL *DL-Lite*$_{\mathcal{R}}$. We show that answering queries posed to MKBs expressed in $Hi(\textit{DL-Lite}_{\mathcal{R}})$ can be done efficiently through FOL rewriting: hence, query answering can be delegated to a DBMS, as in the case of traditional OBDA systems.

1 Introduction

Ontology-based data access (OBDA) [2] is a recent application of Description Logics (DLs) that is gaining momentum. The idea behind OBDA is to use a DL ontology as a means to access a set of data sources, so as to mask the user from all application-dependent aspects of data, and to extract useful information from the sources based on a conceptual representation of the domain, expressed as a TBox in a suitable DL. In current approaches to OBDA, the intensional level of the ontology (the TBox) is fixed once for all at design time, and the mapping assertions specify how the data at the sources correspond to instances of the concepts, roles, and attibutes in the TBox. More precisely, the various mapping assertions determine a sort of virtual ABox, in which the individual objects are built out from data, and the instance assertions are specified through the relationships between the sources and the elements of the ontology.

Several OBDA projects have been carried out in the last years [9], and OBDA systems have been designed to support OBDA applications [1]. The experience gained in this work has shown that there are important aspects that are missing in current OBDA technology. In this paper, we concentrate on three such aspects.

The first aspect is related to the need of making the intensional level of the ontology more dynamic. Indeed, in real applications, the information about which are the concepts and roles that are relevant in the domain of interest is often stored in the data sources. Consider, for example, the database $\mathcal{D}$ of a motor industry shown in figure 1, storing data about different types of cars (table `T-CarTypes`), and various cars of such types (table `T-Cars`) manufactured by the firm. The key observation is that the database $\mathcal{D}$ stores information not only about the instances of concepts, but also about the concepts themselves, and their relationships. For example, table `T-CarTypes` tells us that there are four concepts in our ontology that are subconcepts of the concept `Car`, and, implicitely, tells us that they are mutually disjoint. Table `T-Cars`, on the other

T-CarTypes

Code	TypeName
T1	Coupé
T2	SUV
T3	Sedan
T4	Estate

T-Cars

NumberPlate	CarType	EngineSize	BreakPower	Color	TopSpeed
AB111	T1	2000	200	Silver	260
AF333	T2	3000	300	Black	200
BR444	T2	4000	400	Grey	220
AC222	T4	2000	125	Dark Blue	180
BN555	T3	1000	75	Light Blue	180
BP666	T1	3000	600	Red	240

Fig. 1. The database of a motor industry

hand, provides information about the instances of the various concepts, as well as other properties about such instances.

The second aspect is related to the need of metamodeling constructs in the language used to specify the ontology [4, 6]. Metamodeling allows one to treat concepts and properties as first-order citizens, and to see them as individuals that may constitute the instances of other concepts, called meta-concepts. In our example, it is convenient to introduce in the ontology the concept `Car-Type`, whose instances are exactly the subconcepts of cars stored in table `T-CarTypes`. In this way, we allow users to dynamically acquire knowledge about relevant car types through simple queries asking for the instances of the meta-concept `Car-Type`.

The third aspect deals with the need of designing tractable algorithms for query answering in OBDA systems. In [8], it is argued that, since the data sources used in OBDA systems are likely to be very large, such systems should be based on DLs that are tractable in data complexity. In particular, [8] advocates the use of the *DL-Lite* family, that allows for First-Order Logic (FOL) rewritability of (unions of) conjunctive queries. We remind the reader that in a DL enjoing FOL rewritability, query answering can be divided in two steps. In the first step, called rewriting, using the TBox only, the query q is transformed into a new FOL query q', and in the second step q' is evaluated over the ABox. The correctness of the whole method relies on the fact the answers to q' over the ABox coincide with the certain answers to q over the whole ontology. The challenge is now to design tractable query answering algorithms even in cases where the mappings relate data at the sources both to the extensional and the intensional level of the ontology, and meta-concepts and meta-roles are used in the queries. In this paper, we address the above aspects, and present the following contributions.

(*i*) We introduce the notion of *mapping-based knowledge base* (MKB) (Section 3), to formalize the situation where both the extensional and the intensional level of the ontology are determined by suitable mapping assertions involving the data sources.

(*ii*) We describe the higher-order DL $Hi(\textit{DL-Lite}_{\mathcal{R}})$ (Section 2), based on the approach presented in [5]. In that paper, it is shown how, starting from a traditional DL $\mathcal{L}$, one can define its higher-order version, called $Hi(\mathcal{L})$. Here, we apply this idea, and present $Hi(\textit{DL-Lite}_{\mathcal{R}})$, which is the higher-order version of $\textit{DL-Lite}_{\mathcal{R}}$ [3].

(*iii*) We show that answering queries posed to MKBs expressed in $Hi(\textit{DL-Lite}_{\mathcal{R}})$ can be done efficiently through FOL rewriting (Section 4). More specifically, we describe an algorithm that, given a query q over a MKB, rewrites q into a FOL query that is evaluated taking into account only the mapping assertions $\mathcal{M}_{\mathcal{A}}$ of the MKB involving the extensional level of the ontology. Hence query answering can be delegated to a DBMS, as in the case of traditional OBDA systems.

2 Higher-order *DL-Lite*$_{\mathcal{R}}$

In this section, we describe the higher-order DL $Hi(\textit{DL-Lite}_{\mathcal{R}})$, based on the approach presented in [5]. Every traditional DL $\mathcal{L}$ is characterized by a set $OP(\mathcal{L})$ of *operators*, used to form concept and role expressions, and a set of $MP(\mathcal{L})$ of *meta-predicates*, used to form assertions. Each operator and each meta-predicate have an associated arity. If symbol S has arity n, then we write S/n to denote such a symbol and its arity. For *DL-Lite*$_{\mathcal{R}}$, we have

- $OP(\textit{DL-Lite}_{\mathcal{R}}) = \{\mathit{Inv}/1,\ \mathit{Exists}/1\}$;
- $MP(\textit{DL-Lite}_{\mathcal{R}}) = \{\mathit{Inst}_C/2, \mathit{Inst}_R/3, \mathit{Isa}_C/2, \mathit{Isa}_R/2, \mathit{Disj}_C/2, \mathit{Disj}_R/2\}$.

We assume that the reader is familiar with *DL-Lite*$_{\mathcal{R}}$. Therefore, the intuitive meaning of all the above symbols should be clear. The formal specification of their semantics will be given shortly.

Syntax. We assume the existence of two disjoint, countably infinite alphabets: $\mathcal{S}$, the set of *names*, and $\mathcal{V}$, the set of *variables*. Intutively, the names in $\mathcal{S}$ are the symbols denoting the atomic elements of a $Hi(\textit{DL-Lite}_{\mathcal{R}})$ knowledge base. The building blocks of such a knowledge base are assertions, which in turn are based on terms and atoms.

We inductively define the set of *terms*, denoted by $\tau_{\textit{DL-Lite}_{\mathcal{R}}}(\mathcal{S},\mathcal{V})$, over the alphabets $\mathcal{S}$ and $\mathcal{V}$ for $Hi(\textit{DL-Lite}_{\mathcal{R}})$ as follows:

- if $E \in \mathcal{S}$ then $E \in \tau_{\textit{DL-Lite}_{\mathcal{R}}}(\mathcal{S},\mathcal{V})$;
- if $V \in \mathcal{V}$ then $V \in \tau_{\textit{DL-Lite}_{\mathcal{R}}}(\mathcal{S},\mathcal{V})$;
- if $C/n \in OP(\textit{DL-Lite}_{\mathcal{R}})$ and $t_1,\ldots,t_n \in \tau_{\textit{DL-Lite}_{\mathcal{R}}}(\mathcal{S},\mathcal{V})$ then $C(t_1,\ldots,t_n) \in \tau_{\textit{DL-Lite}_{\mathcal{R}}}(\mathcal{S},\mathcal{V})$.

Ground terms, i.e., terms without variables, are called *expressions*, and the set of expressions is denoted by $\tau_{\textit{DL-Lite}_{\mathcal{R}}}(\mathcal{S})$.

A *DL-Lite*$_{\mathcal{R}}$*-atom*, or simply *atom*, over the alphabets $\mathcal{S}$ and $\mathcal{V}$ for $Hi(\textit{DL-Lite}_{\mathcal{R}})$ is a statement of the form $M(E_1,\ldots,E_n)$ where $M \in MP(\textit{DL-Lite}_{\mathcal{R}})$, n is the arity of M, and for every $1 \leq i \leq n$, $E_i \in \tau_{\textit{DL-Lite}_{\mathcal{R}}}(\mathcal{S},\mathcal{V})$. If X is a subset of $\mathcal{V}$, a is a *DL-Lite*$_{\mathcal{R}}$-atom, and all variables appearing in a belongs to X, then a is called an X-atom in *DL-Lite*$_{\mathcal{R}}$.

Ground *DL-Lite*$_{\mathcal{R}}$-atoms, i.e., *DL-Lite*$_{\mathcal{R}}$-atoms without variables, are called *DL-Lite*$_{\mathcal{R}}$*-assertions*, or simply *assertions*. Thus, an assertion is simply an application of a meta-predicate to a set of expressions. Intuitively, an assertion is an axiom that predicates over a set of individuals, concepts or roles.

A $Hi(\textit{DL-Lite}_{\mathcal{R}})$ *knowledge base (KB)* over $\mathcal{S}$ is a set of *DL-Lite*$_{\mathcal{R}}$-assertions over $\mathcal{S}$. To agree with the usual terminology of DLs, we use the term TBox to denote a set of Isa_C, Isa_R, Disj_C and Disj_R assertions, and the term ABox to denote a set of Inst_C and Inst_R assertions.

Semantics. Our definition of semantics for $Hi(\textit{DL-Lite}_{\mathcal{R}})$ is based on the notion of interpretation structure. An *interpretation structure* is a triple $\Sigma = \langle \Delta, \mathcal{I}_c, \mathcal{I}_r \rangle$ where: (*i*) Δ is a non-empty (possibly countably infinite) set; (*ii*) $\mathcal{I}_c$ is a function that maps each $d \in \Delta$ into a subset of Δ; and (*iii*) $\mathcal{I}_r$ is a function that maps each $d \in \Delta$ into a subset of $\Delta \times \Delta$. In other words, Σ treats every element of Δ simultaneously as: (*i*) an individual; (*ii*) a unary relation, i.e., a concept, through $\mathcal{I}_c$; and (*iii*) a binary relation, i.e., a role, through $\mathcal{I}_r$.

An *interpretation* for $\mathcal{S}$ (simply called an interpretation, when $\mathcal{S}$ is clear from the context) over the interpretation structure Σ is a pair $\mathcal{I} = \langle \Sigma, \mathcal{I}_o \rangle$, where

- $\Sigma = \langle \Delta, \mathcal{I}_c, \mathcal{I}_r \rangle$ is an interpretation structure, and
- $\mathcal{I}_o$ is a function that maps:
  1. each element of $\mathcal{S}$ to a single object in Δ; and
  2. each element $C/n \in OP(\textit{DL-Lite}_{\mathcal{R}})$ to an n-ary function $C^{\mathcal{I}_o} : \Delta^n \rightarrow \Delta$ that satisfies the conditions characterizing the operator C/n. In particular, the conditions for the operators in $OP(\textit{DL-Lite}_{\mathcal{R}})$ are as follows:
     (a) for each $d_1 \in \Delta$, if $d = \mathit{Inv}^{\mathcal{I}_o}(d_1)$ then $d^{\mathcal{I}_r} = (d_1^{\mathcal{I}_r})^{-1}$;
     (b) for each $d_1 \in \Delta$, if $d = \mathit{Exists}(d_1)$ then $d^{\mathcal{I}_c} = \{e \mid \exists e_1 \text{ s.t. } \langle e, e_1 \rangle \in d_1^{\mathcal{I}_r}\}$.

We extend $\mathcal{I}_o$ to ground terms in $\tau_{\textit{DL-Lite}_{\mathcal{R}}}(\mathcal{S})$ inductively as follows: if $C/n \in OP(\textit{DL-Lite}_{\mathcal{R}})$, then $(C(t_1, \ldots, t_n))^{\mathcal{I}_o} = C^{\mathcal{I}_o}(E_1^{\mathcal{I}_o}, \ldots, E_n^{\mathcal{I}_o})$.

We now turn our attention to the interpretation of terms in $\mathit{Hi}(\textit{DL-Lite}_{\mathcal{R}})$. To interpret non-ground terms, we need assignments over interpretations, where an *assignment* μ over $\langle \Sigma, \mathcal{I}_o \rangle$ is a function $\mu : \mathcal{V} \rightarrow \Delta$. Given an interpretation $\mathcal{I} = \langle \Sigma, \mathcal{I}_o \rangle$ and an assignment μ over $\mathcal{I}$, the interpretation of terms is specified by the function $(\cdot)^{\mathcal{I}_o,\mu} : \tau_{\textit{DL-Lite}_{\mathcal{R}}}(\mathcal{S}, \mathcal{V}) \rightarrow \Delta$ defined as follows:

- if $t \in \mathcal{S}$ then $t^{\mathcal{I}_o,\mu} = t^{\mathcal{I}_o}$;
- if $t \in \mathcal{V}$ then $t^{\mathcal{I}_o,\mu} = \mu(t)$;
- if t is of the form $C(t_1, \ldots, t_n)$, then $t^{\mathcal{I}_o,\mu} = C^{\mathcal{I}_o}(t_1^{\mathcal{I}_o,\mu}, \ldots, t_n^{\mathcal{I}_o,\mu})$.

Finally, we define the semantics of atoms, by defining the notion of satisfaction of an atom with respect to an interpretation $\mathcal{I}$ and an assignment μ over $\mathcal{I}$ as follows:

- $\mathcal{I}, \mu \models \mathit{Inst}_C(E_1, E_2)$ if $E_1^{\mathcal{I}_o,\mu} \in (E_2^{\mathcal{I}_o,\mu})^{\mathcal{I}_c}$;
- $\mathcal{I}, \mu \models \mathit{Inst}_R(E_1, E_2, E_3)$ if $\langle E_1^{\mathcal{I}_o,\mu}, E_2^{\mathcal{I}_o,\mu} \rangle \in (E_3^{\mathcal{I}_o,\mu})^{\mathcal{I}_r}$;
- $\mathcal{I}, \mu \models \mathit{Isa}_C(E_1, E_2)$ if $(E_1^{\mathcal{I}_o,\mu})^{\mathcal{I}_c} \subseteq (E_2^{\mathcal{I}_o,\mu})^{\mathcal{I}_c}$;
- $\mathcal{I}, \mu \models \mathit{Isa}_R(E_1, E_2)$ if $(E_1^{\mathcal{I}_o,\mu})^{\mathcal{I}_r} \subseteq (E_2^{\mathcal{I}_o,\mu})^{\mathcal{I}_r}$;
- $\mathcal{I}, \mu \models \mathit{Disj}_C(E_1, E_2)$ if $(E_1^{\mathcal{I}_o,\mu})^{\mathcal{I}_c} \cap (E_2^{\mathcal{I}_o,\mu})^{\mathcal{I}_c} = \emptyset$;
- $\mathcal{I}, \mu \models \mathit{Disj}_R(E_1, E_2)$ if $(E_1^{\mathcal{I}_o,\mu})^{\mathcal{I}_r} \cap (E_2^{\mathcal{I}_o,\mu})^{\mathcal{I}_r} = \emptyset$.

A $\mathit{Hi}(\textit{DL-Lite}_{\mathcal{R}})$ KB $\mathcal{H}$ is satisfied by $\mathcal{I}$ if all the assertions in $\mathcal{H}$ are satisfied by $\mathcal{I}$[1]. As usual, the interpretations $\mathcal{I}$ satisfying $\mathcal{H}$ are called the *models* of $\mathcal{H}$. A $\mathit{Hi}(\textit{DL-Lite}_{\mathcal{R}})$ KB $\mathcal{H}$ is *satisfiable* if it has at least one model.

3 Mapping-based knowledge bases

As we said in the previous section, a $\mathit{Hi}(\textit{DL-Lite}_{\mathcal{R}})$ KB is simply a set of assertions. One might think of such a set of assertions as explicitly stated by the designer of the KB. This is a reasonable assumption only in those cases where the ontology is managed by an ad-hoc system, and is built from scratch for the specific application. However, in many applications, it is of interest to derive the KB directly from a set data sources, so that the assertions of the KB are defined by specific mappings to such data sources. The resulting notion will be called *mapping-based knowledge base*.

In the following, we assume that the data sources are expressed in terms of the relational data model. In other words, all the technical development presented in the rest

[1] We do not need to mention assignments here, since all assertions in $\mathcal{H}$ are ground.

of this section assumes that the set of sources to be linked to the knowledge base is one relational database. Note that this is a realistic assumption, since many data federation tools are now available that are able to wrap a set of heterogeneous sources and present them as a single relational database.

When mapping relational data sources to a knowledge base over $\mathcal{S}$, one should take into account that sources store "data values", and such data values should not be confused with the elements in $\mathcal{S}$. To face this impedance mismatch problem, [8] proposes to structure the mapping assertions in such a way that the elements of the knowledge base are denoted by terms built out from data values stored at the sources using special function symbols. Although we could in principle follow the same idea here, for the sake of simplicity, in this paper we assume that the relational data sources store directly elements of $\mathcal{S}$. Note, however, that all the results presented in the next sections easily extend to the case where mappings build complex terms for denoting the elements of the knowledge base.

We are now ready to provide the definition of mapping-based knowledge base.

Definition 1. *A $Hi(\textit{DL-Lite}_{\mathcal{R}})$* mapping-based knowledge base (MKB) *is a pair $\mathcal{K} = \langle DB, \mathcal{M} \rangle$ such that: (*i*) DB is a relational database; (*ii*) $\mathcal{M}$ is a* mapping*, i.e. a set of* mapping assertions*, each one of the form $\Phi(\boldsymbol{x}) \leadsto \psi$, where Φ is an arbitrary FOL query over DB of arity $n > 0$ with free variables $\boldsymbol{x} = \langle x_1, \ldots, x_n \rangle$, and ψ is an X-atom in $\textit{DL-Lite}_{\mathcal{R}}$, with $X = \{x_1, \ldots, x_n\}$.*

In the following, if $\mathcal{K} = \langle DB, \mathcal{M} \rangle$ is a MKB, then we denote by $\mathcal{M}_A$ the set of mapping assertions from $\mathcal{M}$ whose head predicate is either $Inst_C$ or $Inst_R$. Furthermore, we denote by $\mathcal{M}_T$ the set $\mathcal{M} \setminus \mathcal{M}_A$, i.e., the set of mapping assertions from $\mathcal{M}$ whose head predicate belongs to the set $\{Isa_C, Isa_R, Disj_C, Disj_R\}$. We call a mapping $\mathcal{M}$ an *instance-mapping* if $\mathcal{M} = \mathcal{M}_A$, i.e., if the metapredicates $Inst_C$ and $Inst_R$ are the only ones to appear in the right-hand side of the mapping assertions in $\mathcal{M}$.

In order to define the semantics of a $Hi(\textit{DL-Lite}_{\mathcal{R}})$ MKB $\mathcal{K} = \langle DB, \mathcal{M} \rangle$, we need to define when an interpretation *satisfies an assertion in $\mathcal{M}$ with respect to a database DB*. To this end, we make use of the notion of ground instance of an atom, and the notion of answer to a query over DB. Let ψ be an X-atom with $X = \{x_1, \ldots, x_n\}$, and let $\boldsymbol{v}$ be a tuple of arity n with values from DB. Then the ground instance $\psi[\boldsymbol{x}/\boldsymbol{v}]$ of ψ is the formula obtained by substituting every occurrence of x_i with v_i (for $i \in \{1, .., n\}$) in ψ. Also, if DB is a relational database, and q is a query over DB, we write $ans(\Phi, DB)$ to denote the set of answers to q over DB.

We now specify when an interpretation satisfies a mapping assertion with respect to a database. We say that an interpretation $\mathcal{I}$ satisfies the mapping assertion $\Phi(\boldsymbol{x}) \leadsto \psi$ with respect to the database DB, if for every tuple of values $\boldsymbol{v} \in ans(\Phi, DB)$, the ground atom $\psi[\boldsymbol{x}/\boldsymbol{v}]$ is satified by $\mathcal{I}$. We say that $\mathcal{I}$ is a model of $\mathcal{K} = \langle DB, \mathcal{M} \rangle$ if $\mathcal{I}$ satisfies every assertion in $\mathcal{M}$ with respect to DB.

The following example shows how $Hi(\textit{DL-Lite}_{\mathcal{R}})$ *mapping-based knowledge bases* can be used to model real world situations in a suitable manner.

Example 1. Consider the database $\mathcal{D}$ shown in the introduction. We define a $Hi(\textit{DL-Lite}_{\mathcal{R}})$ MKB $\mathcal{K}_1 = \langle \mathcal{D}, \mathcal{M} \rangle$, where the mapping $\mathcal{M}$ is defined as follows:

- M1: $\{y \mid \texttt{T-CarTypes}(x, y)\} \leadsto Isa_C(y, \texttt{Car})$
- M2: $\{(x, z) \mid \texttt{T-Cars}(x, y, t, u, v, q) \land \texttt{T-CarTypes}(y, z)\} \leadsto Inst_C(x, z)$
- M3: $\{(x, y) \mid \texttt{T-CarTypes}(z_1, x) \land \texttt{T-CarTypes}(z_2, y) \land x \neq y\} \leadsto Disj_C(x, y)$

Intuitively, M1 states that every type of car (whose name appears in the second column of table T-CarTypes, i.e. Coupé, SUV, Sedan, etc.) is a Car. M2, instead, indicates how to correctly retrieve the instances of different types of cars (e.g. car with plate number AB111 has to be retrieved as an instance of the concept Coupé, cars with plate numbers AF333 and BR444 as instances of the concept SUV, and so on). Finally, the intended meaning of assertion M3 is that different types of cars are pairwise disjoint (e.g. a Coupé is not a SUV, a SUV is not a Sedan, and so on). Obviously, mapping assertions are always written by people who know the semantics of the information stored in the database. Notice that the mapping assertions in *M* are able to model the car-types hierarchy without knowing a priori (i.e. at design-time) all the different kinds of cars that are produced by the motor industry.

Suppose now that the motor industry decides to introduce new types of cars to its car fleet, and in particular it decides to produce Campers and Caravans as well, thus extending the hierarchy. As one can imagine, these new kinds of cars share some common characteristics with the previous car types, even though not all of them. Therefore, it might be a reasonable choice for the database designers to introduce a new relational table in $\mathcal{D}$:

T-NewCars

NumberPlate	CarType	Height	Weight	EngineSize	BreakHorsePower
CM777	Camper	2,50 mt	680 Kg	4000 cc	200 bhp
CM888	Camper	2,20 mt	550 Kg	3000 cc	150 bhp
CV333	Caravan	2,30 mt	620 Kg	3000 cc	200 bhp
CV222	Caravan	2,50 mt	580 Kg	4000 cc	250 bhp

The new situation can be modeled in our framework by simply adding to $\mathcal{M}$ the following mapping assertions:

- M4: $\{y \mid \texttt{T-NewCars}(x, y, t, u, v, q)\} \rightsquigarrow \mathit{Isa}_C(y, \texttt{Car})$
- M5: $\{(x, z) \mid \texttt{T-NewCars}(x, z, t, u, v, q)\} \rightsquigarrow \mathit{Inst}_C(x, z)$
- M6: $\{(x, y) \mid \texttt{T-NewCars}(z_1, x) \land \texttt{T-NewCars}(z_2, y) \land x \neq y\} \rightsquigarrow \mathit{Disj}_C(x, y)$

where mapping M4 states that the new kinds of cars (Camper and Caravan) are Cars, the second assertion indicates how to correctly retrieve their instances, (e.g. car with plate number CM777 as an instance of Camper), and mapping M6 states that the new types of cars are pairwise disjoint (i.e. a Camper is not a Caravan).

Notice that if (instead of creating a new table) the new kinds of cars had been simply introduced into the initial table T-CarTypes (thus without modifying $\mathcal{D}$ in any way), the new concepts (Camper and Caravan) would been automatically detected at run-time by mappings M1-M3, whitout requiring any further mapping definition. □

Next, we introduce the notion of query, which in turn relies on the notion of "query atom". Intuitively, a query atom is a special kind of atom, constituted by a meta-predicate applied to a set of arguments, where each argument is either an expression or a variable. More precisely, we define the set of *q-terms* to be $\tau_{\mathit{DL\text{-}Lite}_{\mathcal{R}}}(\mathcal{S}) \cup \mathcal{V}$. We define a *query atom* as an atom constituted by the application of a meta-predicate in $MP(\mathit{DL\text{-}Lite}_{\mathcal{R}})$ to a set of q-terms, and we call a query atom *ground* if no variable occurs in it. A query atom whose meta-predicate is Inst_C or Inst_R is called an *instance-query atom*. A *higher-order conjunctive query (HCQ)* of arity n is an expression of the form $q(x_1, \ldots, x_n) \leftarrow a_1, \ldots, a_m$ where q, called the query predicate, is a symbol not in $\mathcal{S} \cup \mathcal{V}$, every x_i belongs to $\mathcal{V}$, every a_i is a (possibly non-ground) query atom, and all variables $x_1, \dots, x_n$ occur in some a_j. The variables $x_1, \dots, x_n$ are called the *free variables* (or distinguished variables) of the query, while the other variables occurring in $a_1, \dots, a_m$ are called *existential variables*. A HCQ constituted by instance atoms only is called an *instance HCQ* (IHCQ). A *higher-order union of conjunctive queries (HUCQ)* of arity n is a set of HCQs of arity n with the same query predicate. A HUCQ constituted by instance HCQs only is called an *instance HUCQ* (IHUCQ). A HCQ/HUCQ is called *Boolean* if it has no free variables.

Let $\mathcal{I}$ be an interpretation and μ an assignment over $\mathcal{I}$. A Boolean HCQ q of the form $q \leftarrow a_1, \dots, a_n$ is *satisfied* in $\mathcal{I}, \mu$ if every query atom a_i is satisfied in $\mathcal{I}, \mu$. A Boolean HUCQ Q is *satisfied* in $\mathcal{I}, \mu$ if there exists a Boolean HCQ $q \in Q$ that is satisfied in $\mathcal{I}, \mu$. A Boolean HUCQ Q is satisfied in $\mathcal{I}$, written $\mathcal{I} \models Q$, if there exists an assignment μ over $\mathcal{I}$ such that Q is satisifed in $\mathcal{I}, \mu$. Given a Boolean HUCQ Q and a $Hi(\textit{DL-Lite}_{\mathcal{R}})$ KB $\mathcal{K}$, we say that Q is *logically implied* by $\mathcal{K}$ (denoted by $\mathcal{K} \models Q$) if for each model $\mathcal{I}$ of $\mathcal{K}$ there exists an assignment μ such that Q is satisfied by $\mathcal{I}, \mu$.

Given a non-Boolean HUCQ q of the form $q(t_1, \dots, t_n) \leftarrow a_1, \dots, a_m$, a grounding substitution of q is a substitution θ such that $t_1\theta, \dots, t_n\theta$ are ground terms. We call $t_1\theta, \dots, t_n\theta$ a grounding tuple. The set of *certain answers* to q in $\mathcal{K}$ is the set of grounding tuples $t_1\theta, \dots, t_n\theta$ that make the Boolean query $q_\theta \leftarrow a_1\theta, \dots, a_n\theta$ logically implied by $\mathcal{K}$. Notice that, in general, the set of certain answers may be infinite even if the KB is finite. Therefore, it is of interest to define suitable notions of safeness, which guarantee that the set of answers is bounded. This issue, however, is beyond the scope of the present paper. Indeed, in this paper, we focus on Boolean queries only, so as to address the computation of certain answers as a decision problem.

Example 2. Let us refer to the MKB $\mathcal{K}_1 = \langle \mathcal{D}, \mathcal{M} \rangle$ of example 1. Interesting queries that can be posed to $\mathcal{K}_1$ include: (*i*) Return all the instances of *Car* manufactured by the motor industry, each one with its own type: $q(x, y) \leftarrow \textit{Inst}_C(x, y), \textit{Inst}_C(y, \textit{Car})$; (*ii*) Return all the concepts which car with plate number *'AB111'* belongs to: $q(x) \leftarrow \textit{Inst}_C('AB111', x)$.

4 Query answering

In this section, we study the problem of answering IHUCQs over $Hi(\textit{DL-Lite}_{\mathcal{R}})$ MKBs. Our query answering technique is based on query rewriting, so we will first deal with the problem of computing a perfect reformulation of a IHUCQ over a $Hi(\textit{DL-Lite}_{\mathcal{R}})$ KB. Then, we will present a query answering algorithm for MKBs based on the above perfect reformulation technique. In the following, we assume that the MKB is consistent. This does not constitute a limitation, since it is possible to show that checking consistency of a MKB can also be done through query answering, by means of techniques analogous to the ones defined for *DL-Lite*.

We start with some auxiliary definitions. Given an assertion $\alpha = \textit{Inst}_C(e_1, e_2)$, we say that e_2 occurs as a concept argument in α. Given an assertion $\alpha = \textit{Inst}_R(e_1, e_2, e_3)$, we say that e_3 occurs as a role argument in α. Given an assertion $\alpha = \textit{Isa}_C(e_1, e_2)$, we say that e_1 and e_2 occur as concept arguments in α. Given an assertion $\alpha = \textit{Isa}_R(e_1, e_2)$, we say that e_1 and e_2 occur as role arguments in α. Given an assertion $\alpha = \textit{Disj}_C(e_1, e_2)$, we say that e_1 and e_2 occur as concept arguments in α. Given an assertion $\alpha = \textit{Disj}_R(e_1, e_2)$, we say that e_1 and e_2 occur as role arguments in α.

A DL atom is an atom of the form $N(t)$ or $N(t_1, t_2)$, where N is a name and t, t_1, t_2 are either variables or names. An *extended CQ (ECQ)* is a conjunction of DL atoms, Inst_C atoms and Inst_R atoms. An extended UCQ (EUCQ) is a union of ECQs. Given an atom α, $\mathit{Pred}(\alpha)$ denotes the term appearing in predicate position in α (such a term may be either a variable or an expression). Given a TBox $\mathcal{T}$, we denote by $\mathit{Concepts}(\mathcal{T})$ the set $\{e, \mathit{Exists}(e), \mathit{Exists}(\mathit{Inv}(e)) \mid e \in \mathcal{E}$ and e occurs as a concept argument in $\mathcal{T}\}$ and denote by $\mathit{Roles}(\mathcal{T})$ the set $\{e, \mathit{Inv}(e) \mid e \in \mathcal{E}$ and e occurs as a role argument in $\mathcal{T}\}$. Given a mapping $\mathcal{M}$ and a database DB, we denote by $\mathit{Retrieve}(\mathcal{M}, DB)$ the $\mathit{Hi}(\textit{DL-Lite}_{\mathcal{R}})$ KB $\mathcal{H}$ defined as:

$$\mathcal{H} = \{\psi(\boldsymbol{t}) \mid \Phi(\boldsymbol{x}) \rightsquigarrow \psi \in \mathcal{M} \text{ and } DB \models \Phi(\boldsymbol{t})\}$$

Given an instance-mapping $\mathcal{M}$ and an ABox $\mathcal{A}$, we say that $\mathcal{A}$ is *retrievable through* $\mathcal{M}$ if there exists a database DB such that $\mathcal{A} = \mathit{Retrieve}(\mathcal{M}, DB)$.

Query rewriting. We start by providing an intuitive explanation of our rewriting technique. The basic idea is to reduce the perfect reformulation of an IHUCQ over a $\mathit{Hi}(\textit{DL-Lite}_{\mathcal{R}})$ TBox to the perfect reformulation of a standard UCQ over a $\textit{DL-Lite}_{\mathcal{R}}$ TBox, which can be done e.g. by the algorithm *PerfectRef* presented in [3]. To do so, we have to first transform a IHUCQ into a standard UCQ, actually an EUCQ. This is done through a first partial grounding of the query (through the function PMG) and then through the functions $\mathit{Normalize}$ and τ presented below. Once computed the perfect reformulation of the EUCQ, we then have to transform the EUCQ back into a IHUCQ, through the functions $\mathit{Denormalize}$ and τ^- presented below.

Given two IHCQs q, q' and a TBox $\mathcal{T}$, we say that q' is a *partial metagrounding of q with respect to $\mathcal{T}$* if $q' = \sigma(q)$ where σ is a partial substitution of the metavariables of q with the expressions occurring in $\mathcal{T}$ such that, for each metavariable x of q, either $\sigma(x) = x$ or: (i) if x occurs in a concept position in q, then $\sigma(x) \in \mathit{Concepts}(\mathcal{T})$; (ii) if x occurs in a role position in q, then $\sigma(x) \in \mathit{Roles}(\mathcal{T})$. Given an IHCQ q and a TBox $\mathcal{T}$, we denote by $\mathit{PMG}(q, \mathcal{T})$ the set of all partial metagroundings of q with respect to $\mathcal{T}$, i.e., the following IHUCQ Q:

$$Q = \{q' \mid q' \text{ is a partial metagrounding of } q \text{ with respect to } \mathcal{T}\}$$

Moreover, given a IHUCQ Q and a TBox $\mathcal{T}$, we define $\mathit{PMG}(Q, \mathcal{T})$ as the IHUCQ $\bigcup_{q \in Q} \mathit{PMG}(q, \mathcal{T})$.

Given an instance atom α, we define $\mathit{Normalize}(\alpha)$ as follows:

- if $\alpha = \mathit{Inst}_C(e_1, e_2)$ and e_2 has the form $\mathit{Exists}(e')$ where e' is an expression which is not of the form $\mathit{Inv}(e'')$, then $\mathit{Normalize}(\alpha) = \mathit{Inst}_R(e_1, \_, e')$;
- if $\alpha = \mathit{Inst}_C(e_1, e_2)$ and e_2 has the form $\mathit{Exists}(\mathit{Inv}(e'))$ where e' is any expression, then $\mathit{Normalize}(\alpha) = \mathit{Inst}_R(\_, e_1, e')$;
- if $\alpha = \mathit{Inst}_R(e_1, e_2, e_3)$ and e_3 is of the form $\mathit{Inv}^k(e')$ where $k \geq 1$ and k is an even number and e' is an expression which is not of the form $\mathit{Inv}(e'')$, then $\mathit{Normalize}(\alpha) = \mathit{Inst}_R(e_1, e_2, e')$;
- if $\alpha = \mathit{Inst}_R(e_1, e_2, e_3)$ and e_3 is of the form $\mathit{Inv}^k(e')$ where $k \geq 1$ and k is an odd number and e' is an expression which is not of the form $\mathit{Inv}(e'')$, then $\mathit{Normalize}(\alpha) = \mathit{Inst}_R(e_2, e_1, e')$.

Given an IHCQ $q \leftarrow \alpha_1, \ldots, \alpha_n$, $\mathit{Normalize}(q)$ returns the IHCQ $q \leftarrow \mathit{Normalize}(\alpha_1), \ldots, \mathit{Normalize}(\alpha_n)$. Finally, given an IHUCQ Q, we define $\mathit{Normalize}(Q)$ as $\bigcup_{q \in Q} \mathit{Normalize}(q)$.

Given an IHCQ q and an instance-mapping $\mathcal{M}$, $\mathit{Denormalize}(q, \mathcal{M})$ is the IHUCQ Q defined inductively as follows:

- $q \in Q$;
- if $q' \in Q$ and q' contains an atom α of the form $\mathit{Inst}_R(t_1, \_, t_2)$, and either $\mathit{Exists}(t_2)$ occurs in $\mathcal{M}$ or $\mathit{Exists}(x)$ (where x is a variable) occurs in $\mathcal{M}$, then the query obtained from q' by replacing α with the atom $\mathit{Inst}_C(t_1, \mathit{Exists}(t_2))$ belongs to Q;
- if $q' \in Q$ and q' contains an atom α of the form $\mathit{Inst}_R(\_, t_1, t_2)$, and either $\mathit{Exists}(\mathit{Inv}(t_2))$ occurs in $\mathcal{M}$ or $\mathit{Exists}(\mathit{Inv}(x))$ (where x is a variable) occurs in $\mathcal{M}$, then the query obtained from q' by replacing α with the atom $\mathit{Inst}_C(t_1, \mathit{Exists}(\mathit{Inv}(t_2)))$ belongs to Q;
- if $q' \in Q$ and q' contains an atom α of the form $\mathit{Inst}_R(t_1, t_2, t_3)$ and either $\mathit{Inv}(t_2)$ occurs in $\mathcal{M}$ or $\mathit{Inv}(x)$ (where x is a variable) occurs in $\mathcal{M}$, then the query obtained from q' by replacing α with the atom $\mathit{Inst}_R(t_2, t_1, \mathit{Inv}(t_3)))$ belongs to Q.

Finally, given an IHUCQ Q and a mapping $\mathcal{M}$, we define $\mathit{Denormalize}(Q, \mathcal{M})$ as $\bigcup_{q \in Q} \mathit{Denormalize}(q, \mathcal{M})$.

Given an IHUCQ Q and a TBox $\mathcal{T}$, we denote by $\mathit{PerfectRef}(Q, \mathcal{T})$ the EUCQ returned by the query rewriting algorithm for *DL-Lite*$_{\mathcal{R}}$ shown in [3].[2]

We now define the functions τ and τ^- which translate IHUCQs into EUCQs and vice versa. Given an IHCQ q and a TBox $\mathcal{T}$, $\tau(q, \mathcal{T})$ is the ECQ obtained from q as follows: (*i*) for each atom of q of the form $\mathit{Inst}_C(t_1, t_2)$, if $t_2 \in \mathit{Concepts}(\mathcal{T})$ then replace the atom with the atom $t_2(t_1)$; (*ii*) for each atom of q of the form $\mathit{Inst}_R(t_1, t_2, t_3)$, if $t_3 \in \mathit{Roles}(\mathcal{T})$ then replace the atom with the atom $t_3(t_1, t_2)$. Then, given an IHUCQ Q, we define $\tau(Q, \mathcal{T}) = \{\tau(q, \mathcal{T}) \mid q \in Q\}$.

Given a ECQ q and a TBox $\mathcal{T}$, $\tau^-(q, \mathcal{T})$ is the IHCQ obtained from q as follows: (*i*) for each atom of q of the form $t_2(t_1)$, replace the atom with the atom $\mathit{Inst}_C(t_1, t_2)$; (*ii*) for each atom of q of the form $t_3(t_1, t_2)$, replace the atom with the atom $\mathit{Inst}_R(t_1, t_2, t_3)$. Then, given an IHUCQ Q, we define $\tau^-(Q, \mathcal{T}) = \{\tau^-(q, \mathcal{T}) \mid q \in Q\}$.

We are now ready to formally define our rewriting algorithm, which takes as input a IHUCQ, a TBox and an instance-mapping, and returns a new IHUCQ.

```
ALGORITHM $\mathit{RewriteIUCQ}(Q, \mathcal{T}, \mathcal{M})$
INPUT: Boolean IHUCQ $Q$, DL-Lite$_{\mathcal{R}}$ TBox $\mathcal{T}$, instance-mapping $\mathcal{M}$
OUTPUT: Boolean IHUCQ $Q'$
    $Q_0 = \mathit{PMG}(Q, \mathcal{T})$;
    $Q_1 = \mathit{Normalize}(Q_0)$;
    $Q_2 = \tau(Q_1, \mathcal{T})$;
    $Q_3 = \mathit{PerfectRef}(Q_2, \mathcal{T})$;
    $Q_4 = \tau^-(Q_3, \mathcal{T})$;
    $Q' = \mathit{Denormalize}(Q_4, \mathcal{M})$;
    return $Q'$;
```

[2] We are actually considering a slight generalization of the algorithm, which allows for the presence of a ternary relation (Inst_R) in the query.

The IHUCQ returned by $\mathit{RewriteIUCQ}(Q,\mathcal{T},\mathcal{M})$ constitutes a perfect reformulation of the query Q with respect to the TBox $\mathcal{T}$ and the mapping $\mathcal{M}$, as formally stated by the following theorem.

Theorem 1. *Let $\mathcal{T}$ be a TBox, let $\mathcal{M}$ be an instance-mapping and let Q be a IHUCQ. Then, for every ABox $\mathcal{A}$ that is a retrievable through $\mathcal{M}$, $\mathcal{T} \cup \mathcal{A} \models Q$ iff $\mathcal{A} \models \mathit{RewriteIUCQ}(Q,\mathcal{T},\mathcal{M})$.*

Query answering. Based on the above query rewriting technique, we now present an algorithm for query answering over MKBs. Our idea is to first compute a *DL-Lite*$_{\mathcal{R}}$ TBox by evaluating the mapping assertions involving the predicates Isa_C, Isa_R, Disj_C, Disj_R over the database of the MKB; then, such a TBox is used to compute the perfect reformulation of the input IHUCQ.

To complete query answering, we now have to consider the mapping of the predicates Inst_C and Inst_R, and to reformulate the query thus obtained replacing the above predicates with the corresponding FOL queries of the mapping assertions. In this way we obtain a FOL query expressed on the database. This second rewriting step, usually called *unfolding*, can be performed by the algorithm UnfoldDB presented in [8].[3]

In the following, given a mapping $\mathcal{M}$ and a database DB, we denote by $DB_{\mathcal{M}_A}$ the database constituted by every relation R of DB such that R occurs in $\mathcal{M}_A$. Analogously, we denote by $DB_{\mathcal{M}_T}$ the database constituted by every relation R of DB such that R occurs in $\mathcal{M}_T$. We are now ready to present our query answering algorithm.

```
ALGORITHM $\mathit{Answer}(Q,\mathcal{K})$
INPUT: Boolean IHUCQ $Q$, $\mathit{Hi}$(DL-Lite$_{\mathcal{R}}$) MKB $\mathcal{K} = \langle DB, \mathcal{M}\rangle$
OUTPUT: true if $\mathcal{K} \models Q$, false otherwise
    $\mathcal{T} = \mathit{Retrieve}(\mathcal{M}_T, DB_{\mathcal{M}_T})$;
    $Q' = \mathit{RewriteIUCQ}(Q,\mathcal{T},\mathcal{M}_A)$;
    $Q'' = \mathsf{UnfoldDB}(Q',\mathcal{M}_A)$;
    if $DB_{\mathcal{M}_A} \models Q''$
    then return true
    else return false
```

The algorithm starts by retrieving the TBox from the DB through the mapping $\mathcal{M}_T$. Then, it computes the perfect reformulation of the query with respect to the retrieved TBox, and next computes the unfolding of such a query with respect to the mapping $\mathcal{M}_A$. Finally, it evaluates the query over the database.

The following property can be proved by slightly extending the proof of correctness of the algorithm UnfoldDB shown in [8].

Lemma 1. *Let $\mathcal{M}$ be an instance-mapping and let Q be a IHUCQ. Then, for every database DB, $\langle \mathcal{M}, DB\rangle \models Q$ iff $DB_{\mathcal{M}_A} \models$ UnfoldDB$(Q,\mathcal{M})$.*

[3] Here we assume that the algorithm UnfoldDB takes as input a EUCQ and an instance-mapping. This corresponds to actually considering a straightforward extension of the algorithm presented in [8] in order to deal with the presence of the ternary predicate Inst_R.

The above lemma allows us to prove correctness of the algorithm $Answer$.

Theorem 2. *Let $\mathcal{K} = \langle DB, \mathcal{M} \rangle$ be a $Hi(DL\text{-}Lite_{\mathcal{R}})$ MKB, let Q be a IHUCQ. Then, $\mathcal{K} \models Q$ iff $Answer(Q, \mathcal{K})$ returns true.*

Finally, from the algorithm $Answer$ we are able to derive the following complexity results for query answering over $Hi(DL\text{-}Lite_{\mathcal{R}})$ MKBs.

Theorem 3. *Let $\mathcal{K} = \langle DB, \mathcal{M} \rangle$ be a $Hi(DL\text{-}Lite_{\mathcal{R}})$ MKB, and let Q be a IHUCQ. Deciding whether $\mathcal{K} \models Q$ is in AC^0 with respect to the size of $DB_{\mathcal{M}_A}$, is in PTIME with respect to the size of $\mathcal{K}$, and is NP-complete with respect to the size of $\mathcal{K} \cup Q$.*

5 Conclusions

In this paper we have investigated the possibility of generating a knowledge base on the fly, while computing instance queries, from data stored in data sources through asserted mappings. A key point to obtain such a degree of flexibility is relying on higher-order description logics which blur the distinction between classes/roles at the intensional level and individuals at the extensional level. This paper is only scratching the surfaces of the immense possibilities that this approach opens. For example, we may allow the coexistence of multiple TBoxes within the same data sources, and allow the user to select which TBox to load when querying the system, possibly depending on the query, much in the spirit of [7]. The user can in principle even compose on the fly the TBox to use when answering a query. Obviously notions such as authorization views acquire an intriguing flavor in this setting (hiding intensional as well as extensional knowledge), as well as consistency, since we may even allow for contradicting assertions to coexist as long as they are not used together when performing query answering.

References

1. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, M. Rodriguez-Muro, R. Rosati, M. Ruzzi, and D. F. Savo. The Mastro system for ontology-based data access. *Semantic Web Journal*, 2(1):43–53, 2011.
2. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, and R. Rosati. Ontology-based database access. In *Proc. of SEBD 2007*, pages 324–331, 2007.
3. 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.
4. W. Chen, M. Kifer, and D. S. Warren. HILOG: A foundation for higher-order logic programming. *J. of Logic Programming*, 15(3):187–230, 1993.
5. G. De Giacomo, M. Lenzerini, and R. Rosati. Higher-order description logics for domain metamodeling. In *Proc. of AAAI 2011*, 2011.
6. J. Z. Pan and I. Horrocks. OWL FA: a metamodeling extension of OWL DL. In *Proc. of WWW 2006*, pages 1065–1066, 2006.
7. J. Parsons and Y. Wand. Emancipating instances from the tyranny of classes in information modeling. *ACM Trans. on Database Systems*, 25(2):228–268, 2000.
8. 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.
9. D. F. Savo, D. Lembo, M. Lenzerini, A. Poggi, M. Rodríguez-Muro, V. Romagnoli, M. Ruzzi, and G. Stella. MASTRO at work: Experiences on ontology-based data access. In *Proc. of DL 2010*, volume 573 of *CEUR*, `ceur-ws.org`, pages 20–31, 2010.