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

Zhenzhen Gu[1] & 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

unibz KRDB

中国科学院数学与系统科学研究院 Academy of Mathematics and Systems Science Chinese Academy of Sciences

Background and Research Problem

Meta-modeling, usually denoting multiple uses of names like using a name as a class, property and individual simultaneously, has been studied in various description logics (DLs).

However, the other kind of meta-modeling, i.e., non-standard uses of RDFS/OWL vocabulary terms like asserting P to be a sub-property of rdfs:subClassOf, has rarely been discussed.

In order to capture the massive and variety meta-modeling described in the real-world and large-scale knowledge bases (KBs), we address the problem of extending not only multiple uses of names but also non-standard uses of rdf:type in the light-weight DL DL-Lite$_{\mathcal{R}}$.

Main Contributions

- We provide a way of encoding multiple uses of names and 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 meta-knowledge accessing, meta-queries are introduced.
- For data-layer scalability, we provide the way of reducing statisfiability checking and meta-query answering in DL-Lite Full to evaluating queries over the data-layers of KBs via meta-query rewriting and partial variable materialization
- We prove that the considered reasoning tasks in DL-Lite Full still have AC^0 data complexity and PTime KB complexity

DL-Lite Full and Meta-queries

DL-Lite Full is defined by extending DL-Lite$_{\mathcal{R}}$ in the following two aspects:

- Do not distinguish the names for classes, roles and individuals. This means that DL-Lite Full classes, roles and individuals are defined from a same name set $\mathcal{N}$;
- rdf:type can occur in the right-hand sides of inclusion axioms and be used as individuals.

By this way, multiple uses of names as well as the specifications of rdf:type, such as the following axioms, can be captured by DL-Lite Full.

$$\textit{sem:type} \sqsubseteq_r \mathsf{rdf:type}$$
$$\textit{sem:hasActor} \sqsubseteq_r \textit{sem:type}$$

A DL-Lite Full KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ consists of a TBox $\mathcal{T}$ which is a set of class (role) inclusion axioms $\sqsubseteq_c$ ($\sqsubseteq_r$) and an ABox $\mathcal{A}$ which is a set of individual assertions.

Meta-queries Q are defined by allowing variables to occur in the class and role positions of conjunctive queries. For example, " *asking for the relationships that Lucy has*" can be represented as:

$$?p(\textit{Lucy}, ?x) \wedge ?c(?x) \rightarrow q(?p, ?x, ?c)$$

The variables occurring in the class or role positions, like $?p$ and $?c$, are called meta-variables of Q.

The symbol $\mathsf{Answer}(Q, \mathcal{K})$ is used to denote the set of all the answers of a meta-query Q over $\mathcal{K}$.

Reasoning Method Overview

The existing work conclude that under unique name assumption, meta-modeling, i.e., multiple uses of names, can be handled by OWL 2 punning via renaming, and meta-queries can be answered by translating meta-queries into conjunctive queries via materializing meta-variables with names.

However, these do not hold anymore in DL-Lite Full due to the presence of rdf:type in the TBox which makes the KBs to entail extra individual assertions. For example, the axioms and assertions:

$$P \sqsubseteq_r \mathsf{rdf:type},\ A \sqsubseteq_c \exists \mathsf{rdf:type}^-,\ \exists S \sqsubseteq \exists \mathsf{rdf:type}$$
$$P(a, B),\ A(C),\ S(c, e)$$

entail the assertions $B(a)$, $C(o)$ and $o'(c)$ which cannot implied by OWL 2 Punning, where o and o' are anonymous element. The extra entailed individual assertions will further affect the results of satisfiability checking and meta-query answer.

For data layer scalability, the basic idea is to extend the classical query rewriting algorithm PerfectRef of DL-Lite$_{\mathcal{R}}$ to capture the extra knowledge entailed by the non-standard uses of rdf:type. For example, for a query atom $A(?x)$ in a query Q, it is not only rewritten as $B(?x)$ by the axioms $B \sqsubseteq_c A$ to generate new queries, but also rewritten as $P(?x, A)$ by the axiom $P \sqsubseteq_r \mathsf{rdf:type}$ to capture the non-standard uses of rdf:type.

The overall extended algorithm PerfectRef$_\nu$ is shown in Figure 1. For generality, it takes meta-queries as input. In DL-Lite Full, the reduction from satisfiability checking, conjunctive query answering, and meta-query answering to evaluating queries over ABox is realized by this algorithm.

Algorithm 1: PerfectRef$_\nu(Q, \mathcal{T})$

```
Input: Meta-query Q and DL-Lite Full TBox T
Output: Set of meta-queries
RefCQ ← {Q}; CQs ← {};
repeat
    NewCQ ← {};
    foreach Q′ ∈ RefCQ do
        foreach y(x) ∈ body(Q′) do
            foreach B ⊑c y ∈ T do
                Q″ ← Q′[y(x)/I(B, x)];
                NewCQ ← NewCQ ∪ {Q″};
            end
            foreach P ⊑r type ∈ T do
                Q″ ← Q′[y(x)/I(P, x, y)];
                NewCQ ← NewCQ ∪ {Q″};
            end
            foreach P ⊑r type⁻ ∈ T do
                Q″ ← Q′[y(x)/I(P, y, x)];
                NewCQ ← NewCQ ∪ {Q″};
            end
            if x = − then
                foreach B ⊑c ∃type⁻ ∈ T do
                    Q″ ← Q′[y(x)/I(B, y)];
                    NewCQ ← NewCQ ∪ {Q″};
                end
            end
            if y = − then
                foreach B ⊑c ∃type ∈ T do
                    Q″ ← Q′[y(x)/I(B, x)];
                    NewCQ ← NewCQ ∪ {Q″};
                end
            end
        end
        foreach z(x, y) ∈ body(Q′) do
            foreach S ⊑r z ∈ T do
                Q″ ← Q′[z(x, y)/I(S, x, y)];
                NewCQ ← NewCQ ∪ {Q″};
            end
            foreach S ⊑r z⁻ ∈ T do
                Q″ ← Q′[z(x, y)/I(S, y, x)];
                NewCQ ← NewCQ ∪ {Q″};
            end
            if x = − then
                foreach B ⊑c ∃z⁻ ∈ T do
                    Q″ ← Q′[z(−, y)/I(B, y)];
                    NewCQ ← NewCQ ∪ {Q″};
                end
            end
            if y = − then
                foreach B ⊑c ∃z ∈ T do
                    Q″ ← Q′[z(x, −)/I(B, x)];
                    NewCQ ← NewCQ ∪ {Q″};
                end
            end
        end
        foreach (α1, α2) ∈ (body(Q′))²&U(α1, α2) ≠ − do
            Let Q″ be the query obtained by repling α1 and
             α2 in Q′ with U(α1, α2);
            NewCQ ← NewCQ ∪ {Q″};
        end
    end
    RefCQ ← NewCQ − CQs; CQs ← CQs ∪ NewCQ;
until RefCQ = ∅;
return CQs;
```

$$\begin{array}{ll}
I(B, x) = B(x), & \text{if } B \in \mathsf{N} \\
I(B, x) = P(x, -), & \text{if } B = \exists P, P \in \mathsf{N} \\
I(B, x) = P(-, x), & \text{if } B = \exists P^-, P \in \mathsf{N} \\
I(S, x, y) = P(x, y), & \text{if } S = P, P \in \mathsf{N} \\
I(S, x, y) = P(y, x), & \text{if } S = P^-, P \in \mathsf{N}
\end{array}$$

$\mathsf{U}(x_1(y_1, z_1), x_2(y_2, z_2)) = x_3(y_3, z_3)$
$\mathsf{U}(x_1(y_1), x_2(y_2)) = x_3(y_3)$
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$_\nu(\mathcal{T})$

```
Input: DL-Lite Full TBox T
Output: Set of conjunctive queries
CQs ← {};
foreach B ⊑c ¬B′ ∈ T do
    Q ← I(B, ?x) ∧ I(B′, ?x) → q();
    CQs ← CQs ∪ PerfectRef_ν^cq(Q, T);
end
foreach S ⊑r ¬S′ ∈ T do
    Q ← I(S, ?x, ?y) ∧ I(S′, ?x, ?y) → q();
    CQs ← CQs ∪ PerfectRef_ν^cq(Q, T);
end
return CQs;
```

Figure 1: The query rewriting algorithm PerfectRef$_\nu$ and translation axiom Violates$_\nu$

Reasoning Reduction via Query Rewriting

Given a DL-Lite Full KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, then $\mathcal{K}$ is satisfiable *iff* the following equation holds:

$$\bigcup_{q \in \mathsf{Violates}_\nu(\mathcal{T})} \mathsf{Answer}(q, (\emptyset, \mathcal{A})) = \emptyset \tag{1}$$

If $\mathcal{K}$ is satisfiable, then for each conjunctive query q, the equation holds:

$$\mathsf{Answer}(q, \mathcal{K}) = \bigcup_{q' \in \mathsf{PerfectRef}_\nu(q, \mathcal{T})} \mathsf{Answer}(q', (\emptyset, \mathcal{A})) \tag{2}$$

Given a meta-query Q, the reduction of answering Q over $\mathcal{K}$ is realized by first partially materializing the meta-variables of Q using the corresponding names in $\mathcal{T}$ and then answering the resultant queries via rewriting.

A partial binding of Q over $\mathcal{T}$ is a tuple $\vartheta = (\vartheta_r, \vartheta_c)$ of functions such that ϑ_r maps some role variables of Q to rdf:type or the names used as roles in the right-hands sides of inclusion axioms in $\mathcal{T}$ and ϑ_c maps some class variables of $Q\vartheta_r$ to the names used as classes in the right-hand sides of the inclusion axioms in $\mathcal{T}$.

Then by trying each way of materializing the meta-variables of Q over $\mathcal{T}$ partially, all the answers of Q over $\mathcal{K}$ can be obtained, i.e.:

$$\mathsf{Answer}(Q, \mathcal{K}) = \bigcup_{\vartheta \in \mathcal{PB}} \bigcup_{Q' \in \mathsf{PerfectRef}_\nu(Q\vartheta, \mathcal{T})} \mathsf{Answer}(Q', (\emptyset, \mathcal{A})) \tag{3}$$

where $\mathcal{PB}$ denotes the set of all the partial meta-variable bindings of Q over $\mathcal{T}$.

Compared with full meta-variable materialization, partial materialization by just considering the names occurring in the right-hand sides of the inclusion axioms can significantly reduces the number of queries eventually needed to be considered over the data layers of the KBs. This can be seen from the statistics of two actual KBs shown in the table below.

Reasoning Complexity

The above results indicate that satisfiability checking, conjunctive query answering and meta-query answering in DL-Lite Full still have AC^0 data complexity and PTime KB complexity.

Forthcoming Research

The future research mainly respects in capturing the non-standard uses of other RDF(S)/OWL vocabulary terms as well as optimizing the overall approach of meta-query answering in DL-Lite Full.