Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence

# Ontology-Based Data Access with Dynamic TBoxes in DL-Lite

**Floriana Di Pinto, Giuseppe De Giacomo, Maurizio Lenzerini, Riccardo Rosati**
Dipartimento di Ingegneria Informatica, Automatica e Gestionale
SAPIENZA Università di Roma - Via Ariosto 25, I-00185 Roma, Italy
*lastname@dis.uniroma1.it*

## Abstract

In this paper we introduce the notion of *mapping-based knowledge base (MKB)* to formalize the situation where both the extensional and the intensional level of the ontology are determined by suitable mappings to a set of (relational) data sources. This allows for making the intensional level of the ontology as dynamic as traditionally the extensional level is. To do so, we resort to the meta-modeling capabilities of higher-order Description Logics, which allow us to see concepts and roles as individuals, and vice versa. The challenge in this setting is to design tractable query answering algorithms. Besides the definition of MKBs, our main result is that answering instance queries posed to MKBs expressed in $Hi(\textit{DL-Lite}_{\mathcal{R}})$ can be done efficiently. In particular, we define a query rewriting technique that produces first-order (SQL) queries to be posed to the data sources.

## Introduction

Ontology-based data access (OBDA) (Calvanese et al. 2007a) 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 at design time, and the mapping assertions specify how the data at the sources correspond to instances of the concepts, roles, and attributes 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 (Savo et al. 2010), and OBDA systems have been designed to support OBDA applications (Calvanese et al. 2011).

All current works on OBDA share the idea, originally stemmed in data integration and in data exchange (Ullman 2000; Halevy 2001; Lenzerini 2002; Kolaitis 2005), that mappings are used to (virtually) retrieve the extensional information from the sources. The intensional or schema level information, i.e., the ontology of the domain of interest, remains fixed a priori, once and for all. In this paper, we challenge this preconception, and propose to virtualize both the extensional and the intensional information. Initial ideas on generating intensional specifications (in particular, relational schemas) through mappings, has been studied in the data exchange setting in (Papotti and Torlone 2009), where both data and meta-data (typically stored in system tables in relational dbms) are exchanged. In other words, we propose a setting where also the ontology itself is virtually generated through mappings. To do so, we look into two key aspects.



**T-CarTypes**

| Code | Model | Brand | Type |
|---|---|---|---|
| M1 | 1982 PONTIAC FIREBIRD | GM | Coupe |
| M2 | 1966 CADILLAC DEVILLE | GM | Sedan |
| M3 | 1973 FALCON XB GT COUPE | Ford | Coupe |
| M4 | 1967 MUSTANG SHELBY | Ford | Coupe |
| M5 | 1973 MUSTANG MACH 1 | Ford | Coupe |

**T-Cars**

| Number Plate | Code | Color | Prod Country |
|---|---|---|---|
| 111DEVIL | M2 | BLUE | U.S. |
| INTERCEPTOR | M3 | BLACK | AUSTRALIA |
| ELEANOR | M5 | YELLOW | U.S. |
| KITT | M1 | BLACK | U.S. |
| ... | ... | ... | ... |

Figure 1: The database $\mathcal{D}$

The first aspect is related to using the data sources to identify the concepts and roles that are relevant to the domain of interest. Consider, for example, the database $\mathcal{D}$ shown in Figure 1, storing data about different models and different types of cars (table `T-CarTypes`), and various cars of such types (table `T-Cars`) manufactured by motor companies. 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 several concepts in our ontology (`1973 FALCON XB GT`, `1967 MUSTANG SHELBY`, `1973 MUSTANG MACH 1`, and so on) that are subconcepts of the concept `Car`, and, implicitly, tells us that they are mutually disjoint. Table `T-Cars`, on the other hand, provides information about the instances of the various concepts, as well as other properties about them.

The second aspect is related to the need of meta-modeling

constructs in the language used to specify the ontology (Chen, Kifer, and Warren 1993; Pan and Horrocks 2006; De Giacomo, Lenzerini, and Rosati 2011). Meta-modeling 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 meta-concept `Car-Type`, and specializations of it such as `Coupe`, `Sedan`,..., whose instances include the subconcepts of cars stored in table `T-CarTypes`.

We deal with both such aspects here, focusing on the need of designing tractable algorithms for query answering, typical of OBDA systems. Indeed, as argued in (Poggi et al. 2008), the data sources used in OBDA systems are likely to be very large, hence OBDA systems should be based on DLs that are tractable in data complexity. In particular, (Poggi et al. 2008) advocates the use of the *DL-Lite* family, which allows for First-Order Logic (FOL) rewritability of (unions of) conjunctive queries. We remind the reader that in a DL enjoying 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 that the answers to $q'$ over the ABox coincide with the certain answers to $q$ over the whole ontology. The challenge in this paper is to design tractable query answering algorithms even in cases where the mappings relate data at the sources to both the extensional and the intensional level of the ontology, and meta-concepts and meta-roles are used in the queries.

In this paper, we present the following contributions.

(*i*) We introduce the notion of *mapping-based knowledge base* (MKB), 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, achieving a considerable level of flexibility. In doing so, we make use of the notion of higher-order DL, presented in (De Giacomo, Lenzerini, and Rosati 2011), where it is shown how, starting from a traditional DL $\mathcal{L}$, one can define its higher-order version $Hi(\mathcal{L})$. Here, we apply this idea, using the higher-order DL $Hi(\textit{DL-Lite}_\mathcal{R})$.

(*ii*) We introduce an extension of unions of conjunctive queries (UCQs) that makes use of the higher-order features of $Hi(\textit{DL-Lite}_\mathcal{R})$, and we devise an interesting class of such queries that enjoys nice computational properties, the so-called *instance* higher-order UCQs (IHUCQs) where we do not allow subclass and subproperty assertions in the query, though they are obviously allowed in the MKB.

(*iii*) We show that answering queries for IHUCQs posed to MKBs expressed in $Hi(\textit{DL-Lite}_\mathcal{R})$ can be done, through FOL rewriting, efficiently (more precisely, in $AC^0$) with respect to the extensional level of the data sources, i.e., those parts of the data sources which are not involved in the intensional level of the ontology. Specifically, we describe an algorithm that, given an IHUCQ $q$ over a MKB, rewrites $q$ into a FOL query that is evaluated taking into account only the mapping assertions of the MKB involving the extensional level of the ontology. Hence, query answering can be delegated to a DBMS, as in traditional OBDA systems.

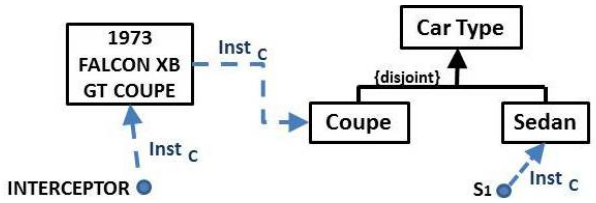


Figure 2: Example of meta-modeling in the *Cars* ontology

Overall, we show that using current OBDA technologies, and extending mappings so as that they now can talk about classes and relationships as objects (requiring higher-order features), one can realize the idea of virtualizing both the extensional and the intensional information in a way that is natural and even very effective. Indeed, our $AC^0$ result can be read as follows: the complexity of higher-order query answering through mappings remains the same as in traditional OBDA settings, when measured only with respect to the extensional level of the data sources.

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

Following (De Giacomo, Lenzerini, and Rosati 2011), we can characterize every traditional DL $\mathcal{L}$ 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: (*i*) $OP(\textit{DL-Lite}_\mathcal{R}) = \{Inv/1, Exists/1\}$; (*ii*) $MP(\textit{DL-Lite}_\mathcal{R}) = \{Inst_C/2, Inst_R/3, Isa_C/2, Isa_R/2, Disj_C/2, Disj_R/2\}$.

Let's assume the existence of two disjoint, countably infinite alphabets: $\mathcal{S}$, the set of *names*, and $\mathcal{V}$, the set of *variables*. Intuitively, 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 $e \in \mathcal{S} \cup \mathcal{V}$ then $e \in \tau_{\textit{DL-Lite}_\mathcal{R}}(\mathcal{S}, \mathcal{V})$;
- if $e \in \tau_{\textit{DL-Lite}_\mathcal{R}}(\mathcal{S}, \mathcal{V})$, and $e$ is not of the form $Inv(e')$ (where $e'$ is any term), then $Inv(e) \in \tau_{\textit{DL-Lite}_\mathcal{R}}(\mathcal{S}, \mathcal{V})$;[1]
- if $e \in \tau_{\textit{DL-Lite}_\mathcal{R}}(\mathcal{S}, \mathcal{V})$ then $Exists(e) \in \tau_{\textit{DL-Lite}_\mathcal{R}}(\mathcal{S}, \mathcal{V})$.

Intuitively, a term denotes either an atomic element, the inverse of an atomic element, or the projection of an atomic element on either the first or the second component. 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 $a(e_1), a(e_1, e_2), a(e_1, e_2, e_3)$ where $a/1, a/2, a/3 \in MP(\textit{DL-Lite}_\mathcal{R})$, and $e_1, e_2, e_3 \in \tau_{\textit{DL-Lite}_\mathcal{R}}(\mathcal{S}, \mathcal{V})$. If $X$ is

[1] Differently from (De Giacomo, Lenzerini, and Rosati 2011), we avoid the construction of terms of the form $Inv(Inv(e))$ which, as roles, are equivalent to $e$. Under this assumption, we do not have safe-range issues when dealing with queries, thus, differently form (De Giacomo, Lenzerini, and Rosati 2011), we consider here non-Boolean queries.

a subset of $\mathcal{V}$, $a$ is an atom, and all variables appearing in $a$ belong to $X$, then $a$ is called an $X$-atom.

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, which intuitively means that an assertion is an axiom that predicates over a set of individuals, concepts or roles.

A $Hi$(*DL-Lite*$_\mathcal{R}$) *knowledge base (KB)* over $\mathcal{S}$ is a finite 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.

**Example 1** The $Hi$(*DL-Lite*$_\mathcal{R}$) *KB* capturing the domain of interest in Figure 2, is defined by the following assertions:

- $Isa_C$(*Coupe, CarType*), $Isa_C$(*Sedan, CarType*),
- $Inst_C$(s, *Sedan*), $Disj_C$(*Coupe, Sedan*),
- $Inst_C$(Interceptor, *1973 FALCON XB GT COUPE*),
- $Inst_C$(*1973 FALCON XB GT COUPE*, *Coupe*).[2]

Notice how the last assertion exploits the meta-modeling capabilities of $Hi$(*DL-Lite*$_\mathcal{R}$), i.e. treating the concept *1973 FALCON XB GT COUPE* as an individual, *instance of* the concept *Coupe*, rather than a concept. □

The semantics of $Hi$(*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*) $\Delta$ 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 $\Delta$; and (*iii*) $\mathcal{I}_r$ is a function that maps each $d \in \Delta$ into a subset of $\Delta \times \Delta$. In other words, $\Sigma$ treats every element of $\Delta$ 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 $\Sigma$ 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 $\Delta$; and
  2. each element $op \in OP$(*DL-Lite*$_\mathcal{R}$) to a function $op : \Delta \rightarrow \Delta$ that satisfies the conditions characterizing the operator $op$. In particular, the conditions for the operators in $OP$(*DL-Lite*$_\mathcal{R}$) are as follows:
     (a) for each $d_1 \in \Delta$, if $d = Inv^{\mathcal{I}_o}(d_1)$ then $d^{\mathcal{I}_r} = (d_1^{\mathcal{I}_r})^{-1}$;
     (b) for each $d_1 \in \Delta$, if $d = Exists(d_1)$ then $d^{\mathcal{I}_c} = \{d \mid \exists d' \text{ s.t. } \langle d, d' \rangle \in d_1^{\mathcal{I}_r}\}$.

We now turn our attention to the interpretation of terms in $Hi$(*DL-Lite*$_\mathcal{R}$). To interpret non-ground terms, we need assignments over interpretations, where an *assignment* $\mu$ 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 $\mu$ 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: (*i*) if $e \in \mathcal{S}$ then $e^{\mathcal{I}_o,\mu} = e^{\mathcal{I}_o}$; (*ii*) if $e \in \mathcal{V}$ then $e^{\mathcal{I}_o,\mu} = \mu(e)$; (*iii*) $op(e)^{\mathcal{I}_o,\mu} = op^{\mathcal{I}_o}(e^{\mathcal{I}_o,\mu})$.

[2]Observe that we can always write such assertions in the traditional DL-notation, e.g. *Coupe* $\sqsubseteq$ *CarType*, *Sedan* $\sqsubseteq$ *CarType*, *Sedan*(s), *Coupe* $\sqsubseteq \neg$*Sedan*, ...

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 $\mu$ over $\mathcal{I}$ as follows:

- $\mathcal{I}, \mu \models Inst_C(e_1, e_2)$ if $e_1^{\mathcal{I}_o,\mu} \in (e_2^{\mathcal{I}_o,\mu})^{\mathcal{I}_c}$;
- $\mathcal{I}, \mu \models Inst_R(e_1, e_2, e_3)$ if $\langle e_1^{\mathcal{I}_o,\mu}, e_2^{\mathcal{I}_o,\mu} \rangle \in (e_3^{\mathcal{I}_o,\mu})^{\mathcal{I}_r}$;
- $\mathcal{I}, \mu \models Isa_C(e_1, e_2)$ if $(e_1^{\mathcal{I}_o,\mu})^{\mathcal{I}_c} \subseteq (e_2^{\mathcal{I}_o,\mu})^{\mathcal{I}_c}$;
- $\mathcal{I}, \mu \models Isa_R(e_1, e_2)$ if $(e_1^{\mathcal{I}_o,\mu})^{\mathcal{I}_r} \subseteq (e_2^{\mathcal{I}_o,\mu})^{\mathcal{I}_r}$;
- $\mathcal{I}, \mu \models Disj_C(e_1, e_2)$ if $(e_1^{\mathcal{I}_o,\mu})^{\mathcal{I}_c} \cap (e_2^{\mathcal{I}_o,\mu})^{\mathcal{I}_c} = \emptyset$;
- $\mathcal{I}, \mu \models Disj_R(e_1, e_2)$ if $(e_1^{\mathcal{I}_o,\mu})^{\mathcal{I}_r} \cap (e_2^{\mathcal{I}_o,\mu})^{\mathcal{I}_r} = \emptyset$.

A $Hi$(*DL-Lite*$_\mathcal{R}$) KB $\mathcal{H}$ is satisfied by $\mathcal{I}$ if all the assertions in $\mathcal{H}$ are satisfied by $\mathcal{I}$[3]. As usual, the interpretations $\mathcal{I}$ satisfying $\mathcal{H}$ are called the *models* of $\mathcal{H}$. A $Hi$(*DL-Lite*$_\mathcal{R}$) KB $\mathcal{H}$ is *satisfiable* if it has at least one model.

## Mapping-based knowledge bases

A $Hi$(*DL-Lite*$_\mathcal{R}$) KB is a finite set of (ground) assertions. As typical in DLs, even in OBDA systems, the assertions concerning the intensional knowledge (e.g., ISAs) are stated once and for all at design time. This is a reasonable assumption in many cases, such as when the ontology is managed by an ad-hoc system, and is built from scratch for the specific application. However, one can easily conceive applications where, e.g., to achieve a much higher level of flexibility, it is of interest to build on-the-fly the KB directly from a set of data sources through suitable mappings that insist both on extensional information (as usual in OBDA) and *intensional* information. In this way, *all* the assertions (not only the extensional ones) of the KB are defined by specific mappings to such data sources. The resulting notion will be called *mapping-based knowledge base* (MKB).

We assume that the data sources are expressed in terms of the relational data model. 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 MKB (or even a KB), one should take into account the impedance mismatch between sources that store "data values", and corresponding objects elements in the MKB (KB) (Poggi et al. 2008). Although we could in principle follow the idea in (Poggi et al. 2008) to address such an impedance mismatch problem, for the sake of simplicity, we will ignore this problem, assuming that the relational data sources store directly names $\mathcal{S}$ of the MKB. Note, however, that all the results presented here easily extend to the case where the impedance mismatch is dealt with as in (Poggi et al. 2008).

Formally, $Hi$(*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 $\Phi(\vec{x}) \rightsquigarrow \psi$, where $\Phi$ is an arbitrary FOL query over $DB$ of arity $n > 0$ with free variables $\vec{x} = \langle x_1, \ldots, x_n \rangle$, and $\psi$ is an $X$-atom in *DL-Lite*$_\mathcal{R}$, with $X = \{x_1, \ldots, x_n\}$.

In order to define the semantics of a $Hi$(*DL-Lite*$_\mathcal{R}$) MKB $\mathcal{K} = \langle DB, \mathcal{M} \rangle$, we need to define when an interpretation

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

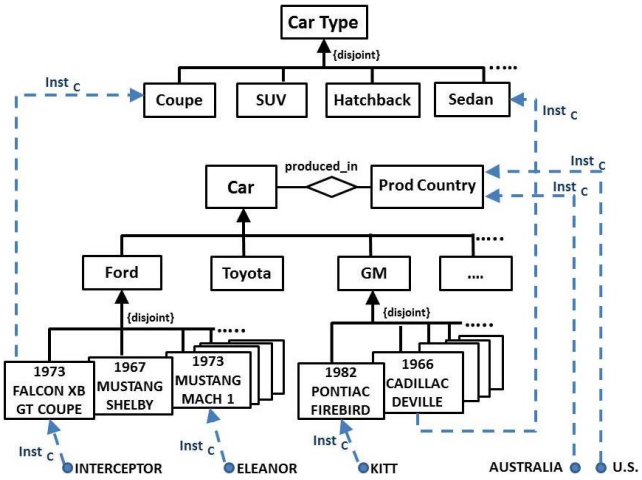


Figure 3: Representation of the *Cars* ontology

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 $\psi$ be an $X$-atom with $X = \{x_1, \ldots, x_n\}$, and let $\vec{v}$ be a tuple of arity $n$ with values from $DB$. Then the ground instance $\psi[\vec{x}/\vec{v}]$ of $\psi$ is the formula obtained by substituting every occurrence of $x_i$ with $v_i$ (for $i \in \{1, .., n\}$) in $\psi$. If $DB$ is a relational database, and $\Phi$ is a query over $DB$, we write $ans(\Phi, DB)$ to denote the set of answers to $\Phi$ over $DB$.

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

**Example 2** The pair $\mathcal{K} = \langle \mathcal{D}, \mathcal{M} \rangle$ is a $Hi(\textit{DL-Lite}_{\mathcal{R}})$ *MKB*, where $\mathcal{D}$ is the database shown in the introduction (cf. Figure 1), and $\mathcal{M}$ is the following set of mapping assertions:

- M1: $\{m \mid \texttt{T-CarTypes}(a, m, b, c)\} \rightsquigarrow \mathit{Isa}_C(m, b)$
- M2: $\{b \mid \texttt{T-CarTypes}(a, m, b, c)\} \rightsquigarrow \mathit{Isa}_C(b, \mathit{Car})$
- M3: $\{t \mid \texttt{T-CarsTypes}(a, m, b, t)\} \rightsquigarrow \mathit{Isa}_C(t, \mathit{CarType})$
- M4: $\{t1, t2 \mid \texttt{T-CarTypes(a,b,c,t1)} \wedge \texttt{T-CarTypes}(d, e, f, t2) \wedge t1 \neq t2\} \rightsquigarrow \mathit{Disj}_C(t1, t2)$
- M5: $\{m1, m2 \mid \texttt{T-CarTypes(c1,m1,a,b)} \wedge \texttt{T-CarTypes}(c2, m2, d, e) \wedge c1 \neq c2\} \rightsquigarrow \mathit{Disj}_C(m1, m2)$
- M6: $\mathit{true} \rightsquigarrow \mathit{Isa}_C(\mathit{Car}, \mathit{Exists}(\mathit{produced\_in}))$
- M7: $\mathit{true} \rightsquigarrow \mathit{Isa}_C(\mathit{ProdCountry}, \mathit{Exists}(\mathit{Inv}(\mathit{produced\_in})))$
- M8: $\{x, p \mid \texttt{T-Cars}(x, a, b, p)\} \rightsquigarrow \mathit{Inst}_R(x, p, \mathit{produced\_in})$
- M9: $\{p \mid \texttt{T-Cars}(a, b, c, p)\} \rightsquigarrow \mathit{Inst}_C(p, \mathit{ProdCountry})$
- M10: $\{m, t \mid \texttt{T-CarTypes(a,m,b,t)}\} \rightsquigarrow \mathit{Inst}_C(m, t)$
- M11: $\{x, y \mid \texttt{T-Cars}(x, c1, a, b) \wedge \texttt{T-CarTypes}(c1, y, d, e)\} \rightsquigarrow \mathit{Inst}_C(x, y)$

The intuition is that, for example, a 1973 MUSTANG MACH 1 *is a* Ford (M1); a Ford *is a* Car (M2); a Coupe *is a* CarType (M3); a Coupe *is not a* Sedan nor any other type of car (M4)[4]; and that a 1967 MUSTANG SHELBY *is not a* 1973 MUSTANG MACH 1 (M5). Moreover, mappings M6 - M7 build the static knowledge concerning the relation *produced_in*, between the concepts *Car* and *ProdCountry*; whereas M8-M9 populate the relation *produced_in* and the concept *ProdCountry*. Mapping M10, by exploiting the meta-modeling capabilities of $Hi(\textit{DL-Lite}_{\mathcal{R}})$, relates the different car models to their specific type, e.g. the 1973 FALCON XB GT COUPE as an *instance of* the concept Coupe. Finally, M11 allows to correctly retrieve the instances of different car models, e.g. the famous car *ELEANOR*, of 1974 movie "Gone in 60 seconds", is an *instance of* the concept 1973 MUSTANG MACH 1, whereas "Mad Max" police car *Interceptor* is an instance of 1973 FALCON XB GT COUPE.

Observe how these mappings allow to *dynamically* build the ontology in Figure 3, without knowing a priori any information about the different types of cars and the different models that are produced at that time by the motor companies. Indeed, all the intensional and extensional knowledge is retrieved at run-time by the mappings in $\mathcal{M}$, from the current instance of database $\mathcal{D}$. Suppose, for example, that a motor company decides to produce new car models, thus conceptually extending the hierarchy at the bottom of Figure 3. In order to deal with these changes, there are two reasonable choices. (*i*) Introduce the new car models in tuples of table T-CarTypes, without altering the database schema (e.g. $\langle$`M6, 1967 CADILLAC ELDORADO, GM, Coupe`$\rangle$). In this case, all the new information is automatically detected at run-time by the mappings in $\mathcal{M}$, and correctly introduced in the ontology. (*ii*) Introduce the information in new relational tables, thus altering the database schema. In this case, the novel situation is captured by defining suitable mapping assertions over the new tables. □

Although the example does not show it, we notice that our framework allows for using variables inside operators, in the head of mapping assertions. This is useful, for example, to extract knowledge from the database catalog: e.g. $FK(x, 2, y, 1) \rightsquigarrow \mathit{Isa}_C(\mathit{Exists}(\mathit{Inv}(x)), y)$ where FK is the table storing the database foreign keys.

## Queries

We start by introducing "query atoms". 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 $\tau_{\textit{DL-Lite}_{\mathcal{R}}}(\mathcal{S}) \cup \mathcal{V}$. Notice we do not allow for non-ground terms, except for variables themselves. We define a *query atom* as an atom constituted by the application of a meta-predicate in $MP(\textit{DL-Lite}_{\mathcal{R}})$ to a set of q-terms, and we call it *ground* if no variable occurs in it. A query atom whose meta-predicate is $\mathit{Inst}_C$ or $\mathit{Inst}_R$ is called an *instance-query atom*.

A *higher-order conjunctive query (HCQ)* 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 $\mathcal{S} \cup \mathcal{V}$, $n$ is the arity of the query, every $a_i$ is a (possibly non-ground) query atom, and all variables $x_1, \ldots, x_n$ belong to $\mathcal{V}$ and occur in some $a_j$. The variables $x_1, \ldots, x_n$ are called the *free variables* (or distinguished variables) of the query, while the

[4]Notice that, in this language (differently from the less expressive UML), it would be possible to define some suitable disjointness assertions that are specific to those pairs of CarTypes that are pairwise disjoint (rather than stating a generic disjointness, involving all the subconcepts of CarTypes).

other variables occurring in $a_1, \ldots, a_m$ are called *existential variables*. A *higher-order union of conjunctive queries (HUCQ)* is a set of HCQs of the same arity with the same query predicate. A HCQ (HUCQ) constituted by *instance atoms only* is called an *instance HCQ ,or IHCQ (IHUCQ)* .

Let $\mathcal{I}$ be an interpretation and $\mu$ an assignment over $\mathcal{I}$. A Boolean HCQ $q$ of the form $q \leftarrow a_1, \ldots, a_n$ is *satisfied* in $\mathcal{I}, \mu$ if every query atom $a_i$ is satisfied in $\mathcal{I}, \mu$. Given a Boolean HCQ $q$ and a $Hi(\textit{DL-Lite}_{\mathcal{R}})$ KB (or MKB) $\mathcal{K}$, 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 $\mu$ such that $q$ is satisfied by $\mathcal{I}, \mu$. Given a non-Boolean HCQ $q$ of the form $q(e_1, \ldots, e_n) \leftarrow a_1, \ldots, a_m$, a grounding substitution of $q$ is a substitution $\theta$ such that $e_1\theta, \ldots, e_n\theta$ are ground terms. We call $e_1\theta, \ldots, e_n\theta$ a grounding tuple. The set of *certain answers* to $q$ in $\mathcal{K}$, denoted by $cert(q, \mathcal{K})$, is the set of grounding tuples $e_1\theta, \ldots, e_n\theta$ that make the Boolean query $q_\theta \leftarrow a_1\theta, \ldots, a_n\theta$ logically implied by $\mathcal{K}$. These notions extend immediately to HUCQs.

**Example 3** Examples of HCQs that can be posed to $\mathcal{K}$ are:

1 Return all the instances of Car that are of type Coupe and were produced in Australia: $q(x) \leftarrow Inst_C(x, y), Inst_C(y, Coupe), Inst_R(x, AUSTRALIA, \textit{produced\_in})$.
2 Return all the pairs of Coupes and Sedan that were produced in the same country:
$q(x, y) \leftarrow Inst_R(x, z, \textit{produced\_in}), Inst_R(y, z, \textit{produced\_in}), Inst_C(x, v), Inst_C(y, w), Inst_C(v, Coupe), Inst_C(w, Sedan)$.
3 Return all the concepts to which a given instance (e.g. Eleanor) belongs to: $q(x) \leftarrow Inst_C(\texttt{ELEANOR}, x)$.
4 Return all the concepts whose instances are the concepts to which Eleanor belongs to:
$q(y) \leftarrow Inst_C(\texttt{ELEANOR}, x), Inst_C(x, y)$.

Observe that all of them are IHCQs. Indeed, here we concentrate on instance queries only.

## Query answering

We now study the problem of answering IHUCQs over $Hi(\textit{DL-Lite}_{\mathcal{R}})$ MKBs. Since our query answering technique is based on query rewriting, 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. [5]

We start with some auxiliary definitions. 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 meta-predicates $Inst_C$ and $Inst_R$ are the only ones to appear in the right-hand side of the mapping assertions in $\mathcal{M}$.

Then, we say that:

- $e'$ occurs as a concept argument in the atom $Inst_C(e, e')$;
- $e''$ occurs as a role argument in the atom $Inst_R(e, e', e'')$;
- $e$,$e'$ occur as concept arguments in the atom $Isa_C(e, e')$;
- $e$,$e'$ occur as role arguments in the atom $Isa_R(e, e')$;
- $e$,$e'$ occur as concept arguments in the atom $Disj_C(e, e')$;
- $e$,$e'$ occur as role arguments in the atom $Disj_R(e, e')$.

A $DL$ atom is an atom of the form $N(e)$ or $N(e_1, e_2)$, where $N$ is a name and $e, e_1, e_2$ 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 $\alpha$, $Pred(\alpha)$ denotes the term appearing in predicate position in $\alpha$ (such a term may be either a variable or an expression). Given a TBox $\mathcal{T}$, we define $Concepts(\mathcal{T}) = \{e, Exists(e), Exists(Inv(e)) \mid e \text{ occurs as a concept argument in } \mathcal{T}\}$ and define $Roles(\mathcal{T}) = \{e, Inv(e) \mid e \text{ occurs as a role argument in } \mathcal{T}\}$. Given a mapping $\mathcal{M}$ and a database $DB$, we denote by $Retrieve(\mathcal{M}, DB)$ the $Hi(\textit{DL-Lite}_{\mathcal{R}})$ KB $\mathcal{H}$ defined as:

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

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 $\mathcal{A} = Retrieve(\mathcal{M}, DB)$.

**Query rewriting.** The basic idea of our rewriting technique is to reduce the perfect reformulation of an IHUCQ over a $Hi(\textit{DL-Lite}_{\mathcal{R}})$ TBox to the perfect reformulation of a standard UCQ over a *DL-Lite*$_{\mathcal{R}}$ TBox, which can be done e.g. by the algorithm *PerfectRef* presented in (Calvanese et al. 2007b). 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 $PMG$ below, which eliminates meta-variables (i.e., variables occurring in predicate positions) from the query, and then through the functions $Normalize$ and $\tau$ presented below. Once computed the perfect reformulation of the EUCQ, the EUCQ is transformed back into a IHUCQ, by the functions $Denormalize$ and $\tau^-$ 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 $\sigma$ is a partial substitution of the meta-variables of $q$ with the expressions occurring in $\mathcal{T}$ such that, for each meta-variable $x$ of $q$, either $\sigma(x) = x$ or: (i) if $x$ occurs in a concept position in $q$, then $\sigma(x) \in Concepts(\mathcal{T})$; (ii) if $x$ occurs in a role position in $q$, then $\sigma(x) \in Roles(\mathcal{T})$. Given an IHCQ $q$ and a TBox $\mathcal{T}$, we define $PMG(q, \mathcal{T})$ as the set of all partial metagroundings of $q$ with respect to $\mathcal{T}$, i.e., the IHUCQ $Q = \{q' \mid q' \text{ is a partial metagrounding of } q \text{ w.r.t. } \mathcal{T}\}$. Moreover, given a IHUCQ $Q$ and a TBox $\mathcal{T}$, we define $PMG(Q, \mathcal{T})$ as the IHUCQ $\bigcup_{q \in Q} PMG(q, \mathcal{T})$.

The above partial metagrounding $PMG$ is a crucial aspect of our rewriting technique. Indeed, it can be shown that, even if in $Hi(\textit{DL-Lite}_{\mathcal{R}})$ the set of expressions that can be built out of a finite set of names occurring in the TBox is infinite, it is actually sufficient to ground the meta-variables of the query only on a finite set of expressions. Formally:

[5] 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*.

**Lemma 4** *Given a IHUCQ $Q$ and a TBox $\mathcal{T}$, for every ABox $\mathcal{A}$, $cert(Q, \mathcal{T} \cup \mathcal{A}) = cert(PMG(Q, \mathcal{T}), \mathcal{T} \cup \mathcal{A})$.*

*Proof (sketch).* Suppose $cert(Q, \mathcal{T} \cup \mathcal{A}) \neq cert(PMG(Q, \mathcal{T}), \mathcal{T} \cup \mathcal{A})$. First, it is straightforward to verify that $cert(Q, \mathcal{T} \cup \mathcal{A}) \supset cert(PMG(Q, \mathcal{T}), \mathcal{T} \cup \mathcal{A})$. Thus, there exists a tuple $t$ such that $t \in cert(Q, \mathcal{T} \cup \mathcal{A}) - cert(PMG(Q, \mathcal{T}), \mathcal{T} \cup \mathcal{A})$. This implies that there exists a model $\mathcal{I}$ for $\langle \mathcal{T}, \mathcal{A} \rangle$ such that $\mathcal{I} \models Q(t)$ and $\mathcal{I} \not\models PMG(Q, \mathcal{T})(t)$. Let $\Delta$ be the domain of $\mathcal{I}$. We now define the following interpretation $\mathcal{I}^{\downarrow}$ over the same domain $\Delta$. For every $d \in \Delta$:

- $d^{\mathcal{I}_0^{\downarrow}} = d^{\mathcal{I}_0}$;
- $d^{\mathcal{I}_c^{\downarrow}} = d^{\mathcal{I}_c}$ if there exists $e \in Concepts(\mathcal{T})$ such that $e^{\mathcal{I}_0} = d$, otherwise $d^{\mathcal{I}_c^{\downarrow}} = \emptyset$;
- $d^{\mathcal{I}_r^{\downarrow}} = d^{\mathcal{I}_r}$ if there exists $e \in Roles(\mathcal{T})$ such that $e^{\mathcal{I}_0} = d$, otherwise $d^{\mathcal{I}_r^{\downarrow}} = \emptyset$.

It is easy to see that $\mathcal{I}^{\downarrow}$ is a model for $\langle \mathcal{T}, \mathcal{A} \rangle$. But now, it is also immediate to verify that $\mathcal{I}^{\downarrow} \models Q(t)$ iff $\mathcal{I}^{\downarrow} \models PMG(Q, T)(t)$, and since by hypothesis $\mathcal{I} \not\models PMG(Q, T)(t)$, then $\mathcal{I}^{\downarrow} \not\models PMG(Q, T)(t)$ as well. Therefore, $\mathcal{I}^{\downarrow} \not\models Q(t)$, thus contradicting the hypothesis that $t \in cert(Q, \mathcal{T} \cup \mathcal{A})$. Consequently, the thesis follows. □

Then, given an instance atom $\alpha$, we define $Normalize(\alpha)$ as follows:

- if $\alpha = Inst_C(e_1, e_2)$ and $e_2$ has the form $Exists(e')$ where $e'$ is an expression which is not of the form $Inv(e'')$, then $Normalize(\alpha) = Inst_R(e_1, \_, e')$;
- if $\alpha = Inst_C(e_1, e_2)$ and $e_2$ has the form $Exists(Inv(e'))$ where $e'$ is any expression, then $Normalize(\alpha) = Inst_R(\_, e_1, e')$.

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

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

- $q \in Q$;
- if $q' \in Q$ and $q'$ contains an atom $\alpha$ of the form $Inst_R(e_1, \_, e_2)$, and either $Exists(e_2)$ occurs in $\mathcal{M}$ or $Exists(x)$ (where $x$ is a variable) occurs in $\mathcal{M}$, then the query obtained from $q'$ by replacing $\alpha$ with the atom $Inst_C(e_1, Exists(e_2))$ belongs to $Q$;
- if $q' \in Q$ and $q'$ contains an atom $\alpha$ of the form $Inst_R(\_, e_1, e_2)$, and either $Exists(Inv(e_2))$ occurs in $\mathcal{M}$ or $Exists(Inv(x))$ (where $x$ is a variable) occurs in $\mathcal{M}$, then the query obtained from $q'$ by replacing $\alpha$ with the atom $Inst_C(e_1, Exists(Inv(e_2)))$ belongs to $Q$;
- if $q' \in Q$ and $q'$ contains an atom $\alpha$ of the form $Inst_R(e_1, e_2, e_3)$ and either $Inv(e_2)$ occurs in $\mathcal{M}$ or $Inv(x)$ (where $x$ is a variable) occurs in $\mathcal{M}$, then the query obtained from $q'$ by replacing $\alpha$ with the atom $Inst_R(e_2, e_1, Inv(e_3)))$ belongs to $Q$.

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

Given an IHUCQ $Q$ and a TBox $\mathcal{T}$, we denote by $PerfectRef(Q, \mathcal{T})$ the EUCQ returned by the query rewriting algorithm for *DL-Lite*$_{\mathcal{R}}$ shown in (Calvanese et al. 2007b).[6] We now define the functions $\tau$ and $\tau^-$ 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 $Inst_C(e_1, e_2)$, if $e_2 \in Concepts(\mathcal{T})$ then replace the atom with the atom $e_2(e_1)$; (*ii*) for each atom of $q$ of the form $Inst_R(e_1, e_2, e_3)$, if $e_3 \in Roles(\mathcal{T})$ then replace the atom with the atom $e_3(e_1, e_2)$. 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 $e_2(e_1)$, replace the atom with the atom $Inst_C(e_1, e_2)$; (*ii*) for each atom of $q$ of the form $e_3(e_1, e_2)$, replace the atom with the atom $Inst_R(e_1, e_2, e_3)$. 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 $RewriteIHUCQ(Q, \mathcal{T}, \mathcal{M})$
INPUT: Boolean IHUCQ $Q$, *DL-Lite*$_{\mathcal{R}}$ TBox $\mathcal{T}$, instance-mapping $\mathcal{M}$
OUTPUT: Boolean IHUCQ $Q'$

$Q_0 = PMG(Q, \mathcal{T})$;
$Q_1 = Normalize(Q_0)$;
$Q_2 = \tau(Q_1, \mathcal{T})$;
$Q_3 = PerfectRef(Q_2, \mathcal{T})$;
$Q_4 = \tau^-(Q_3, \mathcal{T})$;
$Q' = Denormalize(Q_4, \mathcal{M})$;
**return** $Q'$;

The IHUCQ returned by $RewriteIHUCQ(Q, \mathcal{T}, \mathcal{M})$ 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 5** *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 $\mathcal{M}$, $cert(Q, \mathcal{T} \cup \mathcal{A}) = cert(RewriteIHUCQ(Q, \mathcal{T}, \mathcal{M}), \mathcal{A})$.*

*Proof (sketch).* The proof easily follows from Lemma 4, from correctness of the algorithm $PerfectRef$, and from the fact that the functions $Normalize$, $\tau$, $\tau^-$ and $Denormalize$ just perform equivalent transformations of the query. □

**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 reformulate the query thus obtained by replacing the above predicates with the FOL queries of the corresponding mapping assertions. In this way we obtain a FOL query expressed over the database. This second rewriting step, usually called

[6] Actually, we consider a slight generalization of that algorithm, allowing for the presence of a ternary relation ($Inst_R$) in the query.

*unfolding*, can be performed by the algorithm UnfoldDB presented in (Poggi et al. 2008).[7] In the following, given a mapping $\mathcal{M}$ and a database $DB$, we denote by $DB_{\mathcal{M}_T}$ the database constituted by every relation $R$ of $DB$ such that $R$ occurs in $\mathcal{M}_T$. Moreover, we define $DB_{\mathcal{M}_A}$ as the database $DB - DB_{\mathcal{M}_T}$ (i.e., $DB_{\mathcal{M}_A}$ is the portion of $DB$ which is not involved by the mapping $\mathcal{M}_T$). We are now ready to present our query answering algorithm.

ALGORITHM $Answer(Q, \mathcal{K})$
INPUT: IHUCQ $Q$, $Hi(\textit{DL-Lite}_{\mathcal{R}})$ MKB $\mathcal{K} = \langle DB, \mathcal{M} \rangle$
OUTPUT: $cert(Q, K)$
$\mathcal{T} = Retrieve(\mathcal{M}_T, DB_{\mathcal{M}_T})$;
$Q' = RewriteIHUCQ(Q, \mathcal{T}, \mathcal{M}_A)$;
$Q'' = \mathsf{UnfoldDB}(Q', \mathcal{M}_A)$;
**return** $IntEval(Q'', DB_{\mathcal{M}_A})$

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 (Poggi et al. 2008), and allows us to prove correctness of the algorithm $Answer$.

**Lemma 6** *Let $\mathcal{M}$ be an instance-mapping, and let $Q$ be a IHUCQ. Then, for every database $DB$,*

$cert(Q, \langle \mathcal{M}, DB \rangle) = IntEval(\mathsf{UnfoldDB}(Q, \mathcal{M}), DB_{\mathcal{M}_A})$.

**Theorem 7** *Let $\mathcal{K}$=$\langle DB, \mathcal{M} \rangle$ be a $Hi(\textit{DL-Lite}_{\mathcal{R}})$ MKB, let $Q$ be a IHUCQ, and let $U$ be the set of tuples returned by $Answer(Q, \mathcal{K})$. Then, $cert(Q, \mathcal{K}) = U$.*

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

**Theorem 8** *Let $\mathcal{K}$=$\langle DB, \mathcal{M} \rangle$ be a $Hi(\textit{DL-Lite}_{\mathcal{R}})$ MKB, let $Q$ be a IHUCQ and let $\vec{t}$ be a tuple of expressions. Deciding whether $\mathcal{K} \models Q(\vec{t})$ is in $AC^0$ with respect to the size of $DB_{\mathcal{M}_A}$, is in PTIME w.r.t. the size of $\mathcal{K}$, and is NP-complete w.r.t. the size of $\mathcal{K} \cup Q(t)$.*

## Conclusions

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 for 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 (Parsons and Wand 2000). 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.

[7] 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 (Poggi et al. 2008) in order to deal with the presence of the ternary predicate $Inst_R$.

**Acknowledgments** We thank the anonymous reviewers for their comments and Lior Limonad for interesting discussions. We acknowledge the support of EU Project FP7-ICT ACSI (257593).

## References

Calvanese, D.; De Giacomo, G.; Lembo, D.; Lenzerini, M.; Poggi, A.; and Rosati, R. 2007a. Ontology-based database access. In *Proc. of SEBD 2007*, 324–331.

Calvanese, D.; De Giacomo, G.; Lembo, D.; Lenzerini, M.; and Rosati, R. 2007b. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. of Automated Reasoning* 39(3):385–429.

Calvanese, D.; De Giacomo, G.; Lembo, D.; Lenzerini, M.; Poggi, A.; Rodriguez-Muro, M.; Rosati, R.; Ruzzi, M.; and Savo, D. F. 2011. The Mastro system for ontology-based data access. *Semantic Web J.* 2(1):43–53.

Chen, W.; Kifer, M.; and Warren, D. S. 1993. HILOG: A foundation for higher-order logic programming. *J. of Logic Programming* 15(3):187–230.

De Giacomo, G.; Lenzerini, M.; and Rosati, R. 2011. Higher-order description logics for domain metamodeling. In *Proc. of AAAI 2011*.

Halevy, A. Y. 2001. Answering queries using views: A survey. *VLDB Journal* 10(4):270–294.

Kolaitis, P. G. 2005. Schema mappings, data exchange, and metadata management. In *Proc. of PODS 2005*, 61–75.

Lenzerini, M. 2002. Data integration: A theoretical perspective. In *Proc. of PODS 2002*, 233–246.

Pan, J. Z., and Horrocks, I. 2006. OWL FA: a metamodeling extension of OWL DL. In *Proc. of WWW 2006*, 1065–1066.

Papotti, P., and Torlone, R. 2009. Schema exchange: Generic mappings for transforming data and metadata. *Data and Knowledge Engineering* 68(7):665–682.

Parsons, J., and Wand, Y. 2000. Emancipating instances from the tyranny of classes in information modeling. *ACM Trans. on Database Systems* 25(2):228–268.

Poggi, A.; Lembo, D.; Calvanese, D.; De Giacomo, G.; Lenzerini, M.; and Rosati, R. 2008. Linking data to ontologies. *J. on Data Semantics* X:133–173.

Savo, D. F.; Lembo, D.; Lenzerini, M.; Poggi, A.; Rodríguez-Muro, M.; Romagnoli, V.; Ruzzi, M.; and Stella, G. 2010. MASTRO at work: Experiences on ontology-based data access. In *Proc. of DL 2010*.

Ullman, J. D. 2000. Information integration using logical views. *Theor. Comp. Sci.* 239(2):189–210.