future internet

MDPI

Article

Acquiring Ontology Axioms through Mappings to Data Sources †

Floriana Di Pinto [1], Giuseppe De Giacomo [2], Domenico Lembo [2,*], Maurizio Lenzerini [2] and Riccardo Rosati [2]

1 Microsoft; flo_dip@hotmail.it
2 DIAG, Sapienza Università di Roma, Via Ariosto, 25, 00185 Roma, Italy; degiacomo@diag.uniroma1.it (G.D.G.); lenzerini@diag.uniroma1.it (M.L.); rosati@diag.uniroma1.it (R.R.)
* Correspondence: lembo@diag.uniroma1.it

† This paper is an extended version of the paper entitled "Ontology-Based Data Access with Dynamic TBoxes in DL-Lite", presented at AAAI 2012, Toronto, ON, Canada, 22–26 July 2012.

Received: 11 September 2019; Accepted: 5 December 2019; Published: 13 December 2019

Abstract: Although current languages used in ontology-based data access (OBDA) systems allow for mapping source data to instances of concepts and relations in the ontology, several application domains need more flexible tools for inferring knowledge from data, which are able to dynamically acquire axioms about new concepts and relations directly from the data. 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 data sources. This allows for making the intensional level of the ontology as dynamic as the extensional level traditionally is. To do so, we resort to the meta-modeling capabilities of higher-order description logics, in particular the description logic $Hi(DL\text{-}Lite_{\mathcal{R}})$, which allows seeing concepts and relations as individuals, and vice versa. The challenge in this setting is to design efficient algorithms for answering queries posed to MKBs. Besides the definition of MKBs, our main contribution is to prove that answering instance queries posed to MKBs expressed in $Hi(DL\text{-}Lite_{\mathcal{R}})$ can be done efficiently.

Keywords: ontology-based data access; higher-order description logics; DL-Lite; query answering

1. Introduction

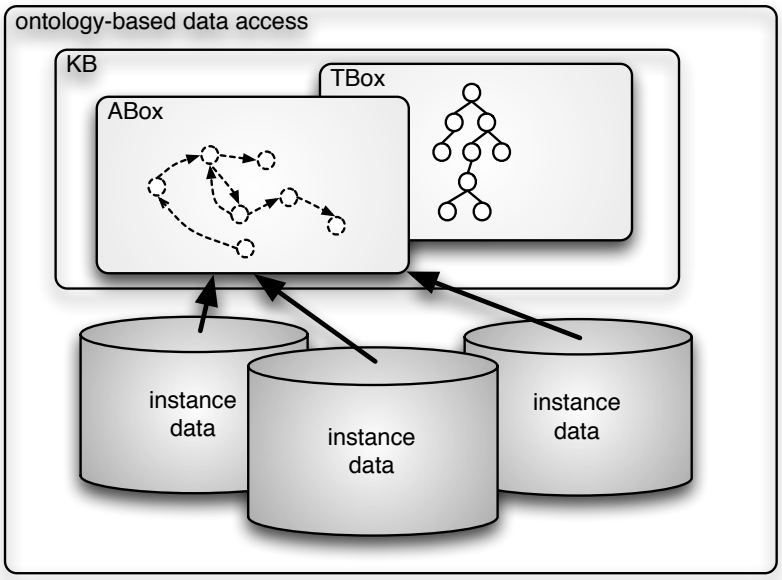
Ontology-based data access (OBDA) [1,2] is a paradigm for accessing data using a conceptual representation of the domain of interest expressed as an ontology. An OBDA system relies on a three-level architecture, consisting of the data layer, the ontology, and the mapping between the two. More in detail,

- the data layer is constituted by the existing data sources that are relevant for the organization,
- the ontology is a declarative and explicit representation of the domain of interest for the organization, formulated in a description logic (DL) [3] so as to take advantage of various reasoning capabilities in accessing data,
- the mapping is a set of declarative assertions specifying how the sources in the data layer relate to the ontology.

Several OBDA projects have been carried out in recent years [4–7], and various OBDA management systems have been designed to support OBDA applications, e.g., [8–10]. In current OBDA systems the ontology is expressed as a DL TBox, i.e., a set of assertions on the relevant concepts and roles (i.e., binary relationships between concepts) of the domain of interest, constituting the intensional level of the representation, and the mapping assertions are used to specify how the data at

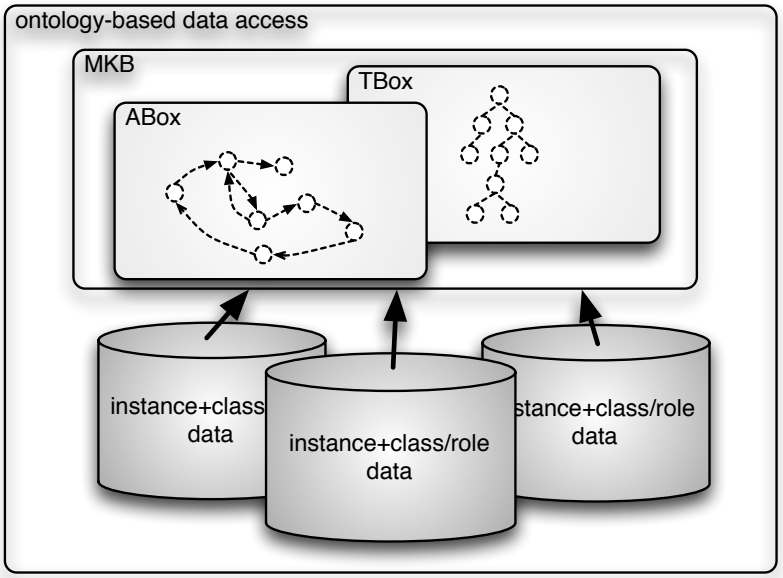
the sources correspond to the instances of the concepts and the relations, which form the extensional level of the representation. Thus, the mapping assertions, together with the source data, determine the so-called virtual ABox, in the sense that the instance assertions are not explicitly given but are specified through the relationships between the data and the elements of the TBox.

From the above observations it should be clear that current works on OBDA share the idea, originally stemmed in data integration and in data exchange [11–14], that mappings are used to (virtually) retrieve extensional information of the ontology from the sources, as shown in Figure 1, while the intensional level information, represented by the TBox, remains fixed a priori, once and for all. In this paper, we challenge this preconception and propose to adopt a virtual approach to the specification of the TBox, thus virtualizing both the extensional and the intensional level of the ontology. In other words, we propose a setting where not only the ABox but also the TBox is specified through mappings linking the data to the ontology, as illustrated in Figure 2. Our approach is based on addressing two issues.



- TBox: fixed a priori
- ABox: virtual
- Sources: data on individuals
- Mapping: assertion mapping sources to ABox axioms
- Querying: individual data of KB

Figure 1. Ontology-Based Data Access (OBDA): virtualizing extensional information.



- TBox: virtual
- ABox: virtual
- Sources: data on individuals, plus metadata on concepts and roles
- Mapping: assertions mapping sources to ABox + TBox axioms
- Querying: data and metadata of MKB

Figure 2. OBDA with dynamic TBoxes virtualizing intensional and extensional information.

The first issue is related to looking at the content of the data sources to identify concepts and roles that are relevant to the domain of interest but that have not been modeled in the TBox, because they were not known at design time. As an example, the database $\mathcal{D}$ in Figure 3 stores data about different models and different types of cars manufactured by motor companies (table `T-CarTypes`), as well as various cars of such types (table `T-Cars`). If we look carefully at the semantics of the data in $\mathcal{D}$, we realize that such database not only stores information about the instances of the concepts in the domain of interest (e.g., the first row of table `T-Cars` collects data about an instance of the concept `Car`) but contains also pieces of data denoting new concepts of the domain. In particular, table `T-CarTypes` contains data denoting concepts such as `1973 FALCON XB GT`, `1967 MUSTANG SHELBY`, `1973 MUSTANG MACH 1`, and so on. Considering the context where these concepts appear, it is not difficult to conclude that they are all mutually disjoint subconcepts of `Car`. Table `T-Cars`, on the other hand, provides information about the instances of the various concepts, as well as other properties about them (i.e., `Color`, `ProdCountry`). We observe that, in order to acquire the knowledge about the concepts mentioned in the data sources, we need a flexible mechanism that is able to map data at the sources to concepts in the ontology. Without such flexibility, the designer would be forced to manually inspect the data sources and enrich the ontology off-line.

`T-CarTypes`

Code	Model	Brand	Type
Mod1	1982 PONTIAC FIREBIRD	GM	Coupe
Mod2	1966 CADILLAC DEVILLE	GM	Sedan
Mod3	1973 FALCON XB GT COUPE	Ford	Coupe
Mod4	1967 MUSTANG SHELBY	Ford	Coupe
Mod5	1973 MUSTANG MACH 1	Ford	Coupe

`T-Cars`

Number Plate	Code	Color	Prod Country
111DEVIL	Mod2	BLUE	U.S.
INTERCEPTOR	Mod3	BLACK	AUSTRALIA
ELEANOR	Mod5	YELLOW	U.S.
KITT	Mod1	BLACK	U.S.

Figure 3. The database $\mathcal{D}$ (from [15]).

The second issue is related to the need of meta-modeling constructs in the language used to specify the ontology [16–19]. Meta-modeling allows concepts and relations to be conceived as first-order citizens and to see them as individuals that are instances of other concepts, called meta-concepts. By exploiting meta-modeling we might introduce in the ontology the meta-concept `Car-Type`, with `Coupe`, `Sedan`, etc. as its specializations. Note that the instances of such specialized concepts include the subconcepts of cars listed in the rows stored in table `T-CarTypes`. With this mechanism, the designer is allowed to specify the means for dynamically acquiring TBox axioms, through simple queries asking for the instances of the meta-concept `Car-Type`. Indeed, without the possibility of using meta-concepts in the ontology, it would be impossible to deal with the first issue mentioned above, that is exactly based on the idea of using mappings in order to transfer the knowledge fragments residing in the data sources to the TBox of the ontology. Note that for the technical development related to meta-modeling, we base our work on the approach and the results reported in [18].

In this paper, we deal with both the issue of acquiring TBox axioms from data sources and the issue of meta-modeling. In particular, we focus on designing tractable algorithms for query answering in such a setting. Indeed, looking for query answering algorithms that are tractable in data complexity is a distinguishing characteristic of OBDA systems. We follow [20], and we work with the *DL-Lite* family of DLs, which enjoys the first-order logic (FOL) rewritability properties. In a DL, enjoying such property answering (unions) of conjunctive queries can be done in two steps. The first one, called rewriting, uses the TBox axioms in order to transform the query q into a new FOL query q'. The second step evaluates q' over the ABox seen as a database.

The challenge we face in this paper is to design tractable query answering algorithms even in cases where the mapping assertions map the data at the sources to both the extensional and the intensional level of the ontology and both meta-concepts and meta-relations are used in the queries. In particular, we present the following contributions.

- We formalize the notion of *mapping-based knowledge base* (MKB), that captures the idea of acquiring the axioms of both the extensional and the intensional level of the ontology through mapping assertions linking the source data to the ontology. This mechanism allows the designer to achieve a level of flexibility that is not possible in current OBDA systems. Our formalization relies on the notion of higher-order DL, as introduced in [18]. Indeed, De Giacomo et al. [18] describe a

methodology that, starting from a traditional DL $\mathcal{L}$, allows one to define its higher-order version $Hi(\mathcal{L})$. Here, we apply this idea and make use of the higher-order DL $Hi(DL\text{-}Lite_{\mathcal{R}})$.

- We propose to query mapping-based knowledge bases by means of an extension of unions of conjunctive queries (UCQs), taking advantage of the higher-order features of $Hi(DL\text{-}Lite_{\mathcal{R}})$. In particular we define a suitable class of such queries, the so-called *instance* higher-order UCQs (IHUCQs), enjoying nice computational properties. The basic characteristic of IHUCQs is to allow higher-order features (i.e., meta-concepts and meta-properties) in the query expression but to disregard subclass and subproperty assertions in the body of the query.
- We study the problem of answering IHUCQs posed to MKBs expressed in $Hi(DL\text{-}Lite_{\mathcal{R}})$. We show that this problem is efficiently solvable by exhibiting an algorithm based on FOL rewriting. The algorithm works in AC^0 with respect to the extensional level of the data sources, i.e., the portion of the data sources that is not involved in the intensional level of the ontology. More precisely, our algorithm, given an IHUCQ q over an MKB, reformulates q into a FOL query that is evaluated taking into account only the portion of the MKB involving the extensional level of the ontology. As a consequence, query answering can be delegated to a database management system, exactly as in the traditional OBDA approach.

To sum up, the main achievement of this paper is to prove that we can extend the mapping language used in current OBDA systems so as to talk about concepts and roles as objects (requiring higher-order features), thus realizing 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 interpreted as follows: the complexity of answering higher-order queries posed on knowledge bases built using such extended mappings remains the same as in traditional OBDA settings, when measured only with respect to the extensional level of the data sources.

We observe that initial ideas on generating the intensional level of a representation through mappings was explored in the data exchange setting. In particular, Papotti and Torlone [21] propose a setting where both data and meta-data stored in relational tables typical of DBMSs, are exchanged. In the context of RDF, the standard language R2RML (https://www.w3.org/TR/r2rml/) for mapping relational databases to RDF datasets, also allows to map patterns in the relational data to assertions (e.g., subsetting) on RDF predicates. In this sense, this is in the same spirit as our proposal, that however is tailored to ontologies and not only to RDF data. In the context of ontologies, this issue is actually new. Indeed, to the best of our knowledge, the only reference dealing with generating TBoxes through mappings is [15], of which this paper is an extended version. More specifically, with respect to [15], the reader can find in the present paper additional examples and discussions, a detailed description of our technique for query answering, and complete proofs for all our results. Furthermore, we revised the analysis of combined complexity of query answering over $Hi(DL\text{-}Lite_{\mathcal{R}})$ MKBs (Theorem 3).

The rest of this paper is structured as follows. In Section 2 we recall the definition of $Hi(DL\text{-}Lite_{\mathcal{R}})$, the DL adopted in our work. In Section 3 we introduce and discuss the notion of mapping-based knowledge base. In Section 4 we illustrate the kinds of queries that we consider in this paper, and in Section 5 we present our algorithm for query answering. Section 6 concludes the paper.

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

We start by recalling some notions on DLs in general and *DL-Lite*$_{\mathcal{R}}$ in particular. Then, we provide the syntax and the semantics of $Hi(DL\text{-}Lite_{\mathcal{R}})$, i.e., the higher-order version of *DL-Lite*$_{\mathcal{R}}$.

DLs [3] are logics that represent the domain of interest in terms of *concepts*, denoting sets of objects, and *roles*, denoting binary relations between (instances of) concepts. Complex concept and role expressions are constructed starting from a set of atomic concepts and roles and by recursively applying suitable constructs.

A DL knowledge base (KB) $\mathcal{K}$ is constituted by two components, $\mathcal{T}$ and $\mathcal{A}$, i.e., $\mathcal{K} = \mathcal{T} \cup \mathcal{A}$, where:

- $\mathcal{T}$, called TBox, is the *terminological component* of $\mathcal{K}$, which contains statements representing intensional knowledge, and
- $\mathcal{A}$, called ABox, is the *assertional component* of $\mathcal{K}$, which contains assertions representing extensional knowledge.

DL-Lite$_\mathcal{R}$ is a member of the *DL-Lite* family of tractable DLs [22,23] and is the logical basis of OWL 2 QL, one of the profiles of OWL, the W3C standard for representing ontologies [24].

DL-Lite$_\mathcal{R}$ expressions are given by the following syntax:

- Concept expressions:

$$\begin{array}{rcl} B & ::= & A \mid \exists Q \\ C & ::= & B \mid \neg B \end{array}$$

- Role expressions:

$$\begin{array}{rcl} Q & ::= & P \mid P^- \\ R & ::= & Q \mid \neg Q \end{array}$$

where A is an atomic concept (i.e., a unary predicate from the signature of the knowledge base), B is a basic concept (i.e, an atomic concept A or an existential restriction on a role $\exists Q$, which denotes individuals occurring in the first component of the role Q, which is called the *domain* of Q), C is a general concept (i.e., a basic concept B or its negation $\neg B$), P is an atomic role (i.e., a binary predicate from the signature of the knowledge base), Q is a basic role (i.e., an atomic role P or its inverse P^-), and R is a general role (i.e., a basic role Q or its negation $\neg Q$). The domain of P^- (i.e., $\exists P^-$) is also called the *range* of P.

A TBox in *DL-Lite*$_\mathcal{R}$ is a finite set of assertions in form:

$$\begin{array}{ll} B \sqsubseteq C & (\textit{concept inclusion assertion}) \\ Q \sqsubseteq R & (\textit{role inclusion assertion}) \end{array}$$

When C and R above assume the form $\neg B'$ and $\neg Q'$, respectively, the above inclusions are called negative inclusions and allow us to specify *disjointness* between concepts or between roles, respectively. Otherwise, the above inclusions are called positive inclusions and allow us to specify *is_a* relations between concepts or roles, respectively.

An Abox in *DL-Lite*$_\mathcal{R}$ is a finite set of membership assertions (i.e., facts) of the form:

$$\begin{array}{ll} A(a) & (\textit{concept membership assertion}) \\ P(a,b) & (\textit{role mebership assertion}) \end{array}$$

where a and b are constants, that is, names for individuals (i.e., predicates of arity 0 from the signature of the knowledge base).

We are now ready to describe the higher-order DL $Hi(\textit{DL-Lite}_\mathcal{R})$. We start by observing that, as discussed in [18], every traditional DL $\mathcal{L}$ can be 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. Given a symbol T, we write T/n to denote that T has arity n. For *DL-Lite*$_\mathcal{R}$, we simply have:

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

provided that *Inv* and *Exists* take only a role as argument, $\textit{Inst}_C$ takes an individual as first argument and a concept as second argument, $\textit{Inst}_R$ takes an individual as first and second argument and a role

as third argument, Isa_C and $Disj_C$ take two concepts as arguments, and Isa_R and $Disj_R$ take two roles as arguments.

Example 1. *Consider the following DL-Lite$_\mathcal{R}$ knowledge base:*

$$\begin{array}{lcl} Ford & \sqsubseteq & Car \\ \exists produced\_in & \sqsubseteq & Car \\ \exists produced\_in^- & \sqsubseteq & ProdCountry \\ Ford(\text{INTERCEPTOR}) & & \\ produced\_in(\text{INTERCEPTOR}, \text{AUSTRALIA}\ ) & & \end{array}$$

In words, the knowledge base says that Ford is a concept that specializes the concept Car, produced_in is a role typed on Car and ProdCountry, which is a concept denoting production countries. More precisely, the domain of produced_in can be instantiated only with instances of Car, as stated by the second inclusion, whereas the range of produced_in can be instantiated only with instances of ProdCountry, as stated by the third inclusion. The above knowledge base can be reformulated in the high-order style syntax for DL-Lite$_\mathcal{R}$ as follows:

$$\begin{array}{l} Isa_C(Ford, Car) \\ Isa_C(Exists(\text{produced\_in}), \text{Car}) \\ Isa_C(Exists(Inv(\text{produced\_in})), \text{ProdCountry}) \\ Inst_C(\text{INTERCEPTOR}, Ford) \\ Inst_R(\text{INTERCEPTOR}, \text{AUSTRALIA}, produced\_in) \end{array}$$

We finally note that in both syntaxes given above, the first three assertions constitute the TBox of the KB, whereas the last two assertions constitute the ABox of the KB.

Let us turn our attention to the definition of the syntax of $Hi(DL\text{-}Lite_\mathcal{R})$. Hereinafter we assume the existence of two disjoint, countably infinite alphabets: $\mathcal{S}$, the set of predicates, also called *names* and $\mathcal{V}$, the set of *variables*. Intuitively, the names in $\mathcal{S}$ are the symbols denoting the atomic elements of a $Hi(DL\text{-}Lite_\mathcal{R})$ 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 $\tau_{DL\text{-}Lite_\mathcal{R}}(\mathcal{S}, \mathcal{V})$, over the alphabets $\mathcal{S}$ and $\mathcal{V}$ for $Hi(DL\text{-}Lite_\mathcal{R})$ as follows:

- if $e \in \mathcal{S} \cup \mathcal{V}$ then $e \in \tau_{DL\text{-}Lite_\mathcal{R}}(\mathcal{S}, \mathcal{V})$;
- if $e \in \tau_{DL\text{-}Lite_\mathcal{R}}(\mathcal{S}, \mathcal{V})$, and e is not of the form $Inv(e')$ (where e' is any term), then $Inv(e) \in \tau_{DL\text{-}Lite_\mathcal{R}}(\mathcal{S}, \mathcal{V})$ (Differently from [18], 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 [18], we consider here non-Boolean queries.);
- if $e \in \tau_{DL\text{-}Lite_\mathcal{R}}(\mathcal{S}, \mathcal{V})$ then $Exists(e) \in \tau_{DL\text{-}Lite_\mathcal{R}}(\mathcal{S}, \mathcal{V})$.

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.

Example 2. *If the names car, produced_in belong to the alphabet $\mathcal{S}$, then the following are $Hi(DL\text{-}Lite_\mathcal{R})$ terms: Car, $Inv(produced\_in)$, $Exists(Inv(produced\_in))$, which, intuitively, denote the concept representing the set of cars, the role produced_in, and the concept representing those individuals (e.g., countries) where something is produced.*

Ground terms, i.e., terms without variables, are called *expressions*, and the set of expressions is denoted by $\tau_{DL\text{-}Lite_\mathcal{R}}(\mathcal{S})$. The terms in the example above are all expressions.

A $Hi(DL\text{-}Lite_\mathcal{R})$-*atom*, or simply *atom*, over the alphabets $\mathcal{S}$ and $\mathcal{V}$ for $Hi(DL\text{-}Lite_\mathcal{R})$ is a statement of the form $p_1(e_1, e_2)$ or $p_2(e_1, e_2, e_3)$ where $p_1/2$, $p_2/3$ belong to $MP(DL\text{-}Lite_\mathcal{R})$, and $e_1, e_2, e_3 \in$

$\tau_{DL\text{-}Lite_{\mathcal{R}}}(\mathcal{S}, \mathcal{V})$. Note that $Hi(DL\text{-}Lite_{\mathcal{R}})$-atoms (and even terms) are not required to respect the proviso we have introduced before to limit the usage of predicates from $OP(DL\text{-}Lite_{\mathcal{R}})$ and $MP(DL\text{-}Lite_{\mathcal{R}})$ so that through them it is possible to express (only) $DL\text{-}Lite_{\mathcal{R}}$ knowledge bases. For instance, a $Inst_C(C, D)$ and $Isa_C(C, E)$ are both legal $Hi(DL\text{-}Lite_{\mathcal{R}})$-atoms, that is, differently from $DL\text{-}Lite_{\mathcal{R}}$ we can instantiate a concept with another concept. If X is 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 $Hi(DL\text{-}Lite_{\mathcal{R}})$-atoms, i.e., $Hi(DL\text{-}Lite_{\mathcal{R}})$-atoms without variables, are called $Hi(DL\text{-}Lite_{\mathcal{R}})$-*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 $Hi(DL\text{-}Lite_{\mathcal{R}})$ *KB* over $\mathcal{S}$ is a finite set of $Hi(DL\text{-}Lite_{\mathcal{R}})$-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, also called intensional assertions, and the term ABox to denote a set of $Inst_C$ and $Inst_R$ assertions, also called extensional assertions. Obviously, a $DL\text{-}Lite_{\mathcal{R}}$ KB is also (a special case of) an $Hi(DL\text{-}Lite_{\mathcal{R}})$ KB.

Example 3. *In Figure 4 we provide a graphical representation of a* $Hi(DL\text{-}Lite_{\mathcal{R}})$ KB *modeling (a portion of) the domain of cars that we use throughout the paper in our examples. The ontology is drawn in Graphol, a diagrammatic language that allows OWL ontologies to be graphically specified [25,26]. In Graphol, as in ER diagrams, concepts and roles are denoted by rectangles and diamonds, respectively, and solid directed arrows represent inclusions. The full hexagon denotes at the same time the union of the concepts connected to it through a dashed line and the disjunction of such concepts. Finally, each octagon represents an individual, and each arrow labeled* instanceOf *from and element A to an element B specifies that A is an instance of B. In words, the ontology is saying that Coupe and Sedan are specific types of Cars (i.e., they are sub-concepts of CarType), and that they are disjoint. In addition,* 1973_FALCON_XB_GT_COUPE *is an instance of* Coupe *(in this respect it can also be seen as an individual) and at the same time it is a concept that specializes Ford (i.e., it is a subconcept of Ford) and has* INTERCEPTOR *as an instance. Finally, Ford is a subconcept of Car, and* s_1 *is an instance of Sedan.*

Below we provide the above ontology through formulas, using the higher order notation given in this section:

$$
\begin{aligned}
&Inst_C(\text{INTERCEPTOR}, \text{1973\_FALCON\_XB\_GT\_COUPE})\\
&Inst_C(\text{1973\_FALCON\_XB\_GT\_COUPE}, Coupe)\\
&Inst_C(s_1, Sedan)\\
&Isa_C(Coupe, CarType)\\
&Isa_C(Sedan, CarType)\\
&Isa_C(Ford, Car)\\
&Isa_C(\text{1973\_FALCON\_XB\_GT\_COUPE}, Ford)\\
&Disj_C(Coupe, Sedan)
\end{aligned}
$$

Notice how the second assertion exploits the meta-modeling capabilities of $Hi(DL\text{-}Lite_{\mathcal{R}})$*, i.e., it treats the concept* 1973_FALCON_XB_GT_COUPE *as an individual* instance of *the concept* Coupe*, rather than as a concept (which is instead the way the penultimate assertion refers to it).*

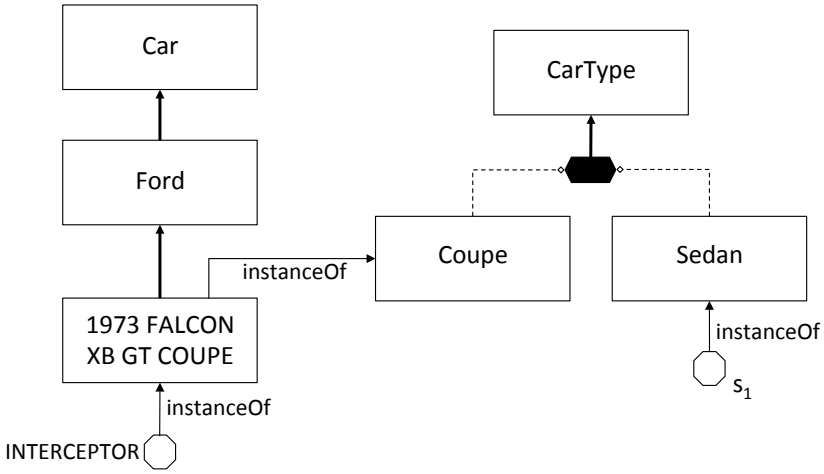


Figure 4. Example of meta-modeling in the *Cars* ontology.

The semantics of $Hi(DL\text{-}Lite_{\mathcal{R}})$ 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 (see Figure 5):

- Δ is a non-empty, possibly countably infinite, set;
- $\mathcal{I}_c$ is a function that maps each $d \in \Delta$ into a subset of Δ; and
- $\mathcal{I}_r$ 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:

- an individual;
- a unary relation, i.e., a concept, through $\mathcal{I}_c$; and
- a binary relation, i.e., a role, through $\mathcal{I}_r$.

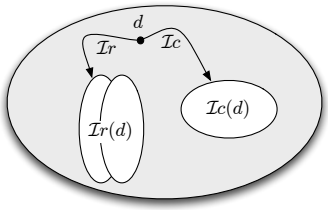


Figure 5. Interpretation structures for $Hi(DL\text{-}Lite_{\mathcal{R}})$.

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:
  - each element of $\mathcal{S}$ to a single object in Δ; and
  - each element $op \in OP(DL\text{-}Lite_{\mathcal{R}})$ to a function $op^{\mathcal{I}_o} : \Delta \rightarrow \Delta$ that satisfies the conditions characterizing the operator op. In particular, the conditions for the operators in $OP(DL\text{-}Lite_{\mathcal{R}})$ are as follows:
    - for each $d_1, d_2 \in \Delta$ such that $d_2 = Inv^{\mathcal{I}_o}(d_1)$, we have that $d_2^{\mathcal{I}_r} = (d_1^{\mathcal{I}_r})^{-1}$, where $(d_1^{\mathcal{I}_r})^{-1}$ is the inverse of the relation $d_1^{\mathcal{I}_r}$, and
    - for each $d_1, d_2 \in \Delta$ such that $d_2 = Exists(d_1)$ we have that $d_2^{\mathcal{I}_c} = \{o \mid$ there exists o' such that $\langle o, o' \rangle \in d_1^{\mathcal{I}_r}\}$.

We now turn our attention to the interpretation of terms in $Hi(DL\text{-}Lite_{\mathcal{R}})$. 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 $(\cdot)^{\mathcal{I}_o,\mu} : \tau_{DL\text{-}Lite_{\mathcal{R}}}(\mathcal{S}, \mathcal{V}) \rightarrow \Delta$ defined as follows:

- if $e \in \mathcal{S}$ then $e^{\mathcal{I}_o,\mu} = e^{\mathcal{I}_o}$;
- if $e \in \mathcal{V}$ then $e^{\mathcal{I}_o,\mu} = \mu(e)$;
- $op(e)^{\mathcal{I}_o,\mu} = op^{\mathcal{I}_o}(e^{\mathcal{I}_o,\mu})$.

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:

- $\mathcal{I}, \mu \models Inst_C(e_1, e_2)$ 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 $Hi(DL\text{-}Lite_{\mathcal{R}})$ KB $\mathcal{H}$ is satisfied by $\mathcal{I}$ if all the assertions in $\mathcal{H}$ are satisfied by $\mathcal{I}$ (We do not need to mention assignments here, since all assertions in $\mathcal{H}$ are ground.). As usual, the interpretations $\mathcal{I}$ satisfying $\mathcal{H}$ are called the *models* of $\mathcal{H}$. A $Hi(DL\text{-}Lite_{\mathcal{R}})$ KB $\mathcal{H}$ is *satisfiable* if it has at least one model.

3. Mapping-Based Knowledge Bases

In a traditional OBDA system, the axioms concerning the intensional level of the representation are stated once and for all at design time. This is a reasonable assumption in many cases, for example when the ontology is built from scratch for a specific application. However, there are application domains where, with the goal of achieving a higher level of flexibility, it is convenient 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. The result is that *all* the axioms (not only the ones relative to the extensional level, like in current OBDA systems) of the knowledge base are defined by mappings to such data sources. This is exactly the idea behind the notion of *mapping-based knowledge base* (MKB), that we define in detail in this section.

In what follows, we assume that the data sources are structured as a relational database. We observe that this does not hamper the generality of our work, since existing data federation tools support the designer in wrapping a set of heterogeneous data sources so as to present them as a single relational database. In addition, we assume that the relational data sources store directly the symbols in $\mathcal{S}$ and in particular the names of elements used in the MKB. In other words, here we ignore the problem of the impedance mismatch between sources that store "data values" and the MKB that contains ontology elements [20]. Note, however, that all the results presented in this paper can be extended to the case where the impedance mismatch is dealt with as in [20].

Definition 1. *A* $Hi(DL\text{-}Lite_{\mathcal{R}})$ mapping-based knowledge base (MKB) *is a pair* $\mathcal{K} = \langle DB, \mathcal{M} \rangle$ *such that:*

- *DB is a relational database;*
- $\mathcal{M}$ *is a* mapping, *i.e., a set of* mapping assertions, *each one of the form* $\Phi(\vec{x}) \rightsquigarrow \psi$, *where* Φ *is an arbitrary FOL query over DB of arity* $n \geq 0$ *with free variables* $\vec{x} = \langle x_1, \ldots, x_n \rangle$, *and* ψ *is an X-atom in* $DL\text{-}Lite_{\mathcal{R}}$, *with* $X = \{x_1, \ldots, x_n\}$.

Note that when the arity n of Φ is 0, then Φ is a boolean query and ψ is a ground atom. In particular, when the Φ is the query *true*, then the mapping assertion $\Phi(\vec{x}) \rightsquigarrow \psi$ corresponds to the $Hi(DL\text{-}Lite_{\mathcal{R}})$ assertion constituted by the ground atom ψ. In this way, we can easily specify through $\mathcal{M}$ also $Hi(DL\text{-}Lite_{\mathcal{R}})$ assertions that are "static", i.e., that do not depend on the current source database instance, and thus we do not need to include a different component in the formalization of an MKB for such static assertions.

We now turn to the semantics of a $Hi(DL\text{-}Lite_{\mathcal{R}})$ MKB $\mathcal{K} = \langle DB, \mathcal{M} \rangle$. We start by defining when an interpretation satisfies an assertion in $\mathcal{M}$ with respect to the source data 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 $\vec{v}$ be a tuple of arity n with values from DB. Then the ground instance $\psi[\vec{x}/\vec{v}]$ of ψ is the formula obtained by substituting every occurrence of x_i with v_i (for $i \in \{1, .., n\}$) in ψ. If DB is a relational database, and Φ is a query over DB, we write $ans(\Phi, DB)$ to denote the set of answers to Φ over DB. With this notion at hand, we are ready to present the following definition.

Definition 2. *An interpretation $\mathcal{I}$* satisfies *a mapping assertion $\Phi(\vec{x}) \rightsquigarrow \psi$ with respect to a 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 $\mathcal{I}$. The interpretation $\mathcal{I}$ is called a* model *of $\mathcal{K} = \langle DB, \mathcal{M} \rangle$ if it satisfies every assertion in $\mathcal{M}$ with respect to DB.*

We now present an example of MKB, with the goal of illustrating how such a notion can capture real world situations by introducing a notable flexibility in modeling the domain of interest.

Example 4. *Consider the $Hi(DL\text{-}Lite_{\mathcal{R}})$ MKB $\mathcal{K} = \langle \mathcal{D}, \mathcal{M} \rangle$, where $\mathcal{D}$ is the database shown in the introduction (cf. Figure 3) and $\mathcal{M}$ is the following set of mapping assertions:*

- *M1:* $\{m, b \mid \texttt{T-CarTypes}(a, m, b, c)\} \rightsquigarrow Isa_C(m, b)$
- *M2:* $\{b \mid \texttt{T-CarTypes}(a, m, b, c)\} \rightsquigarrow Isa_C(b, \text{Car})$
- *M3:* $\{t \mid \texttt{T-CarTypes}(a, m, b, t)\} \rightsquigarrow Isa_C(t, \text{CarType})$
- *M4:* $\{t1, t2 \mid \texttt{T-CarTypes}(a, b, c, t1) \wedge \texttt{T-CarTypes}(d, e, f, t2) \wedge t1 \neq t2\} \rightsquigarrow Disj_C(t1, t2)$
- *M5:* $\{m1, m2 \mid \texttt{T-CarTypes}(c1, m1, a, b) \wedge \texttt{T-CarTypes}(c2, m2, d, e) \wedge m1 \neq m2\} \rightsquigarrow Disj_C(m1, m2)$
- *M6:* $true \rightsquigarrow Isa_C(\text{Car}, Exists(\text{produced\_in}))$
- *M7:* $true \rightsquigarrow Isa_C(\text{ProdCountry}, Exists(Inv(\text{produced\_in})))$
- *M8:* $true \rightsquigarrow Isa_C(Exists(\text{produced\_in}), \text{Car})$
- *M9:* $true \rightsquigarrow Isa_C(Exists(Inv(\text{produced\_in})), \text{ProdCountry})$
- *M10:* $\{x, p \mid \texttt{T-Cars}(x, a, b, p)\} \rightsquigarrow Inst_R(x, p, \text{produced\_in})$
- *M11:* $\{p \mid \texttt{T-Cars}(a, b, c, p)\} \rightsquigarrow Inst_C(p, \text{ProdCountry})$
- *M12:* $\{m, t \mid \texttt{T-CarTypes}(a, m, b, t)\} \rightsquigarrow Inst_C(m, t)$
- *M13:* $\{x, y \mid \texttt{T-Cars}(x, c1, a, b) \wedge \texttt{T-CarTypes}(c1, y, d, e)\} \rightsquigarrow Inst_C(x, y)$

M1 asserts that for every tuple t of the `T-CarTypes` *table, the value appearing in the second column of t denotes a subconcept of the concept denoted by the value in the third column of the same tuple. Thus, for example, considering the fifth tuple t_5 of* `T-CarTypes`*, M1 states that 1973 MUSTANG MACH 1 is a subconcept of the concept Ford. M2 asserts that every value appearing in the third column of* `T-CarTypes` *is a subconcept of* `Car`*. For example, by referring to t_5 again, such tuple states that Ford is a subconcept of* `Car`*. Analogously, M3 asserts that every value appearing in the fourth column of* `T-CarTypes` *is a subconcept of* `CarType`*. For example, t_5 states that Coupe is a subconcept of* `CarType`*. M4 (resp., M5) asserts that the values appearing in the fifth column (resp., second column) of* `T-CarTypes` *denote concepts that are pairwise disjoint. M6–M9 assert properties about the relation* produced_in*, between the concepts* Car *and* ProdCountry*. M10 populates the relation* produced_in*, and M11 does the same for the concept* ProdCountry*. Mapping M12 exploits the meta-modeling capabilities of $Hi(DL\text{-}Lite_{\mathcal{R}})$ and relates the different car models to their specific type. For example, looking at tuple t_5 again, we can infer by M12 that 1973 FALCON XB GT COUPE is an* instance

of *the concept Coupe. Note that M1 asserted that 1973 FALCON XB GT COUPE is a concept, and therefore, we are taking advantage of the possibility provided by Hi(DL-Lite*$_{\mathcal{R}}$*) of defining a concept to be an instance of another concept (a metaconcept). Finally, M13 allows us to correctly assign the instances stored in the* `T-Cars` *table to the concepts corresponding to the different car models. For example, through this mapping we can infer that the "Mad Max" police car* INTERCEPTOR *is an instance of 1973 FALCON XB GT COUPE (see the second tuple of* `T-Cars`*), the famous car* ELEANOR *of movie "Gone in 60 s" is an instance of the concept 1973 MUSTANG MACH 1 (third tuple), and the "Supercar"* KITT *is an* instance of *1982 PONTIAC FIREBIRD (fourth tuple).*

We hope that the above example clarifies the potential of MKBs in acquiring ontology axioms in a flexible way. In particular, let us observe how the domain ontology is *dynamically* built through the mapping, even though no information about the different types of cars and the different models produced by the motor companies was available at design time. Indeed, the mappings in $\mathcal{M}$ retrieve at run-time both intensional and extensional knowledge from the current database instance. Suppose, for example, that a motor company, say GM, decides to produce cars of new model, say 1967 CADILLAC ELDORADO. Given the structure of the database and its intended usage, the natural thing to do for the organization is to add suitable tuples (e.g., $\langle$`Mod6, 1967 CADILLAC ELDORADO, GM, Coupe`$\rangle$) in the `T-CarTypes` table. In our approach, the new information is automatically detected at run-time by the mappings in $\mathcal{M}$ and correctly introduced in the ontology. So, instead of manually changing the ontology and "re-compiling it" at design time, the new concept is dynamically captured at run-time.

In Figure 6, using the Graphol language, we provide the graphical representation of the ontology dynamically constructed from the database $\mathcal{D}$ used in the example (In addition to the Graphol constructs already used in Figure 4, in Figure 6 we also use a diamond to denote a role (namely *produced_in*), and a blank and a full box connected to such role through a dashed line in order to denote its domain and range, respectively. Moreover, the solid double arrowed line denotes a mutual inclusion between concepts (i.e, an equivalence). By using it, we say, for instance, that *Car* is equivalent to the domain of *produced_in*, that is, every car is produced somewhere and only cars are produced (somewhere). It is analogous for the range of *produced_in* (cf. mapping assertion M6–M9).).

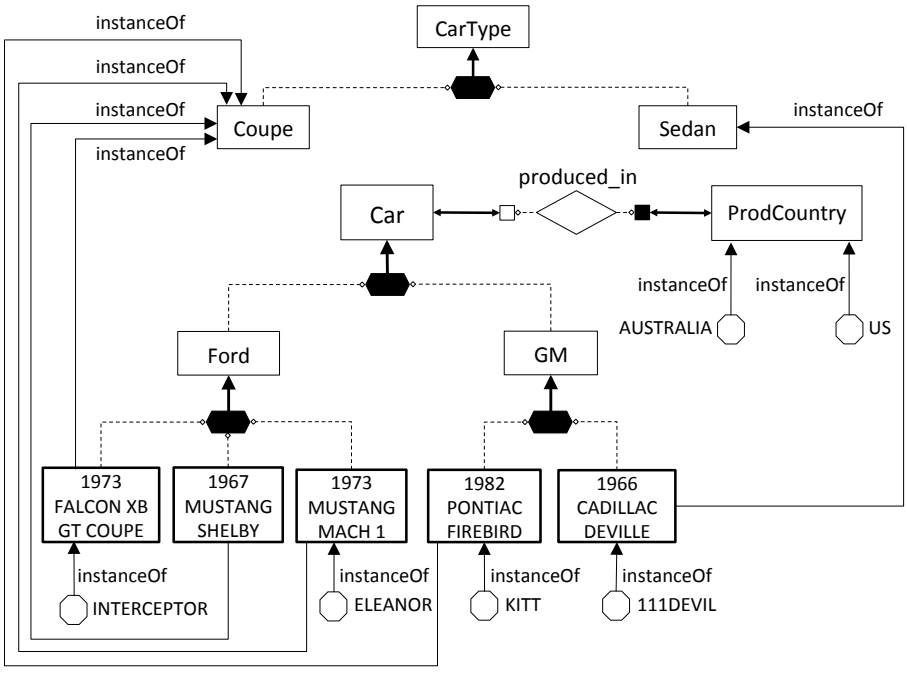


Figure 6. Representation of the *Cars* ontology.

We also want to add that, although the Example 4 does not show it, our framework allows variables to be used inside operators, in the right-hand side of mapping assertions. This is useful, in particular, to extract knowledge from the database catalog. For example, if FK is the database table storing information about foreign keys between binary tables, and such binary tables correspond to roles in the ontology, the mapping assertion

$$FK(x, 2, y, 1) \rightsquigarrow Isa_C(Exists(Inv(x)), y)$$

allows us to transfer the foreign key property at the level of the ontology, by correctly representing every foreign key as an inclusion between the corresponding roles.

4. Queries

In this section we describe the class of queries that we consider in this paper. We start by introducing "query atoms". Intuitively, a query atom is an atom constituted by a meta-predicate applied to a set of arguments, where each argument is either an expression or a variable. The precise definition relies on the notion of *q-terms*: a q-term is any element of the set $\tau_{DL\text{-}Lite_{\mathcal{R}}}(\mathcal{S}) \cup \mathcal{V}$. Thus a q-term is either an expression in $DL\text{-}Lite_{\mathcal{R}}$ or a variable. In other words, we do not allow for non-ground terms in queries, except for variables themselves. We are now ready to define the notion of *query atom*. A query atom is an atom constituted by the application of a meta-predicate in $MP(DL\text{-}Lite_{\mathcal{R}})$ to a set of q-terms. A query atom is called *ground* if no variable occurs in it and is called an *instance-query atom* if its meta-predicate is $Inst_C$ or $Inst_R$. The definition of the syntax of the class of queries that we are interested in is as follows.

Definition 3. *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 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.*

Notice that an HCQ corresponds to an HUCQ formed by a single query. A HCQ is called *Boolean* if it has no free variables. It is analogous for an HUCQ.

An HCQ (HUCQ) constituted by *instance-query atoms only* is called an *instance HCQ or IHCQ (IHUCQ)*.

We now turn our attention to the semantics of queries. Let $\mathcal{I}$ be an interpretation and μ 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 $Hi(DL\text{-}Lite_{\mathcal{R}})$ 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 μ 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 θ such that $e_1\theta, \ldots, e_n\theta$ are ground terms. We call $e_1\theta, \ldots, e_n\theta$ a grounding tuple.

Definition 4. *Given a $Hi(DL\text{-}Lite_{\mathcal{R}})$ KB $\mathcal{K}$ and a non-Boolean HCQ q of the form $q(e_1, \ldots, e_n) \leftarrow a_1, \ldots, a_m$, 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 $q_\theta \leftarrow a_1\theta, \ldots, a_m\theta$ logically implied by $\mathcal{K}$.*

These notions extend immediately to HUCQs.

Example 5. *We illustrate some examples of HCQs that can be posed to the MKB $\mathcal{K}$ of Example 4.*

(i) *Compute the instances of* `Ford` *that were produced in Australia and are of type* `Coupe`*; note that an instance x of* `Ford` *of certain type T is an instance of a car model y such that y is an instance of T (this in fact holds for all instances of* `Car`*, not only for instances of* `Ford`*). It follows that the correct query expression is as follows:*

$$q(x) \leftarrow Inst_C(x, Ford), Inst_C(x, y), Inst_C(y, Coupe), Inst_R(x, \text{AUSTRALIA}, \text{produced\_in}).$$

(ii) *Compute the pairs of cars, one of type Coupe, and one of type Sedan that were produced in the same country:*

$$\begin{aligned} q(x, y) \leftarrow\ & Inst_R(x, z, \text{produced\_in}), Inst_R(y, z, \text{produced\_in}), \\ & Inst_C(x, v), Inst_C(y, w), Inst_C(v, Coupe), Inst_C(w, Sedan). \end{aligned}$$

(iii) *Compute all the concepts in the ontology to which a given object (e.g., Eleanor) belongs to:*

$$q(x) \leftarrow Inst_C(\text{ELEANOR}, x).$$

(iv) *Compute all the concepts in the ontology whose instances are the concepts to which Eleanor or Kitt belong to:*

$$q(y) \leftarrow Inst_C(\text{ELEANOR}, x), Inst_C(x, y) \cup Inst_C(\text{KITT}, x), Inst_C(x, y).$$

We observe that all queries in the above example are actually IHCUQs. This is the class of queries that we deal with in the next section.

5. Query Answering

In this section we study how to answer IHUCQs over $Hi(DL\text{-}Lite_{\mathcal{R}})$ MKBs. In the following we consider only consistent MKBs, i.e., MKBs that have at least one model. This is indeed not a limitation, considering that consistency of a MKB can be checked through query answering, by means of methods similar to those used for checking consistency of *DL-Lite* KBs [23].

Before delving into the details of our technique, we introduce some useful definitions. In what follows, we refer to a MKB $\mathcal{K} = \langle DB, \mathcal{M} \rangle$.

- We denote by $\mathcal{M}_A$ the set of assertions contained in $\mathcal{M}$ having either $Inst_C$ or $Inst_R$ as predicate in their right-hand side.
- We denote by $\mathcal{M}_T$ the set $\mathcal{M} \setminus \mathcal{M}_A$, that is, the set of assertions contained in $\mathcal{M}$ having any of Isa_C, Isa_R, $Disj_C$, $Disj_R$ as predicate in their right-hand side.
- $\mathcal{M}$ is called an *instance-mapping* if $Inst_C$ and $Inst_R$ are the only predicates that appear in the right-hand side of the mapping assertions in $\mathcal{M}$.
- We say that e occurs as a concept argument in the atoms $Inst_C(e, e')$, $Isa_C(e, e')$, $Isa_C(e', e)$, $Disj_C(e, e')$, and $Disj_C(e', e)$.
- We say that e occurs as a role argument in the atoms $Inst_R(e', e'', e)$, $Isa_R(e, e')$, $Isa_R(e', e)$, $Disj_R(e, e')$, and $Disj_R(e', e)$.
- 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 an expression of the form $q(x_1, \ldots, x_n) \leftarrow a_1, \ldots, a_m$ such that $x_1, \ldots, x_n$ belong to $\mathcal{V}$, $a_1, \ldots, a_m$ is a conjunction of atoms, each atom a_j (with $1 \leq j \leq m$) is either a DL atom or an instance-query atom (i.e., an atom whose meta-predicate is $Inst_C$ or $Inst_R$), and each x_i (with $1 \leq j \leq n$) occurs in at least one a_j. An extended UCQ (EUCQ) is a union of ECQs.

- Given a TBox $\mathcal{T}$ (specified in high-order style syntax, cf. Section 2), we define $\mathit{Concepts}(\mathcal{T}) = \{e, \mathit{Exists}(e'), \mathit{Exists}(\mathit{Inv}(e')) \mid e \text{ occurs as a concept argument in } \mathcal{T} \text{ and } e' \text{ occurs as a role argument in } \mathcal{T}\}$, and $\mathit{Roles}(\mathcal{T}) = \{e, \mathit{Inv}(e) \mid e \text{ occurs as a role argument in } \mathcal{T}\}$.
- Given a mapping $\mathcal{M}$ and a database DB, $\mathit{Retrieve}(\mathcal{M}, DB)$ denotes the $\mathit{Hi}(\textit{DL-Lite}_{\mathcal{R}})$ KB $\mathcal{H}$ defined as follows:
$$\mathcal{H} = \{\psi(\vec{t}) \mid \Phi(\vec{x}) \rightsquigarrow \psi \in \mathcal{M} \text{ and } DB \models \Phi(\vec{t})\}$$
where $\vec{t}$ is a tuple of constants and $\vec{x}$ is a tuple of variables having the same arity.
- 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 $\mathcal{A} = \mathit{Retrieve}(\mathcal{M}, DB)$.

The query answering technique we are about to present is based on the following four main steps.

1. In the first step, all intensional assertions are gathered by accessing the sources and using the mapping, in particular the $\mathcal{M}_T$ portion. This way, a $\textit{DL-Lite}_{\mathcal{R}}$ TBox $\mathcal{T}$ is available for the subsequent steps.
2. In the second step, the input query is rewritten on the basis of $\mathcal{T}$, using the algorithm *PerfectRef* presented in [23]. In fact this is not just a trivial call to *PerfectRef*, since we need to transform the input IHUCQ, which cannot be given directly in input to *PerfectRef*, into a EUCQ. Similarly, we also need to translate the rewriting produced by *PerfectRef* in a form that is compatible with the syntax used in the mapping (which is required by the following steps of our algorithm).
3. In the third step, the query obtained by the second step is unfolded using the mapping, in particular the $\mathcal{M}_A$ portion, so as to obtain a query expressed over the alphabet of the source schema.
4. In the fourth step, the query obtained by the third step is evaluated over the source data, so as to obtain the final result.

In Section 5.1 we deal with the second step of our technique. In particular, we study the problem of computing a *perfect rewriting* of an IHUCQ over a $\textit{DL-Lite}_{\mathcal{R}}$ TBox. We recall that given a query q, a perfect rewriting of q with respect to a TBox $\mathcal{T}$ is a query q' such that, for every ABox $\mathcal{A}$, $\mathit{cert}(q, \mathcal{T} \cup \mathcal{A}) = \mathit{cert}(q', \mathcal{A})$ [23]. That is, to obtain the certain answers to q in $\mathcal{T} \cup \mathcal{A}$, it is sufficient to evaluate q' over $\mathcal{A}$ seen as a database (such an evaluation indeed returns the certain answers to q in $\mathcal{A}$). As said, in our rewriting algorithm we first retrieve a TBox through the mapping (step 1). We thus adapt the above notion to the context of MKBs as follows. Given an MKB $\mathcal{K} = \langle DB, \mathcal{M} \rangle$, let $\mathcal{T} = \mathit{Retrieve}(\mathcal{M}_T, DB)$, a perfect rewriting to q with respect to $\mathcal{T}$ and the instance-mapping $\mathcal{M}_A$ is a query q' such that, for every ABox $\mathcal{A}$ retrievable through $\mathcal{M}_A$, $\mathit{cert}(q, \mathcal{M}_T \cup \mathcal{A}) = \mathit{cert}(q', \mathcal{A})$.

After the detailed description of the second step above, in the subsequent Section 5.2, we present the complete query answering algorithm for MKBs based on our perfect rewriting technique.

5.1. Query Rewriting

The basic idea of our technique is to reduce the computation of the perfect rewriting of an IHUCQ over a $\textit{DL-Lite}_{\mathcal{R}}$ TBox to the computation of the perfect rewriting of an UCQ over a $\textit{DL-Lite}_{\mathcal{R}}$ TBox, which can be then done by using the algorithm *PerfectRef* described in [23].

To this aim, we first transform the IHUCQ into a standard UCQ, actually an EUCQ. This is realized through a first partial grounding of the query, using the function *PMG* and then through the functions *Normalize* and τ. In particular, the function *PMG* eliminates the meta-variables, i.e., the variables occurring as a concept or as a role argument, from the query, *Normalize* substitutes Inst_C atoms whose second argument is of the form $\mathit{Exists}(e)$ into Inst_R atoms (i.e., the instance of the domain or the range of a role expression e is reformulated as an instance of e), and τ transforms Inst_C and Inst_R atoms in DL atoms.

Afterwards, the query resulting from the perfect rewriting of the EUCQ is transformed back into an IHUCQ, by the functions *Denormalize* and τ^-. This is because the third step of the query answering algorithm assumes the query to be an IHUCQ.

We now describe in more detail the functions *PMG*, *Normalize*, *Denormalize*, τ and τ^-.

If q, q' are two IHCQs, and $\mathcal{T}$ is a TBox, then q' is a *partial metagrounding of q with respect to $\mathcal{T}$* if $q' = \sigma(q)$, where σ 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:

- if x occurs in a concept position in q, then $\sigma(x) \in \mathit{Concepts}(\mathcal{T})$;
- if x occurs in a role position in q, then $\sigma(x) \in \mathit{Roles}(\mathcal{T})$.

Given an IHCQ q and a TBox $\mathcal{T}$, the function *PMG* applied to q and $\mathcal{T}$ computes the set of all partial metagroundings of q with respect to $\mathcal{T}$, i.e., it computes the IHUCQ $Q = \{q' \mid q' \text{ is a partial metagrounding of } q \text{ w.r.t. } \mathcal{T}\}$. When applied to an IHUCQ Q and a TBox $\mathcal{T}$, the function *PMG* computes the IHUCQ $\bigcup_{q \in Q} \mathit{PMG}(q, \mathcal{T})$.

Example 6. *Consider the MKB $\mathcal{K} = \langle \mathcal{D}, \mathcal{M} \rangle$ given in Example 4, the TBox $\mathcal{T}$ dynamically constructed from $\mathcal{D}$ through $\mathcal{M}$, and represented in Figure 6, and the query (i) given in Example 5. Then, $\mathit{Concepts}(\mathcal{T}) = \{\mathit{Car}, \mathit{Coupe}, \mathit{Ford}, \ldots, \texttt{1973 FALCON XB GT COUPE}, \ldots, \mathit{Exists}(\mathit{produced\_in}), \ldots\}$, $\mathit{Roles}(\mathcal{T}) = \{\mathit{produced\_in}, \mathit{Inv}(\mathit{produced\_in})\}$, and $\mathit{PMG}(q, \mathcal{T})$, contains, among other queries, the query:*

$$q(x) \leftarrow \mathit{Inst}_C(x, \mathit{Ford}), \mathit{Inst}_C(x, \texttt{FALCON}), \mathit{Inst}_C(\texttt{FALCON}, \mathit{Coupe}), \mathit{Inst}_R(x, \text{AUSTRALIA}, \text{produced\_in}) \quad (1)$$

where we abbreviated `1973 FALCON XB GT COUPE` *into* `FALCON` *(to obtain all other queries in $\mathit{PMG}(q, \mathcal{T})$ substitute* `FALCON` *with all elements of $\mathit{Concepts}(\mathcal{T})$).*

Notice that the notion of partial metagrounding *PMG* is crucial for our rewriting method. Indeed, even if in $\mathit{Hi}(\mathit{DL\text{-}Lite}_{\mathcal{R}})$ the set of expressions that can be constructed from a finite set of names occurring in the TBox is infinite, we can in fact limit to ground the meta-variables in the query on a finite set of expressions only, as stated by the following lemma.

Lemma 1. *If Q is an IHUCQ, and $\mathcal{T}$ is a TBox, then for every ABox $\mathcal{A}$, $\mathit{cert}(Q, \mathcal{T} \cup \mathcal{A}) = \mathit{cert}(\mathit{PMG}(Q, \mathcal{T}), \mathcal{T} \cup \mathcal{A})$.*

Proof. Assume $\mathit{cert}(Q, \mathcal{T} \cup \mathcal{A}) \neq \mathit{cert}(\mathit{PMG}(Q, \mathcal{T}), \mathcal{T} \cup \mathcal{A})$. It is easy to prove that $\mathit{cert}(Q, \mathcal{T} \cup \mathcal{A}) \supset \mathit{cert}(\mathit{PMG}(Q, \mathcal{T}), \mathcal{T} \cup \mathcal{A})$. Thus, there exists a tuple t such that $t \in \mathit{cert}(Q, \mathcal{T} \cup \mathcal{A}) - \mathit{cert}(\mathit{PMG}(Q, \mathcal{T}), \mathcal{T} \cup \mathcal{A})$. This implies that there exists a model $\mathcal{I}$ for $\mathcal{T} \cup \mathcal{A}$ such that $\mathcal{I} \models Q(t)$ and $\mathcal{I} \not\models \mathit{PMG}(Q, \mathcal{T})(t)$. Let Δ be the domain of $\mathcal{I}$. We define below an interpretation $\mathcal{I}^{\downarrow}$ over the same domain Δ. 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 $e \in \mathit{Concepts}(\mathcal{T})$ 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 $e \in \mathit{Roles}(\mathcal{T})$ 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 the KB $\mathcal{T} \cup \mathcal{A}$. However, it is also straightforward to verify that $\mathcal{I}^{\downarrow} \models Q(t)$ if and only if $\mathcal{I}^{\downarrow} \models \mathit{PMG}(Q, T)(t)$, and since by hypothesis $\mathcal{I} \not\models \mathit{PMG}(Q, T)(t)$, then $\mathcal{I}^{\downarrow} \not\models \mathit{PMG}(Q, T)(t)$ as well. As a consequence, $\mathcal{I}^{\downarrow} \not\models Q(t)$, which contradicts the hypothesis that $t \in \mathit{cert}(Q, \mathcal{T} \cup \mathcal{A})$, thus proving the thesis. □

We now turn our attention to the functions *Normalize* and *Denormalize*.

Let α be an instance atom, $\mathit{Normalize}(\alpha)$ returns a new atom defined 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 $Normalize(\alpha) = Inst_R(e_1, \_, e')$, where _ denotes an existentially quantified variables;
- if $\alpha = Inst_C(e_1, e_2)$ and e_2 has the form $Exists(Inv(e'))$ where e' is any expression, then $Normalize(\alpha) = Inst_R(\_, e_1, e')$.

The above definition naturally extends to queries. More precisely, let $q \leftarrow \alpha_1, \ldots, \alpha_n$ be an IHCQ , $Normalize(q)$ computes the IHCQ $q \leftarrow Normalize(\alpha_1), \ldots, Normalize(\alpha_n)$. Finally, if Q is an IHUCQ, then $Normalize(Q)$ returns the query $\bigcup_{q \in Q} Normalize(q)$. As an example, consider the query

$$q(x) \leftarrow Inst_C(x, Ford), Inst_C(x, Exists(produced\_in)), Inst_R(x, \text{AUSTRALIA}, produced\_in).$$

$Normalize(q)$ returns the query

$$q(x) \leftarrow Inst_C(x, Ford), Inst_R(x, \_, produced\_in), Inst_R(x, \text{AUSTRALIA}, produced\_in).$$

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

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

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

We formally introduce below the functions τ and τ^-, which transform IHUCQs into EUCQs and vice versa. Let q be an IHCQ and let $\mathcal{T}$ be a TBox, $\tau(q, \mathcal{T})$ is the ECQ obtained from q as follows:

- each atom of q of the form $Inst_C(e_1, e_2)$, such that $e_2 \in Concepts(\mathcal{T})$, is replaced with the atom $e_2(e_1)$;
- each atom of q of the form $Inst_R(e_1, e_2, e_3)$, such that $e_3 \in Roles(\mathcal{T})$, is replaced with the atom $e_3(e_1, e_2)$.

Example 7. *Let us now apply the function τ to the query (1) (which is already normalized), and obtain the query*

$$q(x) \leftarrow Ford(x), \texttt{FALCON}(x), Coupe(\texttt{FALCON}), produced\_in(x, \text{AUSTRALIA}). \quad (2)$$

Then, given an IHUCQ Q, we define $\tau(Q, \mathcal{T}) = \{\tau(q, \mathcal{T}) \mid q \in Q\}$.

Let q be an ECQ and $\mathcal{T}$ be a TBox, then $\tau^-(q, \mathcal{T})$ is the IHCQ obtained from q as follows:

- each atom of q of the form $e_2(e_1)$ is replaced with the atom $Inst_C(e_1, e_2)$;
- each atom of q of the form $e_3(e_1, e_2)$ is replaced with the atom $Inst_R(e_1, e_2, e_3)$.

Then, given an IHUCQ Q, we define $\tau^-(Q, \mathcal{T}) = \{\tau^-(q, \mathcal{T}) \mid q \in Q\}$. An example of application of $\tau^-(Q, \mathcal{T})$ can be obtained by simply reversing the transformation shown in Example 7.

We can now formally define our algorithm for query rewriting. The algorithm takes as input an IHUCQ, a TBox and an instance-mapping, and returns a new IHUCQ. 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$_{\mathcal{R}}$ presented in [23] (Actually, we consider a slight generalization of that algorithm, allowing for the presence of a ternary relation ($Inst_R$) in the query.).

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

Example 8. *To provide an example of a complete execution of the algorithm RewriteIHUCQ*$(Q, \mathcal{T}, \mathcal{M})$*, let us now continue the rewriting described in Example 6 and in Example 7. Let us focus on the PerfectRef function, and apply it to the query (2) (which is contained in the set Q_2 of CQs). PerfectRef transforms the atoms of the query using TBox inclusions as rewriting rules, from right to left. For example, according to the inclusion* $\texttt{FALCON} \sqsubseteq Ford$*, it rewrites the atom* $Ford(x)$ *in query (2) into the atom* $\texttt{FALCON}(x)$ *(intuitively, the PerfectRef encodes in the rewriting the knowledge expressed by the ontology saying that to obtain instances of Ford one has to look also for instances of* `FALCON`*). In this way PerfectRef produces the following query and adds it to the set Q_3 (notice that the atom* $\texttt{FALCON}(x)$ *was already present in the query, thus the effect of the rewriting in this case is simply dropping the first atom of the query):*

$$q(x) \leftarrow \texttt{FALCON}(x), Coupe(\texttt{FALCON}), produced\_in(x, \text{AUSTRALIA}). \tag{3}$$

A similar reformulation is performed for all atoms whose predicate occurs in the right-hand side of a positive inclusion (provided that its arguments are not bound, as described in [23]). Among the queries returned by PerfectRef we consider in the following only query (3), which, as we will see later, will allow us to obtain the answer to the original query (the other queries returned by RewriteIHUCQ$(Q, \mathcal{T}, \mathcal{M})$ *do not really contribute to the final answer in our example). Finally, RewriteIHUCQ*$(Q, \mathcal{T}, \mathcal{M})$ *applies τ^- and Denormalize (which in this case is immaterial) to query (3), thus returning the query*

$$q(x) \leftarrow Inst_C(x, \texttt{FALCON}), Inst_C(\texttt{FALCON}, Coupe), Inst_R(x, \text{AUSTRALIA}, \text{produced\_in}). \tag{4}$$

The IHUCQ returned by *RewriteIHUCQ*$(Q, \mathcal{T}, \mathcal{M})$ constitutes a perfect rewriting 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 an 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. The proof follows from Definition 4, from Lemma 1, from the correctness of the algorithm *PerfectRef* [23], and from the fact that the functions *Normalize*, τ, τ^- and *Denormalize* just perform equivalent transformations of the query. □

5.2. Query Answering

We now provide an algorithm for query answering over MKBs, which makes use of the query rewriting technique presented in the previous subsection. As already said, 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 rewriting of the input IHUCQ.

To complete query answering, we have to also 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 occurring in the corresponding mapping assertions (step 3 of our technique). In this way we obtain an FOL query expressed over the database. This second rewriting step, usually called *unfolding*, can be performed by the algorithm UnfoldDB presented in [20] (Here, we assume that the algorithm UnfoldDB takes as input an EUCQ and an instance-mapping. This corresponds to actually considering a straightforward extension of the algorithm presented in [20] in order to deal with the presence of the ternary predicate $Inst_R$.). 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 $\mathcal{M}_T$. Furthermore, 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, $Hi(DL\text{-}Lite_{\mathcal{R}})$ MKB $\mathcal{K} = \langle DB, \mathcal{M} \rangle$
OUTPUT: $cert(Q, K)$
begin
  $\mathcal{T} = Retrieve(\mathcal{M}_T, DB_{\mathcal{M}_T})$;
  $Q' = RewriteIHUCQ(Q, \mathcal{T}, \mathcal{M}_A)$;
  $Q'' = \mathsf{UnfoldDB}(Q', \mathcal{M}_A)$;
  return $IntEval(Q'', DB_{\mathcal{M}_A})$
end

The algorithm starts by retrieving (function $Retrieve(\mathcal{M}_T, DB_{\mathcal{M}_T})$) the TBox $\mathcal{T}$ from $DB_{\mathcal{M}_T}$ through the mapping $\mathcal{M}_T$. Then, it computes (function $RewriteIHUCQ(Q, \mathcal{T}, \mathcal{M}_A)$) the perfect rewriting of the query with respect to the retrieved TBox and next computes (function $\mathsf{UnfoldDB}(Q', \mathcal{M}_A)$) the unfolding of such a query with respect to the mapping $\mathcal{M}_A$. Finally, it evaluates (function $IntEval(Q'', DB_{\mathcal{M}_A})$ the query over the database and returns the result of the evaluation.

Example 9. *Let us consider again the MKD system and the query of Example 6. We notice that $\mathcal{M}_T$ and $\mathcal{M}_A$ consists of the mapping assertions M1-M9 and M10-M-13 given in Example 4, respectively, and $DB_{\mathcal{M}_T} = DB_{\mathcal{M}_A}$ coincides with the database $\mathcal{D}$ described in Figure 3. Thus, the TBox $\mathcal{T}$ returned by $Retrieve(\mathcal{M}_T, DB_{\mathcal{M}_T})$ coincides with the TBox described in Figure 6 (not considering InstanceOf arrows). Q' is as discussed in Example 8. As said, amog all queries in Q' we consider only query (4). By unfolding it through $\mathsf{UnfoldDB}(Q', \mathcal{M}_A)$, we obtain (among other queries) the FOL query*

$$\{x \mid \texttt{T-Cars}(x, c1, a, b) \wedge \texttt{T-CarTypes}(c1, FALCON, d, e) \wedge \texttt{T-CarTypes}(a', FALCON, b', coupe) \wedge \texttt{T-Cars}(x, a'', b'', \text{AUSTRALIA})\}$$

More in detail, we have unfolded the atom $Inst_C(x, FALCON)$ using mapping M13, atom $Inst_C(FALCON, coupe)$ using mapping M12, and the atom $Inst_R(x, \text{AUSTRALIA}, produced\_in)$ using mapping M10. Notice also that other queries are produced by the function UnfoldDB *applied to the query (4) and the $\mathcal{M}_A$ (e.g., by unfolding $Inst_C(FALCON, coupe)$ through M12). However, the one we showed is sufficient to obtain the answers to the original query. It is indeed easy to see that by evaluating the above query over $DB_{\mathcal{M}_A}$ (see Figure 3), we obtain the answer* `INTERCEPTOR`*, and this is the only answer to the query (i) given in Example 5 evaluated over the MKD system $\mathcal{K}$.*

To prove correctness of the above algorithm, we first state the following property, whose proof immediately follows from the definition of $Retrieve(\mathcal{M}, DB)$ and the definition of model of a $Hi(DL\text{-}Lite_{\mathcal{R}})$ MKB.

Lemma 2. *Let $\mathcal{K} = \langle DB, \mathcal{M} \rangle$ be a $Hi(DL\text{-}Lite_{\mathcal{R}})$ MKB and let $\mathcal{H} = Retrieve(\mathcal{M}, DB)$. The set of models of $\mathcal{K}$ and $\mathcal{H}$ coincide.*

We also need the following additional property.

Lemma 3. *Let $\mathcal{M}$ be an instance-mapping, and let Q be an IHUCQ. Then, for every database DB,*

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

Proof. The proof follows from Definition 4 and by a natural, slight extension (which we leave to the reader) of the proof of correctness of the algorithm UnfoldDB shown in [20]. □

We are now ready to show the 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 an IHUCQ, and let U be the set of tuples returned by $Answer(Q, \mathcal{K})$. Then, $cert(Q, \mathcal{K}) = U$.*

Proof. The proof immediately follows from Theorem 1 and Lemmas 2 and 3. □

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, let Q be an 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 PSPACE with respect to the size of $\mathcal{K}$, and is NP-complete with respect to the size of $Q(\vec{t})$.*

Proof. To decide whether $\mathcal{K} \models Q(\vec{t})$ we can execute $Answer(Q(\vec{t}), \mathcal{K})$. Then, membership in AC^0 with respect to the size of $DB_{\mathcal{M}_A}$ follows from the fact that the only step of the algorithm *Answer* that depends on $DB_{\mathcal{M}_A}$ is $IntEval(Q'', DB_{\mathcal{M}_A})$, where Q'' is a FOL query and the fact that evaluating an FOL query over a database is in AC^0 in data complexity. Membership in PSPACE with respect to the size of $\mathcal{K}$ follows from the fact that evaluating a FOL query is in PSPACE in combined complexity (whereas the other steps of the algorithm *Answer* are in PTIME with respect to the size of the $\mathcal{K}$). Finally, NP-completeness with respect to the size of $Q(\vec{t})$ follows from Lemma 2, from the fact that computing $Retrieve(\mathcal{M}, DB)$ can obviously be done in constant time with respect to the size of $Q(\vec{t})$ and from the fact that evaluating an IHUCQ over a $Hi(DL\text{-}Lite_{\mathcal{R}})$ KB is NP-complete in query complexity. □

6. Conclusions

In this paper we have investigated the issue of generating both the TBox and the ABox of a DL ontology on the fly from data stored in data sources through asserted mappings. The two main ingredients for obtaining such a degree of flexibility are (i) enriching the mapping language so as to extract from the data sources the knowledge about possible concepts and roles that are relevant to the domain of interest and (ii) relying on higher-order description logics which blur the distinction between concepts/roles at the intensional level and individuals at the extensional level.

The approach presented here can be useful in a number of scenarios. Although in this paper we have discussed a single example, we point out that such example can actually be seen as a prototypical instance of so-called *product databases*, where one needs to model information about types and models of products (akin to the table T-CarTypes in Figure 3), as well as specific data about single products

(like table T-Car in Figure 3). In the applications with product databases, for example in the context of e-commerce, when new products become available, they are typically acquired by accessing catalogues, where the above mentioned information about types and models of products is reported. In our framework, such catalogues are simply modelled as data sources with suitable mappings to the ontology (similarly to the mappings from the table T-CarTypes in Example 4), and such mappings are used to dynamically extend the ontology with new classes and relationships between them present in the catalogue.

We believe that the approach described in this paper can be a starting point for several investigations going beyond what we have presented here. 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 [27]. In addition, the user can in principle even compose on the fly the TBox to use when answering a query. Obviously notions such as authorization views and consistency acquire an intriguing flavor in this setting. As for the former notion, the mechanisms described in this paper can be used to hide intensional knowledge for privacy purposes, thus extending the approaches aiming at filtering only extensional knowledge. As for consistency, an interesting idea is to allow for contradicting TBox axioms coming to the data sources to coexist as long as they are not used together when performing query answering. Other promising directions to pursue include handling possible inconsistencies in MKBs, in the spirit of [28], considering more sophisticated form of mappings, such as the ones in the context of peer-to-peer data integration [29], conceiving the assertions coming from the mapping as updates on the current knowledge base (in the spirit of [30]), and exploiting the ideas presented here as a basis for the problem of acquiring knowledge graphs (see [31]) from existing data sources.

Author Contributions: Writing—original draft, F.D.P., G.D.G., D.L., M.L. and R.R.

Funding: This work was partly supported by EPSRC (grants M025268 and N023056), by MIUR (PRIN 2017 project "HOPE"), by the European Research Council under the European Unions Horizon 2020 Programme through the ERC Advanced Grant WhiteMec (No. 834228), and by Sapienza Università di Roma (2019 project CQEinOBDM).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Lenzerini, M. Managing Data through the Lens of an Ontology. *AI Mag.* **2018**, *39*, 65–74. [CrossRef]
2. Xiao, G.; Calvanese, D.; Kontchakov, R.; Lembo, D.; Poggi, A.; Rosati, R.; Zakharyaschev, M. Ontology-Based Data Access: A Survey. In Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI), Stockholm, Sweden, 13–19 July 2018; pp. 5511–5519.
3. Baader, F.; Calvanese, D.; McGuinness, D.; Nardi, D.; Patel-Schneider, P.F. (Eds.) *The Description Logic Handbook: Theory, Implementation and Applications*, 2nd ed.; Cambridge University Press: Cambridge, UK, 2007.
4. Savo, D.F.; Lembo, D.; Lenzerini, M.; Poggi, A.; Rodríguez-Muro, M.; Romagnoli, V.; Ruzzi, M.; Stella, G. MASTRO at Work: Experiences on Ontology-Based Data Access. In Proceedings of the 23rd International Workshop on Description Logic (DL), Waterloo, ON, Canada, 4–7 May 2010; Volume 573, pp. 20–31.
5. Antonioli, N.; Castanò, F.; Coletta, S.; Grossi, S.; Lembo, D.; Lenzerini, M.; Poggi, A.; Virardi, E.; Castracane, P. Ontology-based Data Management for the Italian Public Debt. In Proceedings of the 8th International Conference on Formal Ontology in Information Systems (FOIS), Rio de Janeiro, Brazil, 22–26 September 2014; pp. 372–385.
6. Giese, M.; Soylu, A.; Vega-Gorgojo, G.; Waaler, A.; Haase, P.; Jiménez-Ruiz, E.; Lanti, D.; Rezk, M.; Xiao, G.; Özçep, Ö.L.; et al. Optique: Zooming in on Big Data. *IEEE Comput.* **2015**, *48*, 60–67. [CrossRef]
7. López, V.; Stephenson, M.; Kotoulas, S.; Tommasi, P. Data Access Linking and Integration with DALI: Building a Safety Net for an Ocean of City Data. In Proceedings of the 14th International Semantic Web Conference (ISWC), Bethlehem, PA, USA, 11–15 October 2015; Lecture Notes in Computer Science; Springer: Basel, Switzerland, 2015, Volume 9367, pp. 186–202.

8. Calvanese, D.; De Giacomo, G.; Lembo, D.; Lenzerini, M.; Poggi, A.; Rodriguez-Muro, M.; Rosati, R.; Ruzzi, M.; Savo, D.F. The Mastro System for Ontology-based Data Access. *Semant. Web J.* **2011**, *2*, 43–53. [CrossRef]
9. De Giacomo, G.; Lembo, D.; Lenzerini, M.; Poggi, A.; Rosati, R.; Ruzzi, M.; Savo, D.F. MASTRO: A Reasoner for Effective Ontology-Based Data Access. In Proceedings of the 1st International Workshop on OWL Reasoner Evaluation (ORE), Manchester, UK, 1 July 2012; Volume 858.
10. Rodriguez-Muro, M.; Kontchakov, R.; Zakharyaschev, M. Ontology-Based Data Access: Ontop of Databases. In Proceedings of the 12th International Semantic Web Conference (ISWC), Sydney, Australia, 21–25 October 2013; Lecture Notes in Computer Science; Springer: Basel, Switzerland, 2013; Volume 8218, pp. 558–573.
11. Ullman, J.D. Information Integration using Logical Views. *Theor. Comput. Sci.* **2000**, *239*, 189–210. [CrossRef]
12. Halevy, A.Y. Answering Queries Using Views: A Survey. *Very Large Database J.* **2001**, *10*, 270–294. [CrossRef]
13. Lenzerini, M. Data Integration: A Theoretical Perspective. In Proceedings of the 21st ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS), Madison, WI, USA, 3–5 June 2002; pp. 233–246.
14. Kolaitis, P.G. Schema Mappings, Data Exchange, and Metadata Management. In Proceedings of the 24th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS), Baltimore, MD, USA, 13–15 June 2005; pp. 61–75.
15. Di Pinto, F.; De Giacomo, G.; Lenzerini, M.; Rosati, R. Ontology-Based Data Access with Dynamic TBoxes in *DL-Lite*. In Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI), Toronto, ON, Canada, 22–26 July 2012; pp. 719–725. Available online: https://www.aaai.org/ocs/index.php/AAAI/AAAI12/paper/view/5021 (accessed on 10 December 2019).
16. Chen, W.; Kifer, M.; Warren, D.S. HILOG: A Foundation for Higher-Order Logic Programming. *J. Log. Program.* **1993**, *15*, 187–230. [CrossRef]
17. Pan, J.Z.; Horrocks, I. OWL FA: A Metamodeling Extension of OWL DL. In Proceedings of the 15th International World Wide Web Conference (WWW), Edinburgh, UK, 23–26 May 2006; pp. 1065–1066.
18. De Giacomo, G.; Lenzerini, M.; Rosati, R. Higher-Order Description Logics for Domain Metamodeling. In Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI), San Francisco, CA, USA, 7–11 August 2011.
19. de Carvalho, V.A.; Almeida, J.P.A.; Fonseca, C.M.; Guizzardi, G. Multi-level ontology-based conceptual modeling. *Data Knowl. Eng.* **2017**, *109*, 3–24. [CrossRef]
20. Poggi, A.; Lembo, D.; Calvanese, D.; De Giacomo, G.; Lenzerini, M.; Rosati, R. Linking Data to Ontologies. *J. Data Semant.* **2008**, *X*, 133–173. [CrossRef]
21. Papotti, P.; Torlone, R. Schema Exchange: Generic Mappings for Transforming Data and Metadata. *Data Knowl. Eng.* **2009**, *68*, 665–682. [CrossRef]
22. Calvanese, D.; De Giacomo, G.; Lenzerini, M.; Rosati, R.; Vetere, G. *DL-Lite*: Practical Reasoning for Rich DLs. In Proceedings of the 17th International Workshop on Description Logic (DL), Vienna, Austria, 17–20 July 2004; Volume 104.
23. Calvanese, D.; De Giacomo, G.; Lembo, D.; Lenzerini, M.; Rosati, R. Tractable Reasoning and Efficient Query Answering in Description Logics: The *DL-Lite* Family. *J. Autom. Reason.* **2007**, *39*, 385–429. [CrossRef]
24. Motik, B.; Fokoue, A.; Horrocks, I.; Wu, Z.; Lutz, C.; Cuenca Grau, B. OWL Web Ontology Language Profiles. W3C Recommendation, World Wide Web Consortium, 2009. Available online: http://www.w3.org/TR/owl-profiles/ (accessed on 10 December 2019).
25. Lembo, D.; Pantaleone, D.; Santarelli, V.; Savo, D.F. Easy OWL Drawing with the Graphol Visual Ontology Language. In Proceedings of the 15th International Conference on the Principles of Knowledge Representation and Reasoning (KR), Cape Town, South Africa, 25–29 April 2016; pp. 573–576.
26. Lembo, D.; Pantaleone, D.; Santarelli, V.; Savo, D.F. Drawing OWL 2 ontologies with Eddy the editor. *AI Commun.—Eur. J. Artif. Intell.* **2018**, *31*, 97–113. [CrossRef]
27. Parsons, J.; Wand, Y. Emancipating Instances from the Tyranny of Classes in Information Modeling. *ACM Trans. Database Syst.* **2000**, *25*, 228–268. [CrossRef]
28. Lembo, D.; Lenzerini, M.; Rosati, R.; Ruzzi, M.; Savo, D.F. Inconsistency-tolerant query answering in ontology-based data access. *J. Web Semant.* **2015**, *33*, 3–29. [CrossRef]
29. Calvanese, D.; Damaggio, E.; De Giacomo, G.; Lenzerini, M.; Rosati, R. Semantic Data Integration in P2P Systems. In Proceedings of the International Workshop on Databases, Information Systems and Peer-to-Peer Computing (DBISP2P 2003), Berlin, Germany, 7–8 September 2003.

30. De Giacomo, G.; Lenzerini, M.; Poggi, A.; Rosati, R. On the Update of Description Logic Ontologies at the Instance Level. In Proceedings of the 21st National Conference on Artificial Intelligence (AAAI), Boston, MA, USA, 16–20 July 2006; pp. 1271–1276.
31. Ehrlinger, L.; Wöß, W. Towards a Definition of Knowledge Graphs. In Proceedings of the Posters and Demos Track of the 12th International Conference on Semantic Systems (SEMANTiCS), Leipzig, Germany, 12–15 September 2016.