DL-Lite Full: A Sub-language of OWL 2 Full for Powerful Meta-modeling

Zhenzhen Gu[1] and Songmao Zhang[2]

[1] Faculty of Computer Science, Free University of Bozen-Bolzano, Bolzano, Italy
[2] Academy of Mathematics and Systems Sciences, Chinese Academy of Sciences, Beijing, China
zhgu@unibz.it, smzhang@math.ac.cn

Abstract. We address the problem of encoding meta-modeling in real-world knowledge bases (KBs), i.e., multiple uses of names and especially non-standard uses of rdf:type, in DL-Lite$_{\mathcal{R}}$, and propose a sub-language of OWL 2 Full, called DL-Lite Full, for the web-scale Open Data for powerful meta-modeling. For meta-knowledge access, meta-queries are introduced by allowing variables to occur in the class and role positions of conjunctive queries. For scalability, based on the techniques of DL-Lite$_{\mathcal{R}}$, we provide a way of reducing both satisfiability checking and conjunctive query answering in DL-Lite Full to evaluating queries over the data layers of the KBs, and further an approach of answering meta-queries via meta-query rewriting and partial variable materialization. Based on these, we obtain that the considered reasoning tasks in DL-Lite Full still have PTime KB complexity and AC_0 data complexity.

Keywords: Semantic Web · DL-Lite$_{\mathcal{R}}$ · OWL 2 Full · Meta-modeling.

1 Introduction

Description logics (DLs) [5], decidable sub-languages of the first-order logic, lay the logic foundation of the famous and popular web ontology language OWL. The latest version OWL 2 [12, 24] includes two expressive sub-languages OWL 2 DL and OWL 2 Full as well as three profiles OWL 2 QL, OWL 2 EL and OWL 2 RL for scalability at different aspects. OWL 2 DL is formalized based on the expressive DL $\mathcal{SROIQ}$ [38] and OWL 2 QL, OWL 2 EL and OWL 2 RL are underlined by the lightweight DLs DL-Lite$_{\mathcal{R}}$ [6, 7], $\mathcal{EL}$ [9] and $\mathcal{DLP}$ [10] respectively. OWL 2 Full is obtained by removing the restrictions for decidability. The distinctive feature of OWL 2 Full is meta-modeling where ordinary names can be used in multiple ways, such as asserting Mother to be a class and at the same time to be an individual of family roles. Besides, RDF(S)/OWL vocabulary terms which correspond to logic constructors can also be used as ordinary names to describe data and knowledge, such as asserting sem:type and sem:subTypeOf to be a sub-property of rdfs:type and rdfs:subClassOf, respectively.

Meta-modeling plays an import role in describing complex patterns and has been heavily used in some actual open knowledge bases (KBs) such as the commonsense KBs including SUMO [17] for the top-level concepts and OpenCyc [18] for across-domain knowledge, and the life sciences KBs [19] like FMA [20]. This can be seen from the statistics in our previous work [31]. Besides, some large scale Open Data [1, 2, 4] also contain massive meta-modeling, like the Billion Triple Challenge (BTC) data set [21], where among the names used as classes, 25.5% are also specified as individuals, and among the names used as roles, 8.7% are individuals at the same time. And some terms like $\mathsf{moive:film\_cut}$ are used as classes, roles and individuals simultaneously.

High expressivity of OWL 2 Full leads to the undecidability of reasoning [26], making complete reasoning impossible and developing practical systems difficult. In order to cater for the requirement of meta-modeling, OWL 2 DL provides a technique called punning which syntactically allows names to have multiple uses while semantically treats the multiple uses of names as different semantic entities. As a syntactical solution, punning will not draw any extra conclusions compared with OWL 2 DL. Besides punning, there already exist some works [26–32, 34–36, 39–43] studying extending DLs like $\mathcal{SROIQ}$ and DL-Lite$_{\mathcal{R}}$ with meta-modeling by allowing names to have multiple uses. However, in all these work, except [39] which investigates extending meta-modeling with instantiation in $\mathcal{SROIQ}$ and $\mathcal{SHOIQ}$, non-standard uses of RDF(S)/OWL vocabulary terms have been ignored. Although, compared with multiple uses of ordinary names, non-standard uses of RDF(S)/OWL vocabulary terms relatively less happen, as declared in [45], it nonetheless can be useful, such as distinguishing different kinds of instantiations.

In order to capture the massive meta-modeling described in Open Data, in this paper, we discuss encoding both multiple uses of ordinary names and non-standard use of $\mathsf{rdf:type}$ in DL-Lite$_{\mathcal{R}}$ (the language underling OWL 2 QL and especially designed for ontology based data access [6, 11]), and provide a sub-language of OWL 2 Full called DL-Lite Full. Non-standard uses of other RDF(S)/OWL vocabulary terms will be discussed in the future work. For meta-knowledge accessing, meta-queries are introduced by allowing variables to occur in the class and role positions of queries. We define the syntax and semantics of DL-Lite Full and meta-queries. For data layer scalability, we provide the way of reducing satisfiability checking and conjunctive query answering in DL-Lite Full to evaluate queries over the data layer of KBs and further the way of answering meta-queries through meta-query rewriting and partial variable materialization. Based on this, we obtain that the considered reasoning tasks in DL-Lite Full still have AC$_0$ data complexity and PTime KB complexity.

2 The definition of DL-Lite Full and meta-queries

2.1 The syntax of DL-Lite Full and meta-queries

Different from DL-Lite$_{\mathcal{R}}$, DL-Lite Full has only one name set N for classes, roles and individuals. This means that each name in N can be used as classes, roles and individuals simultaneously. In order to capture the non-standard uses of $\mathsf{rdf:type}$, N contains a special name type. DL-Lite Full is defined as follows.

Definition 1 *In DL-Lite Full, basic roles S, general roles R, basic classes B, and general classes C are defined as follows:*

$$S ::= P \mid P^{-}, \quad B ::= A \mid \exists S,$$
$$R ::= S \mid \neg S, \quad C ::= B \mid \neg B$$

where $A, P \in \mathsf{N}$. A DL-Lite Full axiom takes the form of $S \sqsubseteq_r R$ or $B \sqsubseteq_c C$, and a DL-Lite Full TBox $\mathcal{T}$ is a finite set of DL-Lite Full axioms where type *does not occur in the left-hand sides of inclusion axioms ($\sqsubseteq$). A DL-Lite Full individual assertion has the form $P(a,b)$ or $A(a)$, where $P, a, b, A \in \mathsf{N}$ and $P \neq \mathsf{type}$, and a DL-Lite Full ABox $\mathcal{A}$ is a finite set of individual assertions. A DL-Lite Full KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ is a tuple of DL-Lite Full TBox $\mathcal{T}$ and ABox $\mathcal{A}$.*

DL-Lite Full does not separate the names for classes, roles and individuals. Thus, we use $\sqsubseteq_r$ and $\sqsubseteq_c$ to distinguish between role inclusion axioms and class inclusion axioms. Besides, type can be used as ordinary names to describe schema knowledge. By this way, the specifications of the RDF vocabulary term $\mathsf{rdf : type}$ described in the real world KBs, such as $\mathsf{sem:type} \sqsubseteq_r \mathsf{type}$ and $\mathsf{sem:hasActor} \sqsubseteq_r \mathsf{sem : type}$, can be captured by DL-Lite Full. Here, type is forbidden to occur in the left-hands of inclusion axioms, since reference [22] has suggested that such kinds of axioms are generally best left to philosophers. As a matter of fact, we did not find such use of type in the actual KBs. Moreover, we emphasize that individual assertions ***with the form*** $\mathsf{type}(a, b)$ can be captured by $b(a)$.

Let V be a set of variables such that $\mathsf{V} \cap \mathsf{N} = \emptyset$. We define meta-queries.

Definition 2 *A query atom has the form $x(y,z)$ or $y(z)$, where $x, y, z \in \mathsf{N} \cup \mathsf{V}$ and $x \neq \mathsf{type}$. A meta-query (MQ) Q is an expression of the form $\alpha_1 \wedge \cdots \wedge \alpha_n \rightarrow q(\boldsymbol{x})$ where α_i are query atoms, $\boldsymbol{x}$ is a tuple of elements in $\mathsf{V} \cup \mathsf{N}$ and each variable in $\boldsymbol{x}$ occurs in some α_i. We use $\mathsf{body}(Q)$ to denote the body $\bigcup_{i=1}^{n} \{\alpha_i\}$ of Q and $\mathsf{head}(Q)$ the head $\boldsymbol{x}$ of Q. Q is called a **Boolean** query if $\mathsf{head}(Q) = ()$.*

Notice that query atoms in the form of $\mathsf{type}(x, y)$ can be captured by $y(x)$. We call the variables occurring in the $\circ$ positions of the atoms $\circ(x, y)$ and $\circ(x)$ as ***role variables*** and ***class variables***, respectively. Without class and role variables, meta-queries degrade into conjunctive queries (CQs). By allowing class and role variables in MQs, schema knowledge and data can be queried uniformly.

Example 1 *"Asking for the relationships that Lucy has" can be formally represented as the MQ: $?p(Lucy, ?x) \wedge ?c(?x) \rightarrow q(?p, ?x, ?c)$.*

2.2 The semantics of DL-Lite Full and meta-queries

DL-Lite Full and MQs are captured by the ν-semantics defined in [26] which is based on HiLog and takes a similar way of OWL 2 RDF-Based semantics to interpret the multiple uses of names. Here, the ν-semantics is extended based on OWL 2 RDF-Based semantics to capture the non-standard use of type.

ν-semantics. A ν-interpretation $\mathcal{V} = (\Delta^{\mathcal{V}}, \cdot^{\mathcal{V}}, \mathcal{R}^{\mathcal{V}}, \mathcal{C}^{\mathcal{V}})$ is a quadruple where $\Delta^{\mathcal{V}}$ is a non-empty domain set, and $\cdot^{\mathcal{V}}$, $\mathcal{R}^{\mathcal{V}}$ and $\mathcal{C}^{\mathcal{V}}$ are functions such that $\cdot^{\mathcal{V}}$ maps each $n \in \mathsf{N}$ to a **distinct** element in $\Delta^{\mathcal{V}}$, $\mathcal{R}^{\mathcal{V}}$ maps each $o \in \Delta^{\mathcal{V}}$ to a subset of $\Delta^{\mathcal{V}} \times \Delta^{\mathcal{V}}$, $\mathcal{C}^{\mathcal{V}}$ maps each $o \in \Delta^{\mathcal{V}}$ to a subset of $\Delta^{\mathcal{V}}$, and $\mathcal{R}^{\mathcal{V}}(\mathsf{type}^{\mathcal{V}}) = \{(e, o) | o \in \Delta^{\mathcal{V}} \wedge e \in \mathcal{C}^{\mathcal{V}}(o)\}$ holds. The interpretation of other elements is shown in Fig. 1.(a). ν-models, ν-satisfiability and ν-entailment ($\models_\nu$) are defined as usual.

Syntax	Semantics
P	$\mathcal{R}^{\mathcal{V}}(P^{\mathcal{V}})$
P^-	$\{(x, y) \| (y, x) \in \mathcal{R}^{\mathcal{V}}(P^{\mathcal{V}})\}$
$\neg S$	$\Delta^{\mathcal{V}} \times \Delta^{\mathcal{V}} - \mathcal{R}^{\mathcal{V}}(S^{\mathcal{V}})$
A	$\mathcal{C}^{\mathcal{V}}(A^{\mathcal{V}})$
$\exists S$	$\{x \| \exists y.(x, y) \in \mathcal{R}^{\mathcal{V}}(S^{\mathcal{V}})\}$

Syntax	Semantics
$\neg B$	$\Delta^{\mathcal{V}} - \mathcal{C}^{\mathcal{V}}(B^{\mathcal{V}})$
$B \sqsubseteq_c C$	$\mathcal{C}^{\mathcal{V}}(B^{\mathcal{V}}) \subseteq \mathcal{C}^{\mathcal{V}}(C^{\mathcal{V}})$
$S \sqsubseteq_r R$	$\mathcal{R}^{\mathcal{V}}(S^{\mathcal{V}}) \subseteq \mathcal{R}^{\mathcal{V}}(R^{\mathcal{V}})$
$A(a)$	$a^{\mathcal{V}} \in \mathcal{C}^{\mathcal{V}}(A^{\mathcal{V}})$
$P(a, b)$	$(a^{\mathcal{V}}, b^{\mathcal{V}}) \in \mathcal{R}^{\mathcal{V}}(P^{\mathcal{V}})$

(a)

α	$\delta(\alpha)$	α	$\delta(\alpha)$
$B \sqsubseteq \neg\exists\mathsf{type}$	$\mathsf{I}_z(B, x) \wedge y(x) \to q()$	$B \sqsubseteq \neg\exists\mathsf{type}^-$	$\mathsf{I}_z(B, x) \wedge x(y) \to q()$
$P \sqsubseteq \neg\mathsf{type}$	$\mathsf{I}(P, x, y) \wedge y(x) \to q()$	$P \sqsubseteq \neg\mathsf{type}^-$	$\mathsf{I}(P, x, y) \wedge x(y) \to q()$

where $x, y, z \in \mathsf{V}$, and if $B \in \mathsf{N}$, $\mathsf{I}_z(B, x) = B(x)$; if $B = \exists P$ and $P \in \mathsf{N}$, $\mathsf{I}_z(B, x) = P(x, z)$; if $B = \exists P^-$ and $P \in \mathsf{N}$, $\mathsf{I}_z(B, x) = P(z, x)$; if $S \in \mathsf{N}$, $\mathsf{I}(S, x, y) = S(x, y)$; and if $S = P^-$ and $P \in \mathsf{N}$, $\mathsf{I}(S, x, y) = P(y, x)$

(b)

Fig. 1. (a) Interpretation of DL-Lite Full elements w.r.t. a ν-interpretation $\mathcal{V}$, (b) translation of disjoint axioms to MQs.

In a ν-interpretation, each name is mapped to a domain element and each domain element has a class extension and a role extension, and the role extension of the interpretation of type captures the class extension of all domain elements.

For a tuple $\boldsymbol{u}$, we use $|\boldsymbol{u}|$ and $\boldsymbol{u}[i]$ to denote the length and the i-th element of $\boldsymbol{u}$, respectively. For a query Q such that $|\mathsf{head}(Q)| = |\boldsymbol{u}|$, we use $Q[\mathsf{head}(Q)/\boldsymbol{u}]$, abbreviated as $Q(\boldsymbol{u})$, to denote the result of replacing each occurrence of $\mathsf{head}(Q)[i]$ in Q with $\boldsymbol{u}[i]$ for $1 \leq i \leq |\boldsymbol{u}|$.

Semantics of meta-queries. For a MQ Q and ν-interpretation $\mathcal{V}$, a binding π of Q over $\mathcal{V}$ is a function that maps each variable in Q to an element in $\Delta^{\mathcal{V}}$ and each name a in Q to $a^{\mathcal{V}}$. We write $\mathcal{V}, \pi \models_\nu Q$ if $(\pi(y), \pi(z)) \in \mathcal{R}^{\mathcal{V}}(\pi(x))$ for each $x(y, z) \in \mathsf{body}(Q)$ and $\pi(y) \in \mathcal{C}^{\mathcal{V}}(\pi(x))$ for each $x(y) \in \mathsf{body}(Q)$. For a DL-Lite Full KB $\mathcal{K}$, a tuple $\boldsymbol{u}$ of names and with length $|\mathsf{head}(Q)|$, is called a certain answer of Q over $\mathcal{K}$ if for each ν-model $\mathcal{V}$ of $\mathcal{K}$, there exists a binding π of $Q(\boldsymbol{u})$ over $\mathcal{K}$ such that $\mathcal{V}, \pi \models_\nu Q(\boldsymbol{u})$ holds. We use $\mathsf{ans}_\nu(Q, \mathcal{K})$ to denote the set of all the certain answers of Q over $\mathcal{K}$. Notice that $\boldsymbol{u}$ can be an empty tuple $()$ in the case that Q is a Boolean query. In this situation, $\mathsf{ans}_\nu(Q, \mathcal{K})$ just contains empty tuple if Q is satisfied by every ν-model of $\mathcal{K}$ (Q is true over $\mathcal{K}$).

Overall Assumption. In the following, we just consider the KBs without the axioms containing $\neg\exists\mathsf{type}$, $\neg\exists\mathsf{type}^-$, $\neg\mathsf{type}$ and $\neg\mathsf{type}^-$ in the right-hand sides, since, as shown in the proposition below, reasoning with the KBs containing such axioms can be captured by reasoning with the KBs without such axioms via translating these axioms into MQs by function δ defined in Fig.1.(b).

Proposition 1 *For DL-Lite Full KB $\mathcal{K}$, let $\mathcal{K}'$ be the KB obtained from $\mathcal{K}$ by dropping the axioms with $\neg\exists\mathsf{type}(^-)$ or $\neg\mathsf{type}(^-)$. Then (1) $\mathcal{K}$ is ν-satisfiable iff $\mathcal{K}'$ is satisfiable, and $\mathsf{ans}_\nu(\delta(\alpha),\mathcal{K}') = \emptyset$ for each axiom α in $\mathcal{K}$ with $\neg\exists\mathsf{type}(^-)$ or $\neg\mathsf{type}(^-)$; (2) if $\mathcal{K}$ is ν-satisfiable, $\mathsf{ans}_\nu(Q,\mathcal{K}) = \mathsf{ans}_\nu(Q,\mathcal{K}')$ for each MQ Q.*

The **proofs** of this proposition and the results in the following sections are shown in the Appendix (https://github.com/Lucy321456/Files/blob/master/DL21.pdf). The motivation of making this assumption is to simplify the description of the provided method for reasoning with DL-Lite Full by first discussing satisfiability checking and CQ answering via CQ rewriting, and then based on the results, studying MQ answering via MQ rewriting. As shown in Fig.1, axioms with $\neg\exists\mathsf{type}$ ($\neg\mathsf{type}$) actually correspond to MQs rather than CQs.

3 Satisfiability checking and conjunctive query answering

Here, we discuss satisfiability checking and CQ answering in DL-Lite Full. In DL-Lite$_\mathcal{R}$, the considered reasoning tasks can be eventually reduced to evaluating queries over the data layers of KBs. Thus the basic idea is to make use of the techniques especially designed for DL-Lite$_\mathcal{R}$. Before that we need to analyze the relationships between DL-Lite Full and DL-Lite$_\mathcal{R}$.

3.1 Relationships between DL-Lite Full and DL-Lite$_\mathcal{R}$

The work like [26, 29, 40] conclude that under unique name assumption (UNA), meta-modeling, i.e., multiple uses of names, can be handled by OWL 2 Punning via renaming. However, this does not hold anymore in DL-Lite Full due to the presence of type in the TBox. Next, let's first see the translation by renaming.

$\tau_r(P) = v_r(P)$	$\tau_c(\neg B) = \neg\tau_c(B)$	$\tau(\mathcal{T}) = \{\tau(\alpha) \mid \alpha \in \mathcal{T}\}$
$\tau_r(P^-) = v_r(P)^-$	$\tau(B \sqsubseteq_c C) = \tau_c(B) \sqsubseteq \tau_c(C)$	$\tau(\mathcal{A}) = \{\tau(\alpha) \mid \alpha \in \mathcal{A}\}$
$\tau_r(\neg S) = \neg v_r(S)$	$\tau(S \sqsubseteq_r R) = \tau_r(S) \sqsubseteq \tau_r(S)$	$\tau(\mathcal{K}) = (\tau(\mathcal{T}), \tau(\mathcal{A}))$
$\tau_c(A) = v_c(A)$	$\tau(A(x)) = v_c(A)(x)$	$\tau(Q) = \bigwedge \tau(\alpha) \to q(\boldsymbol{x})$
$\tau_c(\exists S) = \exists\tau_r(S)$	$\tau(P(x,y)) = v_r(P)(x,y)$	

Fig. 2. Definition of the translation functions, where $A, P \in \mathsf{N}$, $P \neq \mathsf{type}$, $x, y \in \mathsf{N} \cup \mathsf{V}$.

Let C and R be the sets of names for DL-Lite$_\mathcal{R}$ classes and roles such that C, R and N are pairwise disjoint and have the same cardinality. For simplicity, we use N as the set for DL-Lite$_\mathcal{R}$ individuals. Let v_c and v_r be two bijective functions that map each name $a \in \mathsf{N}$ to a class name in C and a role name in R, respectively. The conversion of DL-Lite Full classes, roles, axioms, assertions, query atoms as well as DL-Lite Full KBs $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ and CQs $Q : \bigwedge \alpha \rightarrow q(\boldsymbol{x})$ by functions τ_c, τ_r and τ **via renaming** is defined in Figure 2.

In the following, for a DL-Lite$_\mathcal{R}$ KB $\mathcal{O}$ and CQ q over $\mathcal{O}$, we use $\mathsf{ans}(q, \mathcal{O})$ to denote the set of all certain answers of q over $\mathcal{O}$. The next lemma shows that renaming can still guarantee the soundness of the considered reasoning tasks.

Lemma 1 *For a DL-Lite Full KB $\mathcal{K}$, (1) if $\mathcal{K}$ is ν-satisfiable then $\tau(\mathcal{K})$ is satisfiable; and (2) $\mathsf{ans}(\tau(Q), \tau(\mathcal{K})) \subseteq \mathsf{ans}_\nu(Q, \mathcal{K})$ for each CQ Q.*

However, unlike the existing work, even adopting UNA, completeness cannot be guaranteed, since non-standard uses of type may entail extra individual assertions which further have an impact on the considered reasoning tasks.

Example 2 *Consider the following DL-Lite Full KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ where $\mathcal{T} = \{P \sqsubseteq_r \mathsf{type},\ A \sqsubseteq_c \exists \mathsf{type}^-\}$ and $\mathcal{A} = \{P(a, B),\ A(C)\}$. $\mathcal{K}$ ν-entails $B(a)$, and $Q : C(x) \rightarrow q()$ is true over $\mathcal{K}$, i.e., $\mathsf{ans}_\nu(Q, \mathcal{K}) = \{()\}$, since C is ν-entailed to have individuals. However, such conclusions are not implied by $\tau(\mathcal{K})$. Besides, if we add $C \sqsubseteq_c \neg C$ to $\mathcal{K}$, $\mathcal{K}$ is no longer ν-satisfiable, while $\tau(\mathcal{K})$ is still satisfiable.*

In order to capture the extra conclusions entailed by non-standard uses of type, an intuitive way is to materialize such entailment to $\tau(\mathcal{K})$ by these 3 steps:

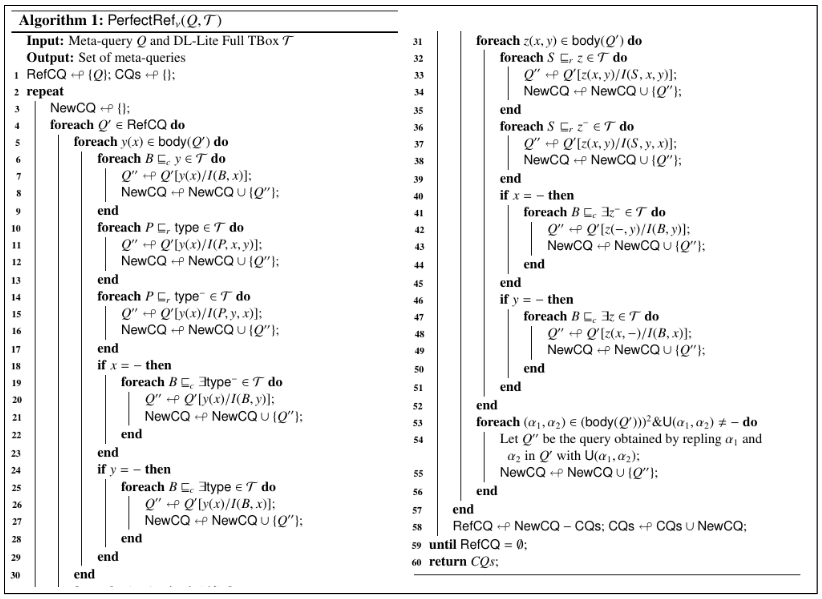
Step 1. For each $v_r(\mathsf{type})(a, A)$ entailed by $\tau(\mathcal{K})$, add $v_c(A)(a)$ to $\tau(\mathcal{K})$;
Step 2. For each $\exists v_r(\mathsf{type})^-(A)$ entailed by $\tau(\mathcal{K})$, add $v_c(A)(o)$ to $\tau(\mathcal{K})$, where $o \in \mathsf{N}$ is a fresh name not occurring in $\mathcal{K}$;
Step 3. For each $B(A)$, $B \sqsubseteq \exists S$ and $S^- \sqsubseteq v_r(\mathsf{type})$ entailed by $\tau(\mathcal{K})$, add $S(A, o)$ and $v_c(A)(o)$ to $\tau(\mathcal{A})$, where o is a fresh name not occurring in $\mathcal{K}$ and $S(A, o)$ denotes $P(A, o)$ if $S = P$ and $P(o, A)$ if $S = P^-$, where $P \in \mathsf{R}$.

For distinction, we denote the KB obtained by Step 1-3 as $\tau^{\mathsf{m}}(\mathcal{K})$. Step 2 and 3 respectively capture the situation that $\mathcal{K}$ entails A having individuals and A and $v_r^-(P)$ sharing some individuals. In the above procedure, we do not need to execute Step 2-3 again, since the fresh names o do not occur in $\mathcal{K}$, thus even if o are entailed to have individuals, such knowledge does not affect the results of satisfiability checking and answering the queries solely containing names in $\mathcal{K}$.

Theorem 1 *For a DL-Lite Full KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, (1) $\mathcal{K}$ is ν-satisfiable iff $\tau^{\mathsf{m}}(\mathcal{K})$ is satisfiable; and (2) for a CQ Q and tuple $\boldsymbol{u}$ such that $|\boldsymbol{u}| = |\mathsf{head}(Q)|$ and solely contains names occurring in $\mathcal{K}$, then $\boldsymbol{u} \in \mathsf{ans}_\nu(Q, \mathcal{K})$ iff $\boldsymbol{u} \in \mathsf{ans}(\tau(Q), \tau(\mathcal{K}))$.*

3.2 Reasoning via query rewriting

By Theorem 1, the intuitive way of realizing the reduction is to make use of the techniques for DL-Lite$_\mathcal{R}$ via encoding the inclusion axioms into the queries. The difference is that for completeness, extra encoding needs to be done to capture the non-standard use of type. For example, for an atom $A(x)$ in a query Q, we not only need to use the axiom $B \sqsubseteq_c A$ in the TBox to replace $A(x)$ in Q with $I(B, x)$ to generate a new query, but also take the axiom $P \sqsubseteq_r \mathsf{type}/\mathsf{type}^-$ into consideration to further replace $A(x)$ with $I(P, x, A)/I(P, A, x)$ to capture the individuals of A implied by the axioms referring type, where I is a query atom generating function defined in Fig.3. If x or A is a ***unbound variable*** (denoted as $-$), i.e., the variables not occurring in $\mathsf{head}(Q)$ and occurring only once in Q), $B \sqsubseteq_c \exists\mathsf{type}$ or $B \sqsubseteq_c \exists\mathsf{type}^-$ also needs to be taken into consideration.



```
Algorithm 1: PerfectRef_ν(Q, T)
   Input: Meta-query $Q$ and DL-Lite Full TBox $\mathcal{T}$
   Output: Set of meta-queries
 1 RefCQ $\hookleftarrow$ {$Q$}; CQs $\hookleftarrow$ {};
 2 repeat
 3     NewCQ $\hookleftarrow$ {};
 4     foreach $Q' \in$ RefCQ do
 5         foreach $y(x) \in \mathsf{body}(Q')$ do
 6             foreach $B \sqsubseteq_c y \in \mathcal{T}$ do
 7                 $Q'' \hookleftarrow Q'[y(x)/I(B, x)]$;
 8                 NewCQ $\hookleftarrow$ NewCQ $\cup \{Q''\}$;
 9             end
10             foreach $P \sqsubseteq_r \mathsf{type} \in \mathcal{T}$ do
11                 $Q'' \hookleftarrow Q'[y(x)/I(P, x, y)]$;
12                 NewCQ $\hookleftarrow$ NewCQ $\cup \{Q''\}$;
13             end
14             foreach $P \sqsubseteq_r \mathsf{type}^- \in \mathcal{T}$ do
15                 $Q'' \hookleftarrow Q'[y(x)/I(P, y, x)]$;
16                 NewCQ $\hookleftarrow$ NewCQ $\cup \{Q''\}$;
17             end
18             if $x = -$ then
19                 foreach $B \sqsubseteq_c \exists\mathsf{type}^- \in \mathcal{T}$ do
20                     $Q'' \hookleftarrow Q'[y(x)/I(B, y)]$;
21                     NewCQ $\hookleftarrow$ NewCQ $\cup \{Q''\}$;
22                 end
23             end
24             if $y = -$ then
25                 foreach $B \sqsubseteq_c \exists\mathsf{type} \in \mathcal{T}$ do
26                     $Q'' \hookleftarrow Q'[y(x)/I(B, x)]$;
27                     NewCQ $\hookleftarrow$ NewCQ $\cup \{Q''\}$;
28                 end
29             end
30         end
31         foreach $z(x, y) \in \mathsf{body}(Q')$ do
32             foreach $S \sqsubseteq_r z \in \mathcal{T}$ do
33                 $Q'' \hookleftarrow Q'[z(x, y)/I(S, x, y)]$;
34                 NewCQ $\hookleftarrow$ NewCQ $\cup \{Q''\}$;
35             end
36             foreach $S \sqsubseteq_r z^- \in \mathcal{T}$ do
37                 $Q'' \hookleftarrow Q'[z(x, y)/I(S, y, x)]$;
38                 NewCQ $\hookleftarrow$ NewCQ $\cup \{Q''\}$;
39             end
40             if $x = -$ then
41                 foreach $B \sqsubseteq_c \exists z^- \in \mathcal{T}$ do
42                     $Q'' \hookleftarrow Q'[z(-, y)/I(B, y)]$;
43                     NewCQ $\hookleftarrow$ NewCQ $\cup \{Q''\}$;
44                 end
45             end
46             if $y = -$ then
47                 foreach $B \sqsubseteq_c \exists z \in \mathcal{T}$ do
48                     $Q'' \hookleftarrow Q'[z(x, -)/I(B, x)]$;
49                     NewCQ $\hookleftarrow$ NewCQ $\cup \{Q''\}$;
50                 end
51             end
52         end
53         foreach $(\alpha_1, \alpha_2) \in (\mathsf{body}(Q'))^2 \& \mathsf{U}(\alpha_1, \alpha_2) \neq -$ do
54             Let $Q''$ be the query obtained by repling $\alpha_1$ and
                 $\alpha_2$ in $Q'$ with $\mathsf{U}(\alpha_1, \alpha_2)$;
55             NewCQ $\hookleftarrow$ NewCQ $\cup \{Q''\}$;
56         end
57     end
58     RefCQ $\hookleftarrow$ NewCQ $-$ CQs; CQs $\hookleftarrow$ CQs $\cup$ NewCQ;
59 until RefCQ $= \emptyset$;
60 return $CQs$;
```

$I(B, x) = B(x),$	if $B \in \mathsf{N}$
$I(B, x) = P(x, -),$	if $B = \exists P, P \in \mathsf{N}$
$I(B, x) = P(-, x),$	if $B = \exists P^-, P \in \mathsf{N}$
$I(S, x, y) = P(x, y),$	if $S = P, P \in \mathsf{N}$
$I(S, x, y) = P(y, x),$	if $S = P^-, P \in \mathsf{N}$

$\mathsf{U}(x_1(y_1, z_1), x_2(y_2, z_2)) = x_3(y_3, z_3)$
$\mathsf{U}(x_1(y_1), x_2(y_2)) = x_3(y_3)$
where for $u \in \{x, y, z\}$, conditions (1)-(3) hold:
(1) if $u_1 = -$ then $u_3 = u_2$;
(2) if $u_2 = -$ then $u_3 = u_1$;
(3) if $u_1 \neq -$ and $u_2 \neq -$ then $u_1 = u_2 = u_3$ holds.
otherwise $\mathsf{U}(x_1(y_1, z_1), x_2(y_2, z_2)) = -$ and
$\mathsf{U}(x_1(y_1), x_2(y_2)) = -$

```
Algorithm 2: Violates_ν(T)
   Input: DL-Lite Full TBox $\mathcal{T}$
   Output: Set of conjunctive queries
 1 CQs $\hookleftarrow$ {};
 2 foreach $B \sqsubseteq_c \neg B' \in \mathcal{T}$ do
 3     $Q \hookleftarrow I(B, ?x) \wedge I(B', ?x) \rightarrow q()$;
 4     CQs $\hookleftarrow$ CQs $\cup$ PerfectRef$^{cq}_\nu(Q, \mathcal{T})$;
 5 end
 6 foreach $S \sqsubseteq_r \neg S' \in \mathcal{T}$ do
 7     $Q \hookleftarrow I(S, ?x, ?y) \wedge I(S', ?x, ?y) \rightarrow q()$;
 8     CQs $\hookleftarrow$ CQs $\cup$ PerfectRef$^{cq}_\nu(Q, \mathcal{T})$;
 9 end
10 return $CQs$;
```

Fig. 3. The algorithm $\mathsf{PerfectRef}_\nu$ and $\mathsf{Violates}_\nu$.

The concrete rewriting is shown in algorithm $\mathsf{PerfectRef}_\nu$ in Fig. 3, where the atom unification operator U is also defined in Fig. 3. Before the rewriting, unbound variables in the query are replaced with "$-$'. The algorithm will also be used to rewrite MQs. Thus $\mathsf{PerfectRef}_\nu$ takes MQs as input (The x in line

5 and z in line 21 maybe variables. We will explain in the next section). For satisfiability checking, the reduction is realized by rewriting the queries corresponding to the axioms with $\neg$, and then evaluate the finally obtained queries over the ABox of KB to check whether the KB implies knowledge that validates the negative axioms. The concrete way is shown in Algorithm $\mathsf{Violates}_\nu$ in Fig. 3. The termination of these two algorithms hold trivially, and the correctness can be guaranteed by the theorem below.

Theorem 2 *For a DL-Lite Full KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, (1) $\mathcal{K}$ is ν-satisfiable iff $\mathsf{ans}_\nu(Q, (\emptyset, \mathcal{A})) = \emptyset$ for each $Q \in \mathsf{Violates}_\nu(\mathcal{T})$; (2) if $\mathcal{K}$ is ν-satisfiable, then for each CQ Q, $\mathsf{ans}_\nu(Q, \mathcal{K}) = \bigcup_{Q' \in \mathsf{PerfectRef}_\nu(Q,\mathcal{T})} \mathsf{ans}_\nu(Q', (\emptyset, \mathcal{A}))$.*

Example 3 *Consider the KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ where $\mathcal{T} = \{\exists P \sqsubseteq_c A,\ A \sqsubseteq_c \exists \mathsf{type}^-\}$ and $\mathcal{A} = \{P(B, a)\}$. For the CQ $Q : B(?x) \to q()$, by Algorithm 1, we can get $\mathsf{PerfectRef}_\nu(Q, \mathcal{T}) = \{q_1 : B(-) \to q(),\ q_2 : A(B) \to q(),\ q_3 : P(B, -) \to q()\}$. Obviously, $\mathsf{ans}_\nu(q_3, \mathcal{A}) = \{()\}$. Then by Theorem 2, $\mathsf{ans}_\nu(Q, \mathcal{K}) = \{()\}$ holds.*

Based on Theorem 2 and the algorithms in Fig.3, we can further obtain the complexity of satisfiability checking and CQ answering in DL-Lite Full.

Theorem 3 *For a DL-Lite Full KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, ν-satisfiability checking and CQ answering can be realized in PTime w.r.t. $|\mathcal{T}|$ and AC_0 w.r.t. $|\mathcal{A}|$.*

4 Meta-query answering in DL-Lite Full

In the following, for a function f, we use $\mathsf{dom}(f)$ to denote the domain of f, Qf to denote the result of replacing each occurrence of a in Q with $f(a)$ for each $a \in \mathsf{dom}(f)$, and $[Qf]$ to denote the query obtained by replacing each atom $\mathsf{type}(x, y)$ with $y(x)$ with the motivation of unifying the format of queries.

4.1 Meta-query answering via meta-variable materialization

For MQ answering, a direct way is to convert MQs into CQs by materializing meta-variables [29] with names. However, again unlike the existing work, completeness cannot be guaranteed, since axioms referring type may enable the KBs not only to entail extra class individual assertions but also to imply that a named or anonymous individual may have anonymous classes, i.e., the classes implied to exist, such as the element e shown in the example below.

Example 4 *Consider the following DL-Lite Full KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ where $\mathcal{T} = \{A \sqsubseteq_c \exists P, \exists P^- \sqsubseteq_c \exists S, P \sqsubseteq_r \mathsf{type}\}$ and $\mathcal{A} = \{A(a)\}$, and the query $Q :?x(a) \wedge S(?x, ?y) \to q()$. Obviously, $\mathsf{ans}_\nu(Q, \mathcal{K}) = \{()\}$, i.e., Q is true over $\mathcal{K}$, since there exist anonymous elements e and e' such that $e(a)$ and $S(e, e')$ are entailed by $\mathcal{K}$. However, no matter what name occurring in $\mathcal{K}$ you replace $?x$ with, the resultant query always has an empty answer set over $\mathcal{K}$.*

Fortunately, such entailment can be captured by rewriting the query atoms $?x(y)$ based on the axioms referring type *like* $P \sqsubseteq_r \mathsf{type}$ (Line 10-17 in Algorithm 1) rather than doing materialization of $?x$. Next, we formalize this approach.

Definition 3 *For a MQ Q and DL-Lite Full KB $\mathcal{K}$, a MV-Binding $\theta = (\theta_r, \theta_c)$ of Q over $\mathcal{K}$ is a function such that θ_r maps each role variable of Q to* type *or a name occurring in $\mathcal{K}$, and θ_c maps* ***some class variables*** *of $[Q\theta_r]$ to the names occurring in $\mathcal{K}$. We use $Q\theta$ to denote $[Q\theta_r]\theta_c$, and use $\mathsf{MVB}(Q, \mathcal{K})$ to denote the set of all the MV-Bindings of Q over $\mathcal{K}$.*

In Definition 3, to capture the implied anonymous class elements, θ maps some class variables of Q to names, and let the reminder class atoms $?x(y)$ to be rewritten based on the axioms referring type. The way of reducing MQ answering to CQ evaluation over the ABoxes of KBs is shown in the theorem below.

Theorem 4 *For a v-satisfiable DL-Lite Full KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, MQ Q and tuple $\boldsymbol{u}$ of names, then $\boldsymbol{u} \in \mathsf{ans}_\nu(Q, \mathcal{K})$ iff there exist MV-Binding $\theta \in \mathsf{MVB}(Q, \mathcal{K})$ and CQ $Q' \in \mathsf{PerfectRef}_\nu(Q\theta, \mathcal{T})$ such that $\boldsymbol{u} \in \mathsf{ans}_\nu(Q', (\emptyset, \mathcal{A}))$.*

Note that in Theorem 4, the set $\bigcup_{\theta \in \mathsf{MVB}(Q,\mathcal{K})} \mathsf{PerfectRef}_\nu(Q\theta, \mathcal{T})$ may contain queries with class variables. However, just evaluating the CQs in the set over the ABox is enough to obtain all the certain answers of Q over $\mathcal{K}$.

Example 5 *Consider the MQ $Q\,{:}\,?c(a) \wedge ?p(a, ?x) \rightarrow q(?c, ?p, ?x)$ and ν-satisfiable DL-Lite Full KB $\mathcal{K} = (\{A_1 \sqsubseteq_c A_2\},\ \{A_1(a)\} \cup \bigcup_{i=1}^{n} \{P_i(a, b_i)\})$, where $P_i \neq \mathsf{type}$ for $1 \leq i \leq n$. For Q, if $?p$ is bound to* type *then the resultant query has two class variables $?c$ and $?x$, otherwise it has only one class variable $?c$. Then:*

$$\begin{gathered}\mathsf{MVB}(Q, \mathcal{K}) = \\ \bigcup_{o \in \mathsf{N}_\mathcal{K} - \{\mathsf{type}\}} \{(\{?p \rightarrow o\}, \{\})\} \cup \bigcup_{o \in \mathsf{N}_\mathcal{K} - \{\mathsf{type}\}} \bigcup_{e \in \mathsf{N}_\mathcal{K}} \{(\{?p \rightarrow o\}, \{?c \rightarrow e\})\} \cup \\ \{(\{?p \rightarrow \mathsf{type}\}, \{\})\} \cup \bigcup_{o \in \mathsf{N}_\mathcal{K}} \{(\{?p \rightarrow \mathsf{type}\}, \{?c \rightarrow o\})\}\} \cup \\ \bigcup_{o \in \mathsf{N}_\mathcal{K}} \{(\{?p \rightarrow \mathsf{type}\}, \{?x \rightarrow o\})\} \cup \bigcup_{o \in \mathsf{N}_\mathcal{K}} \bigcup_{e \in \mathsf{N}_\mathcal{K}} \{(\{?p \rightarrow \mathsf{type}\}, \{?c \rightarrow o, ?x \rightarrow e\})\}\end{gathered}$$

where $\mathsf{N}_\mathcal{K}$ consists of all the names in $\mathcal{K}$. By trying all the MV-Bindings, we can get $\mathsf{ans}_\nu(Q, \mathcal{K}) = \bigcup_{i=1}^{2} \bigcup_{j=1}^{n} \{(A_i, P_j, b_j)\} \ \cup\ \bigcup_{i=1}^{2} \bigcup_{j=1}^{2} \{(A_i, \mathsf{type}, A_j)\}$. □

4.2 Meta-query answering via partial meta-variable materialization and meta-query rewriting

For completeness, not only the names occurring in the TBox but also thus solely occurring in the ABox need to be considered when materializing meta-variables. This makes a large number of candidates need to be tried when answering MQs. Besides, it will also violate the desire of the separation between TBox and ABox reasoning when answering MQs. Next based on two observations and Theorem 5, we provide a novel way of answering MQs via partial meta-variable materialization and MQ rewriting, where just the names in the TBox are referred.

The first observation is that under ν-semantics, answering a MQ over a DL-Lite Full ABox can be realized by evaluating a CQ over a database without try any meta-variable materialization, shown in the lemma below.

Lemma 2 *For a MQ Q and DL-Lite Full ABox $\mathcal{A}$, we use $\mathcal{DB}$ to denote the database $\{T(a,P,b)|P(a,b) \in \mathcal{A}\} \cup \{T(a,\mathsf{type},A)|A(a) \in \mathcal{A}\}$ and Q' the query $\bigwedge_{x(y,z)\in\mathsf{body}(Q)} T(y,x,z) \wedge \bigwedge_{x(y)\in\mathsf{body}(Q)} T(y,\mathsf{type},x) \rightarrow q(\mathsf{head}(Q))$. Then $\mathsf{ans}_v(Q,(\emptyset,\mathcal{A})) = \mathsf{ans}(Q',\mathcal{DB})$ holds.*

The second observation is that when rewriting queries by the algorithm $\mathsf{PerfectRef}_\nu$, if the names occurring in the class/role positions of query atoms do not occur in the right-hand sides of any inclusion axioms, then these query atoms will not be extended to generate new queries.

Enlightened by these, next, we provide the way of answering MQs via partial meta-variable materialization and MQ rewriting. For a DL-Lite Full TBox $\mathcal{T}$, we use $\mathsf{N}^{rc}_{\mathcal{T}}$ (resp. $\mathsf{N}^{rr}_{\mathcal{T}}$) to denote the set of all the names used as classes (resp. roles) in the right-hand sides of the inclusion axioms in $\mathcal{T}$. The way of partially materializing meta-variables of MQs is shown below.

Definition 4 *For a MQ Q and DL-Lite Full TBox $\mathcal{T}$, a partial MV-Binding $\vartheta = (\vartheta_r, \vartheta_c)$ of Q over $\mathcal{T}$ is a function such that ϑ_r maps some role variables of Q to the names in $\mathsf{N}^{rr}_{\mathcal{T}} \cup \{\mathsf{type}\}$ and ϑ_c maps some class variables of $[Q\vartheta_r]$ to the names in $\mathsf{N}^{rc}_{\mathcal{T}}$. And $(\{\},\{\})$, i.e., without binding any role and class variable of Q, is also a partial MV-Binding of Q over $\mathcal{T}$. We use $\mathsf{PMVB}(Q,\mathcal{T})$ to denote the set of all the partial MV-bindings of Q over $\mathcal{T}$.*

Partial MV-Bindings materialize both class variables and role variables **partially**. The resultant MQs will be rewritten by the algorithm $\mathsf{PerfectRef}_\nu$ directly. In the rewriting procedure, role variables $?y$ in the atoms $?y(x,y)$ will be treated as names not occurring in the TBox, thus these atoms will not be extended to generate new queries (Line 31-52 of $\mathsf{PerfectRef}_\nu$). And class variables $?x$ in the atoms $?x(y)$ are also treated as names not occurring in the TBox, and these atoms will just be rewritten based on the axioms referring type, like $P \sqsubseteq_r \mathsf{type}$ (Line 5-30 of $\mathsf{PerfectRef}_\nu$).

Next, before giving the concrete way of answering MQs via **partial** meta-variable materialization and MQ **rewriting**, we first analyze the relationships between these two ways of materializing the meta-variables of MQs.

Definition 5 *For MQ Q, KB $\mathcal{K}=(\mathcal{T},\mathcal{A})$ and partial MV-Binding $\vartheta \in \mathsf{PMVB}(Q,\mathcal{T})$, a MV-Binding $\theta \in \mathsf{MVB}(Q,\mathcal{K})$ is called an **extension** of ϑ over $\mathcal{K}$ if for each role variable x of Q, if $x \in \mathsf{dom}(\vartheta)$ then $\theta(x) = \vartheta(x)$ otherwise $\theta(x) \notin \mathsf{N}^{rr}_{\mathcal{T}}$; and for each class variable x of Q, if $x \in \mathsf{dom}(\vartheta)$ then $\theta(x) = \vartheta(x)$ otherwise $\theta(x) \notin \mathsf{N}^{rc}_{\mathcal{T}}$. We use $\mathsf{ePMVB}(\vartheta,Q,\mathcal{K})$ to denote the set of the extensions of ϑ of Q over $\mathcal{K}$.*

Actually, extensions of partial MV-Binding ϑ are the MV-Bindings obtained by mapping the remainder meta-variables, i.e., meta-variables of $Q\vartheta$, to the names not occurring in the right-hand sides of the inclusion axioms in $\mathcal{T}$.

Example 6 *(Example 5 con't) $\vartheta = (\{\},\{?c \rightarrow A_2\})$ is a partial MV-Binding of Q over $\mathcal{K}$'s TBox, and $\mathsf{ePMVB}(\vartheta,Q,\mathcal{K}) = \cup_{o\in \mathsf{N}_{\mathcal{K}} - \mathsf{N}^{rr}_{\mathcal{T}} - \{\mathsf{type}\}} \{(\{?p \rightarrow o\},\{?c \rightarrow A_2\})\}$ is the set consisting of the MV-Bindings obtained by binding the remainder role variable $?p$ to the names in $\mathsf{N}_{\mathcal{K}} - \mathsf{N}^{rr}_{\mathcal{T}} - \{\mathsf{type}\}$.*

By Definition 5, the lemma below indicates that (a) extensions of all the partial MV-Bindings can cover all the MV-Bindings; and (b) the certain answers obtained by trying all the extensions of a partial MV-Binding via MQ rewriting can be captured by answering a MQ through rewriting.

Lemma 3 *For a MQ Q and DL-Lite Full KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, we can get that* $\mathsf{MVB}(Q, \mathcal{K}) = \bigcup_{\vartheta \in \mathsf{PMVB}(Q,\mathcal{T})} \mathsf{ePMVB}(\vartheta, Q, \mathcal{K})$*, and for each* $\vartheta \in \mathsf{PMVB}(Q, \mathcal{T})$*:*

$$\bigcup_{\theta \in \mathsf{ePMVB}(\vartheta,Q,\mathcal{K})} \bigcup_{Q' \in \mathsf{PerfectRef}_\nu(Q\theta,\mathcal{T})} \mathsf{ans}_\nu(Q', (\emptyset, \mathcal{A})) \subseteq \bigcup_{Q' \in \mathsf{PerfectRef}_\nu(Q\vartheta,\mathcal{T})} \mathsf{ans}_\nu(Q', (\emptyset, \mathcal{A})) \subseteq \mathsf{ans}_\nu(Q, \mathcal{K})$$

Combing Lemma 3 and Theorem 4, we can finally obtain the way of answering MQs via partial meta-variable materialization and meta-query rewriting.

Theorem 5 *For a MQ Q and ν-satisfiable DL-Lite Full KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, then for a tuple $\boldsymbol{u}$ of names, $\boldsymbol{u} \in \mathsf{ans}_\nu(Q, \mathcal{K})$ iff there exists $\vartheta \in \mathsf{PMVB}(Q, \mathcal{T})$ and $Q' \in \mathsf{PerfectRef}_\nu(Q\vartheta, \mathcal{T})$ such that $\boldsymbol{u} \in \mathsf{ans}_\nu(Q', (\emptyset, \mathcal{A}))$ holds.*

Example 7 (Example 5 cont'd) $\mathsf{N}^{\mathsf{rr}}_{\mathcal{T}} = \emptyset$ *and* $\mathsf{N}^{\mathsf{rc}}_{\mathcal{T}} = \{A_2\}$*. By Definition 4, Q has totally the following 6 partial MV-Bindings over $\mathcal{T}$:*

$$\begin{array}{ll}
\vartheta_1 = (\{\}, \{\}) & \vartheta_4 = (\{?p \to \mathsf{type}\}, \{?c \to A_2\}) \\
\vartheta_2 = (\{\}, \{?c \to A_2\}) & \vartheta_5 = (\{?p \to \mathsf{type}\}, \{?x \to A_2\}) \\
\vartheta_3 = (\{?p \to \mathsf{type}\}, \{\}) & \vartheta_6 = (\{?p \to \mathsf{type}\}, \{?c \to A_2, ?x \to A_2\})
\end{array}$$

Then by Theorem 5, all the certain answers of Q over $\mathcal{K}$ can be obtained by evaluating over $\mathcal{A}$ the rewritten of the queries $Q\vartheta_1$–$Q\vartheta_6$ over $\mathcal{T}$. Thus, through partial meta-variable materialization, we just need 6 times of query rewriting rather than $2 \times (2n + 3) + 2 \times (2n + 3)^2$ times (n is shown in Example 5).

KB	Num_C	Num_rC	Num_rC /Num_C	Num_R	Num_rR	Num_rR/Num_R
BTC2012	531,637	92,495	17%	74,932	3,936	5%
DBpedia	225,416	19,269	8%	41,762	468	1%

Fig. 4. Statistics of the names used as classes/roles in BTC 2012 dataset and DBpedia, where Num_C (Num_R) denotes the number of names used as classes (roles), and Num_rC (Num_rR) number of names used as classes (roles) in the right-hand sides of axioms.

Just considering the names occurring in the right-hand sides of the inclusion axioms can significantly reduce the number of candidates needed to be tried for the meta-variables of MQs Q, and further the number of queries eventually needed to be considered over the ABoxes of the KBs. This can be seen from the statistics of two actual KBs shown in Fig. 4.

Finally, based on Lemma 2 and Theorem 5, we can further obtain the complexity of MQ answering in DL-Lite Full.

Theorem 6 *For DL-Lite Full KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ and MQ Q, $\mathsf{ans}_\nu(Q, \mathcal{K})$ can be obtained in* PTime *w.r.t. $|\mathcal{K}|$ and AC^0 w.r.t. $|\mathcal{A}|$.*

5 Related work

The work [26–32, 34–36] focused on extending multiple uses of names in expressive DLs, such as $\mathcal{SHOIN}$ [37] and $\mathcal{SROIQ}$ [38]. Compared with these work, we concentrate on a light-weight DL, i.e., DL-Lite$_\mathcal{R}$, and study not only multiple uses of names but also non-standard uses of rdf:type.

For light-weight DLs, [40–44] discuss extending DL-Lite$_\mathcal{R}$ with multiple uses of names and MQ answering. The main difference between their work and ours embodies in the following aspects. We study non-standard uses of rdf:type in addition. As shown in Example 2 and 4, non-standard uses of rdf:type will enable a KB to entail a class having extra named individuals or anonymous individuals and individuals having anonymous classes, i.e., the classes entailed to exist. This makes punning and full materialization of the meta-variables of MQs, i.e., translating MQs into CQs via replacing all meta-variables with names, cannot guarantee the completeness of satisfiability checking and MQ answering anymore. Thus, we further provide the method for the considered reasoning tasks via partial meta-variable materialization and MQ rewriting. Here, meta-variables of MQs are partially materialized using the names occurring in the right-hand sides of axioms. This can not only guarantee the completeness of the considered reasoning tasks but also significantly reduce the number of queries needed to be evaluated over the data layers of KBs. There just exists one work, i.e., [39], discussing meta-modeling with instantiation in $\mathcal{SROIQ}$ and $\mathcal{SHOIQ}$, where non-standard use of rdf:type can be captured. The authors provide methods of reducing satisfiability checking in the extended languages to satisfiability checking in the DL languages. However, this technique cannot be applied to DL-Lite$_\mathcal{R}$, since DL-Lite$_\mathcal{R}$ does not support the constructors, like enumeration, used when translating the KBs in the extended languages to DL KBs.

6 Conclusion and future work

We studied encoding multiple uses of names and non-standard uses of rdf:type, in DL-Lite$_\mathcal{R}$, and proposed a sub-language of OWL 2 Full called DL-Lite Full. This paper focuses on developing the theoretical aspects of DL-Lite Full for the considered reasoning tasks. Future work will mainly focus on the following aspects. Even with partial meta-variable materialization, answering a MQ over a KB may still need to answer many MQs via MQ rewriting. Heuristics like the ones in our previous work [28,30] need to be developed to optimize the procedure of MQ answering. Besides, DL-Lite Full just captures the non-standard uses of rdfs : type. Extending DL-Lite Full to capture the non-standard uses of other RDF(S)/OWL vocabulary terms as well as studying the rewriting ability and complexity of such extensions is also worth studying.

Acknowledgments

We thank the reviewers and program committee for their valuable comments and suggestions which will help us a lot to improve the whole work.

References

1. O'Riain, S., Curry, E., and Harth, A.: XBRL and open data for global financial ecosystems: A linked data approach. The International Journal of Accounting Information Systems 13(2): 141–162 (2012).
2. Polleres, A., and Steyskal, S.: Semantic Web Standards for Publishing and Integrating Open Data. In: Standards and Standardization: Concepts, Methodologies, Tools, and Applications (2015).
3. Bizer, C., Heath, T., and Berners-Lee, T.: Linked data-the story so far. J. on Semantic Web and Information Systems 5(3) 1–22 (2009).
4. Gu, Z., Zhang, S., and Cao, C.: Reasoning and querying web-scale open data based on DL-LiteA in a divide-and-conquer way. J. Web Semant 55: 122–144 (2019).
5. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., and Patel-Schneider, P., eds.: The Description Logic Handbook: Theory, Implementation, and Applications. Cambrige University Press, second edition (2007).
6. Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., and Rosati, R.: Linking data to ontologies. *Journal on Data Semantics* 10(2008): 133–173 (2008).
7. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., and Rosati, R.: Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *Journal of automated reasoning* 39(3): 385–429 (2009).
8. Artale, A., Calvanese. D., Kontchakov, R., and Zakharyaschev, M.: DL-Lite without UNA. In *Proceedings of DL'09* (2009).
9. Baader, F., Brandt, S., and Lutz, C.: Pushing the EL envelope. In *Proceedings of IJCAI'05* (2005).
10. Grosof, B. N., Horrocks, I., Volz, R., and Decker, S.: Description logic programs: Combining logic programs with description logic. In Proceedings of the 12th international conference on World Wide Web (2003).
11. Xiao, G., Calvanese, D., Kontchakov, R., Lembo, D., Poggi, A., Rosati, R., and Zakharyaschev, M.: Ontology-Based Data Access: A Survey. In *Proceedings of IJCAI'18* (2018).
12. Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P. F., and Rudolph, S.: OWL 2 Web Ontology Language Primer (Second Edition). W3C Recommendation (2012).
13. Hayes, P. J., and Patel-Schneider, P. F.: RDF 1.1 Semantics. W3C Recommendation (2014).
14. Bishop, B., Kiryakov, A., Ognyanov, D., Peikov, I., Tashev, Z., and Velkov, R.: FactForge: A fast track to the web of Data. *Semantic Web Journal* 2(2): 157–166 (2011).
15. Umbrich, J., Hogan, A., Polleres, A., and Decker, S.: Link Traversal Querying for a diverse Web of Data. *Semantic Web Journal* 6(6): 585–624 (2012).
16. OWL 2 Punning, `https://www.w3.org/TR/owl2-new-features/#F12:\_Punning`. Last accessed 21 Jun 2021.
17. SUMO ontology, `http://www.adampease.org/OP/`. Last accessed 21 Jun 2021.
18. OpenCyc Ontology, `http://sw.opencyc.org/`. Last accessed 21 Jun 2021.
19. Life Science Ontologies, `https://bioportal.bioontology.org/ontologies`. Last accessed 21 Jun 2021.
20. FMA ontology, `https://www.bioontology.org/wiki/index.php/FMAInOwl`. Last accessed 21 Jun 2021.
21. BTC data set, `https://km.aifb.kit.edu/projects/btc-2012/`. Last accessed 21 Jun 2021

22. Hogan, A.: Exploiting RDFS and OWL for Integrating Heterogeneous, Large-Scale, Linked Data Corpora. Ph.D. thesis, National University of Ireland, Galway, 2011.
23. Schneider, M.: OWL 2 Web Ontology Language RDF-Based Semantics (Second Edition). W3C Recommendation (2012).
24. Golbreich, C., and Wallace, E. K.: OWL 2 Web Ontology Language New Features and Rationale (Second Edition). W3C Recommendation (2012).
25. Chen, W., Kifer, M., and Warren, D. S.: HILOG: a foundation for higher-order logic programming. *Journal of Logic Programming* 15(3): 187–230 (1993).
26. Motik, B.: On the Properties of Metamodeling in OWL. *Journal of Logic and Computation* 17(4): 617–637 (2007).
27. De Giacomo, G., Lenzerini, M., and Rosati, R.: Higher-Order Description Logics for Domain Metamodeling. In *Proceedings of AAAI'11* (2011).
28. Gu, Z., and Zhang, S.: Querying Large and Expressive Biomedical Ontologies. In *Proceedings of HPCC'15* (2015).
29. Gu, Z.: Meta-modeling extension of Horn-SROIQ and Query Answering. In *Proceedings of DL'16* (2016).
30. Gu, Z., and Zhang, S.: The more irresistible Hi(SRIQ) for meta-modeling and meta-query answering. *Frontiers of Computer Science* 12(5): 1029–1031 (2018).
31. Gu, Z., Cao, C., and Zhang, S.: An Expressive Sub-language of OWL 2 Full for Domain Meta-modeling. In *Proceedings of DL'19* (2019).
32. Pan, J. Z., and Horrocks, I.: OWL FA: A Metamodeling Extension of OWL DL. In *Proceedings of WWW'06* (2006).
33. Homola, M., Kl'uka, J., Svátek, V., and Vacura, M.: Typed higher-order variant of SROIQ - why not?. In *Proceedings of DL'14* (2014).
34. Motz, R., Rohrer, E., Severi, P.: Reasoning for ALCQ extended with a flexible meta-modeling hierarchy. In: 4th Joint International Semantic Technology Conference (2014).
35. Motz, R., Rohrer, E., and Severi, P.: The description logic SHIQ with a flexible meta-modeling hierarchy. *Journal of Web Semantics* 35(2015): 214–234 (2015).
36. Glimm, B., Rudolph, S., and Völker, J.: Integrated metamodeling and diagnosis in OWL 2. In *Proceedings of ISWC'10* (2010).
37. Horrocks, I., Sattler, U.: A tableau decision procedure for $\mathcal{SHOIQ}$. Journal of automated reasoning, 39(3), 249-276, 2007.
38. Horrocks, I., Kutz, O., Sattler, U.: The Even More Irresistible SROIQ. In: 10th International Conference on Principles of Knowledge Representation and Reasoning, 2006.
39. Kubincová, P., Kl'uka, J., and Homola, M.: Towards Expressive Metamodelling with Instantiation. In *Proceedings of DL'15* (2015).
40. De Giacomo, G.; Lenzerini, M.; and Rosati, R. 2011. Higher-Order Description Logics for Domain Metamodeling. In *Proceedings of AAAI'11*.
41. Lenzerini, M., Lepore, L., and Poggi, A.: Practical higher-order query answering over Hi(DL-Lite$_R$) knowledge bases. In *Proceedings of DL'14* (2014).
42. Lenzerini, M., Lepore, L., and Poggi, A.: Answering meta-queries over Hi(OWL2QL) ontologies. In *Proceedings of IJCAI'16* (2016).
43. Lenzerini, M., Lepore, L., and Poggi, A.: Metaquerying made practical for OWL 2 QL ontologies. *Information Systems* 88(2020) 101294 (2020).
44. Lenzerini, M., Lepore, L., and Poggi, A.: Metamodeling and metaquerying in OWL 2 QL. *Artificial Intelligence* 292 (2021).
45. Patel-Schneider, P. F.: Reasoning in RDFS is inherently serial, at least in the worst case. In Proceedings of the 11th International Semantic Web Conference, Posters & Demonstrations Track. (2012).