Extending *DL-Lite*$_{\mathcal{A}}$ with (Singleton) Nominals

Maxim G. Haddad and Diego Calvanese

KRDB Research Centre for Knowledge and Data
Free University of Bozen-Bolzano
Piazza Domenicani 3, Bolzano, Italy
lastname@inf.unibz.it

Abstract. In this paper we study the extension of description logics of the *DL-Lite* family with singleton nominals, which correspond in OWL 2 to the *ObjectHasValue* construct. Differently from arbitrary (non-singleton) nominals, which make query answering intractable in data complexity, we show that both knowledge base satisfiability and conjunctive query answering stay first-order rewritable when *DL-Lite*$_{\mathcal{A}}$ is extended with singleton nominals. Our technique is based on a practically implementable algorithm based on rewriting rules, in the style of those implemented in current state-of-the-art OBDA systems based on *DL-Lite*, such as Quest. This allows us to follow the tradition of the *DL-Lite* family for employing relational database technology for query answering with optimal data complexity.

1 Introduction

The *DL-Lite* family [7] is a family of descriptions logics (DLs) which is designed for optimal data complexity of reasoning and query answering, while maintaining enough expressive power to capture basic ontology and conceptual data modeling languages [6]. It is at the basis of the OWL 2 QL profile of the *Web Ontology Language* (OWL) [11].

The *DL-Lite* family has been investigated thoroughly in recent years, and several extensions with respect to the original set of constructs studied in [7] have been proposed [1,5,14,8]. In particular, *DL-Lite*$_{\mathcal{A}}$ [12,6] is a significant representative of the *DL-Lite* family, featuring the the possibility of expressing both functionality of roles and role inclusion assertions.

Nominals, i.e., concepts interpreted as a singleton, are a significant construct in DLs, investigated both for expressive DLs [16], including OWL 2 [4], and for lightweight ones [2]. It is known that the addition to *DL-Lite* of arbitrary nominals, i.e., concepts interpreted as a given (in general non-singleton) set of individuals causes data complexity of query answering to become intractable [15]. However, the impact of *singleton nominals* only, i.e., concepts interpreted as a single individual, has not been investigated so far for the DLs of the *DL-Lite* family. Such construct corresponds to the *ObjectHasValue* construct in the context of OWL 2.

In this paper, we fill this gap, and present *DL-Lite*$^{o}_{\mathcal{A}}$, which extends *DL-Lite*$_{\mathcal{A}}$ with singleton nominals. We concentrate on the two most significant reasoning

problems investigated in the context of *DL-Lite*, namely knowledge base satisfiablity, and conjunctive query answering. We show that under this extension both inference tasks stay first-order rewritable. Our technique is based on a practically implementable algorithm based on rewriting rules, in the style of those implemented in current state-of-the-art OBDA systems based on *DL-Lite*, such as QuOnto and Quest. This allows us to follow the tradition of the *DL-Lite* family for employing relational database technology for query answering with optimal data complexity.

After some preliminaries in Section 2, we introduce *DL-Lite*$^o_{\mathcal{A}}$ in Section 3. We discuss, in Section 4, the problem of query answering for satisfiable knowledge bases, and then show, in Section 5, how to rely on it to address knowledge base satisfiability. We draw some conclusions in Section 6. Proofs of most theorems can be found in the appendix.

2 The Description Logic *DL-Lite*$^o_{\mathcal{A}}$

We introduce the technical preliminaries, and the lightweight DL *DL-Lite*$_{\mathcal{A}}$ [12,6], on which we base our results.

In *DL-Lite*$_{\mathcal{A}}$[1], starting from atomic concepts and atomic roles, respectively denoted by A and P (possibly with subscripts), we can build *basic concepts* B and *basic roles* R according to the following syntax:

$$B \longrightarrow A \mid \exists R \qquad\qquad R \longrightarrow P \mid P^-$$

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

$B_1 \sqsubseteq B_2$	(*concept inclusion assertion*)
$B_1 \sqsubseteq \neg B_2$	(*concept disjointness assertion*)
$R_1 \sqsubseteq R_2$	(*role inclusion assertion*)
$R_1 \sqsubseteq \neg R_2$	(*role disjointness assertion*)
(funct R)	(*functionality assertion*)

In order to guarantee the good computational properties of *DL-Lite*$_{\mathcal{A}}$, in the TBox we must avoid the interaction between functionality assertions and role inclusion assertions that would be caused by allowing a functional role to be specialized [12]. Formally, if the TBox contains (funct P) or (funct P^-), then it cannot contain an assertion of the form $P' \sqsubseteq P$ or $P' \sqsubseteq P^-$, for some role P'.

An *ABox* $\mathcal{A}$ is a finite set of *membership assertions* of the form $A(d)$, $\neg A(d)$, $P(d, d')$, or $\neg P(d, d')$, where d, d' denote individuals. In the following, We use R^- to denote P^- if $R = P$, and P if $R = P^-$. Similarly, we use $R(x, y)$ to denote $P(x, y)$ if $R = P$, and $P(y, x)$ if $R = P^-$. A TBox $\mathcal{T}$ and an ABox $\mathcal{A}$ constitute a *knowledge base* (KB) $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$.

[1] We do not distinguish here between data values and objects, and hence do not introduce datatypes and attributes as done in [12], since they do not affect reasoning as far as the results in this paper are concerned.

The formal semantics in *DL-Lite*$_\mathcal{A}$ is given in the standard way, by relying on first-order *interpretations* $\mathcal{I} = \langle \Delta^\mathcal{I}, \cdot^\mathcal{I} \rangle$. We refer to [3] for more details, and just observe that *DL-Lite*$_\mathcal{A}$ adopts the *unique name assumption* (UNA), i.e., syntactically different individuals are interpreted as different domain elements. We make use of the standard notions of *satisfaction* and *model*. In this paper, we will focus on two reasoning problems, namely, query answering and KB satisfiability, defined in the standard way [7]. The *KB satisfiability problem* is to check, given a KB $\mathcal{K}$, whether $\mathcal{K}$ admits at least one model. To define query answering, we present some definitions.

A *conjunctive query* (CQ) q over a KB $\mathcal{K}$ is a first-order formula of the form: $q(\boldsymbol{x}) = \exists \boldsymbol{y}.conj(\boldsymbol{x}, \boldsymbol{y})$, such that $conj(\boldsymbol{x}, \boldsymbol{y})$ is a conjunction of atoms of the form $A(t)$ and $P(t, t')$, where A and P are respectively an atomic concept and an atomic role of $\mathcal{K}$, and t, t' are terms, i.e., constants in $\mathcal{K}$ or variables in $\boldsymbol{x}$ and $\boldsymbol{y}$. The free variables $\boldsymbol{x}$ of $q(\boldsymbol{x})$ are also called *distinguished variables*, and their number is called the *arity* of q. A *boolean query* is a query without distinguished variables. A *union of conjunctive queries* (UCQ) is a disjunction $q(\boldsymbol{x}) = \bigvee_{i=1,\ldots,n} \exists \boldsymbol{y_i}.conj_i(\boldsymbol{x}, \boldsymbol{y_i})$ of CQs of the same arity. We sometimes use the Datalog notation for (U)CQs:

$$\begin{aligned} q(\boldsymbol{x}) &\leftarrow \exists \boldsymbol{y_1}.conj_1(\boldsymbol{x}, \boldsymbol{y_1}) \\ &\cdots \\ q(\boldsymbol{x}) &\leftarrow \exists \boldsymbol{y_n}.conj_n(\boldsymbol{x}, \boldsymbol{y_n}) \end{aligned}$$

Given an interpretation $\mathcal{I}$, we denote by $q^\mathcal{I}$ the set of tuples of elements of $\Delta^\mathcal{I}$ obtained by evaluating q in $\mathcal{I}$. The *certain answers* of q over a KB $\mathcal{K}$ is the set $ans(q, \mathcal{K})$ of tuples $\boldsymbol{a}$ of constants appearing in $\mathcal{K}$, such that $\boldsymbol{a}^\mathcal{I} \in q^\mathcal{I}$ for every model $\mathcal{I}$ of $\mathcal{K}$. If $\mathcal{K}$ is unsatisfiable, then $ans(q, \mathcal{K})$ is trivially the set of all possible tuples of constants in $\mathcal{K}$ whose arity is the same as that of the query. We denote such a set by $AllTup(q, \mathcal{K})$. The problem of *query answering* is the computation of the set of *certain answers* for given (U)CQ q and KB $\mathcal{K}$.

Both for KB satisfiability and for query answering, we are interested in the *data complexity*, which is the complexity of the problem measured in the size of the ABox only (i.e., assuming the TBox and the query to be fixed).

3 Adding Nominals to *DL-Lite*$_\mathcal{A}$

We present now the DL *DL-Lite*$^o_\mathcal{A}$, which extends *DL-Lite*$_\mathcal{A}$ with nominals. Specifically, in *DL-Lite*$^o_\mathcal{A}$, we allow for using as basic concepts also *nominals*, i.e., concepts of the form $\{d\}$, which are interpreted as the singleton denoted by the individual d. Hence, basic concepts are built according to the following syntax:

$$B \longrightarrow A \mid \exists R \mid \{d\}$$

Apart from that, *DL-Lite*$^o_\mathcal{A}$ is defined exactly as *DL-Lite*$_\mathcal{A}$.

We observe that, due to the lack of disjunction in the DLs of the *DL-Lite* family, we restrict the attention to so-called *singleton nominals*, which cannot be composed using disjunction into *multiple element nominals*.

Nominals in *DL-Lite*$^o_{\mathcal{A}}$ may appear in the left-hand and in the right-hand side of concept inclusion and disjointness assertions, and the two kinds of occurrences play quite different roles in expressiveness and inference. Indeed, a concept inclusion assertion $\{d\} \sqsubseteq A$ with a nominal $\{d\}$ in the left-hand side, corresponds to an ABox assertion $A(d)$. Similarly, a concept disjointness assertion $A \sqsubseteq \neg\{d\}$ (or its equivalent form $\{d\} \sqsubseteq \neg A$) corresponds to an ABox assertion $\neg A(d)$. Hence, we can eliminate these two kinds of assertions by moving them to the ABox. Observe also that an assertion of the form $\{d\} \sqsubseteq \{d'\}$, where $d \neq d'$, is always satisfied due to the UNA, and hence can be safely eliminated from the TBox.

Formally, we call a *DL-Lite*$^o_{\mathcal{A}}$ TBox *normalized*, if all its nominals appear only in concept inclusion assertions of the form:

$$B \sqsubseteq \{d\} \tag{1}$$

A *DL-Lite*$^o_{\mathcal{A}}$ KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is normalized if $\mathcal{T}$ is so.

We can transform each *DL-Lite*$^o_{\mathcal{A}}$ KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ into an equivalent normalized KB $\mathcal{K}_n = \langle \mathcal{T}_n, \mathcal{A}_n \rangle$ as follows:

1. Initialize $\mathcal{T}_n$ to $\mathcal{T}$ and $\mathcal{A}_n$ to $\mathcal{A}$.
2. For each assertion $\alpha \in \mathcal{T}_n$ of the form $\{d\} \sqsubseteq \neg\{d'\}$, where $d \neq d'$, remove α from $\mathcal{T}_n$.
3. For each assertion $\alpha \in \mathcal{T}_n$ of the form $B \sqsubseteq \neg\{d\}$, where B is not a nominal, replace in $\mathcal{T}_n$ α with $\{d\} \sqsubseteq \neg B$.
4. For each nominal $\{d\}$ appearing in the left-hand side of a concept inclusion or disjointness assertion $\{d\} \sqsubseteq C$ in $\mathcal{T}_n$, introduce a fresh atomic concept A_d.
5. For each concept inclusion or disjointness assertion $\alpha \in \mathcal{T}_n$ of the form $\{d\} \sqsubseteq C$, replace α in $\mathcal{T}_n$ with $A_d \sqsubseteq C$ and add $A_d(d)$ to $\mathcal{A}_n$.

Notice that Steps 4 and 5 above deal correctly also with the case of an (unsatisfiable) inclusion assertion of the form $\{d\} \sqsubseteq \{d'\}$, which for $d \neq d'$ generate the assertions $A_d(d)$ and $A_d \sqsubseteq d'$.

It is easy to see that the above construction of $\mathcal{K}_n$ from $\mathcal{K}$ can be done in logarithmic space in the size of $\mathcal{K}$. The following result is an easy consequence of the fact that each of the above transformations preserves the semantics of the KB, and hence $\mathcal{K}_n$ is a model-conservative extension of $\mathcal{K}$ [10].

Lemma 1. *Let $\mathcal{K}$ be a DL-Lite$^o_{\mathcal{A}}$ KB and let $\mathcal{K}_n$ be the normalized DL-Lite$^o_{\mathcal{A}}$ KB constructed as described above. Then $\mathcal{K}_n$ is satisfiable iff $\mathcal{K}$ is satifiable, and for each UCQ Q over $\mathcal{K}$, we have that $ans(Q, \mathcal{K}) = ansQ, \mathcal{K}_n$.*

By Lemma 1, we can from now on assume without loss of generality to deal only with *DL-Lite*$^o_{\mathcal{A}}$ KBs that are normalized. For a normalized KB, all nominals appear in the right-hand side of concept inclusion assertions that have the form as in Equation (1). In what follows, given a normalized *DL-Lite*$^o_{\mathcal{A}}$ TBox $\mathcal{T}$, we consider it partitioned into four parts denoted as follows: $\mathcal{T}_n$ contains the concept and role disjointness assertions; $\mathcal{T}_f$ contains the functionality assertions;

$\mathcal{T}_o$ contains the concept inclusion assertions involving nominals; $\mathcal{T}_p$ contains the remaining concept inclusions, and the role inclusions. We call $\mathcal{T}_p \cup \mathcal{T}_o$ the set of *positive inclusions* (PIs).

The concept inclusions in $\mathcal{T}_o$ require a specific treatment in the query answering approach based on rewriting, since they *restrict* the concept on the left-hand side of the concept inclusion to be interpreted as a singleton. This leads us to introduce the notion of *domain* and *range restricted roles*, i.e., roles whose domain or range is restricted to a nominal. For such a role P for which $\mathcal{T}_o$ contains $\exists P \sqsubseteq \{d\}$, for a model $\mathcal{I}$ of $\mathcal{T}$ we have that $P^\mathcal{I} \subseteq \{d^\mathcal{I}\} \times \Delta^\mathcal{I}$, making the interpretation of the role similar to that of a concept. If a concept is restricted to a nominal via $B \sqsubseteq \{d\}$, then every other concept that is restricted to the nominal indirectly, e.g., via $B' \sqsubseteq B$, should be restricted to the same nominal. Similarly, if a *functional* role is *domain* or *range restricted*, then the other component should have maximally one element, i.e., it is restricted to an unnamed nominal. To represent this fact we make use of underscore elements with subscripts $_0, _1, \ldots$. Hence, each of the occurances of these underscore elements denotes an *unnamed individual*, possibly coinciding with individuals representing nominals or with other unnamed individuals (i.e, for unnamed individuals the unique name assumption does not apply). We will use the underscore symbol without a subscript to denote an arbitrary unnamed element.

This motivates us to define the notion of *singleton closure* of a TBox.

Definition 1. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a normalized DL-Lite$^o_\mathcal{A}$ KB. Then the* singleton closure *$\mathcal{SC}(\mathcal{T})$ of $\mathcal{T}$, is obtained by including $\mathcal{T}$ in $\mathcal{SC}(\mathcal{T})$, and closing it according to the following rules:*

1. *If $B_1 \sqsubseteq \{d\}$ is in $\mathcal{SC}(\mathcal{T})$ and $B_2 \sqsubseteq B_1$ is in $\mathcal{T}_p$, then $B_2 \sqsubseteq \{d\}$ is in $\mathcal{SC}(\mathcal{T})$.*
2. *If $\exists P \sqsubseteq \{d\}$ is in $\mathcal{SC}(\mathcal{T})$ and (funct P) is in $\mathcal{T}$, then $\exists P^- \sqsubseteq \{_x\}$ is in $\mathcal{SC}(\mathcal{T})$, such that $_x$ is a new underscore element.*
3. *If $\exists P^- \sqsubseteq \{d\}$ and (funct P^-) is in $\mathcal{T}$, then $\exists P \sqsubseteq \{_x\}$ is in $\mathcal{SC}(\mathcal{T})$, such that $_x$ is a new underscore element.*

Lemma 2. *For every ABox A. The KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is satisfiable iff $\langle \mathcal{SC}(\mathcal{T}), A \rangle$ is satisfiable.*

In the following, we make use of these notions to devise query answering and reasoning techniques for *DL-Lite*$^o_\mathcal{A}$.

4 First-order Rewritablity of Query Answering

Our aim is to extend the query rewriting algorithm of [7,6] to handle also nominals. For this, we exploit the fact that *DL-Lite*$^o_\mathcal{A}$, similarly to other DLs of the *DL-Lite* family, enjoys a canonical model property. Specifically, we show how to adapt the notion of *restricted chase*, adopted in [9] for the case of inclusion dependencies in databases, and extended to *DL-Lite* in [7,8], to show the existence of a(n in general infinite) *canonical model* for any given *DL-Lite*$^o_\mathcal{A}$ KB. We

can construct the chase of a KB starting from the ABox, and applying positive inclusion assertions to the set of membership assertions obtained so far.

The critical difference in *DL-Lite*$^o_{\mathcal{A}}$ with respect to the chase introduced in [7] is that we have to take care of restricted roles (cf. Section 3), and hence make use of the singleton closure. As above, we use the underscore symbols to denote unnamed individuals. Throughout the construction of the canonical interpretation, some unnamed individuals will be named, and this will change the singleton closure. However, the cardinality of the singleton closure does not change. For the application of the chase rules it is assumed that we have a total (lexicographic) ordering on the assertions and on the constants (including the ones not occurring in our KB). The *chase* of $\mathcal{K}$ is the set of membership assertions $chase(\mathcal{K}) = \bigcup_{j\in\mathbb{N}} chase_i(\mathcal{K})$, starting from $chase_0(\mathcal{K}) = \mathcal{A}$, and $\mathcal{SC}_0$ is obtained from $\mathcal{SC}(\mathcal{T})$ by replacing the unnamed individuals in $\mathcal{SC}(\mathcal{T})$, which occur in the ABox with their names. For example, if $\exists R \sqsubseteq \{_{\text{-}x}\}$ and $R(a, b)$ is in the ABox, then $\exists R \sqsubseteq \{a\}$ will replace $\exists R \sqsubseteq \{_{\text{-}x}\}$ in $\mathcal{SC}_0$. Then, $chase_{i+1}(\mathcal{K})$ is obtained from $chase_i(\mathcal{K})$ by adding the membership assertion β_{new}, i.e., $chase_{i+1}(\mathcal{K}) = chase_i(\mathcal{K}) \cup \{\beta_{new}\}$, where β_{new} is the membership assertion obtained from $chase_i(\mathcal{K})$ by applying one of the chase rules. Similarly, $\mathcal{SC}_{i+1} = \mathcal{SC}_i[\sigma]$, the result of applying σ on $\mathcal{SC}_i$, where σ is the substitution resulting from the application of one of the chase rules. The substitution σ is the empty substitution by default, unless a concept restricted to an unnamed individual in the singleton closure has been mapped to a named individual. In that case, we will substitute the underscore with the named individual.

Table 1. Chase Rules for *DL-Lite*$^o_{\mathcal{A}}$

rule	α	β	$\notin chase_i(\mathcal{K})$	$\in \mathcal{SC}_i$	$\notin \mathcal{SC}_i$	β_{new}	σ
cr1	$A_1 \sqsubseteq A_2$	$A_1(a)$	$A_2(a)$			$A_2(a)$	[]
cr2	$A \sqsubseteq \exists R$	$A(a)$	$R(a, _{\text{-}x})$		$R^- \sqsubseteq \{_{\text{-}x}\}$	$R(a, a_{new})$	[]
cr2o	$A \sqsubseteq \exists R$	$A(a)$	$R(a, _{\text{-}x})$	$R^- \sqsubseteq \{d\}$		$R(a, d)$	[]
cr2u	$A \sqsubseteq \exists R$	$A(a)$	$R(a, _{\text{-}x})$	$R^- \sqsubseteq \{_{\text{-}x}\}$		$R(a, a_{new})$	$[R^- \sqsubseteq \{_{\text{-}x}\}/R^- \sqsubseteq \{a_{new}\}]$
cr3	$\exists R \sqsubseteq A$	$\exists R(a)$	$A(a)$			$A(a)$	[]
cr4	$\exists R_1 \sqsubseteq \exists R_2$	$\exists R_1(a)$	$R_2(a, _{\text{-}x})$		$R_2^- \sqsubseteq \{_{\text{-}x}\}$	$R_2(a, a_{new})$	[]
cr4o	$\exists R_1 \sqsubseteq \exists R_2$	$\exists R_1(a)$	$R_2(a, _{\text{-}x})$	$R_2^- \sqsubseteq \{d\}$		$R_2(a, d)$	[]
cr4u	$\exists R_1 \sqsubseteq \exists R_2$	$\exists R_1(a)$	$R_2(a, _{\text{-}x})$	$R_2^- \sqsubseteq \{_{\text{-}x}\}$		$R_2(a, a_{new})$	$[R_2^- \sqsubseteq \{_{\text{-}x}\}/R_2^- \sqsubseteq \{a_{new}\}]$
cr5	$R_1 \sqsubseteq R_2$	$R_1(a, b)$	$R_2(a, b)$			$R_2(a, b)$	[]

The chase rules are listed in Table 1. β is the first (in lexicographic order) membership assertion in $chase_i(\mathcal{K})$ such that there exists a PI (i.e., an inclusion in $\mathcal{T}_p \cup \mathcal{T}_o$) that is *applicable to it*, and α is the first such PI. The notion of *applicability* of a PI extends in a straightforward way the one in [7] (see below). The three subsequent columns in the table express the conditions under which the chase rule is applied in $chase_i(\mathcal{K})$, in terms of the atom that should be missing from $chase_i(\mathcal{K})$, and possibly the inclusion that should be present or absent from $\mathcal{SC}_i$.

Algorithm PerfectRef(q, $\mathcal{T}$)
Input: union of conjunctive queries Q, *DL-Lite*$^o_\mathcal{A}$ TBox $\mathcal{T}$
Output: union of conjunctive queries Q_r
$Q_r := Q$;
$\mathcal{ST} = \mathcal{SC}(\mathcal{T})$;
repeat (1)
  $Q'_r := Q_r$;
  for each $q \in Q'_r$
  (O) $q = reducesingleton(q, \mathcal{SC}(\mathcal{T}))$;
  (a) **for each** g in q
    for each PI I in $\mathcal{ST}$
      if I is applicable to g
      then $Q_r := Q_r \cup \{remdup(q[g/gr(g, I)])\}$;
  (b) **for each** g_1, g_2 in q
    if g_1 and g_2 unify
    then $Q_r := Q_r \cup \{remdup(\tau(reduce(q, g_1, g_2)))\}$;
until $Q'_r = Q_r$;
return Q_r.

Fig. 1. The algorithm PerfectRef

The *canonical interpretation can(*$\mathcal{K}$*)* $= \langle \Delta^{can(\mathcal{K})}, \cdot^{can(\mathcal{K})} \rangle$ is defined from *chase*($\mathcal{K}$) as follows: $\Delta^{can(\mathcal{K})}$ is the set of constants occurring in *chase*($\mathcal{K}$), $a^{can(\mathcal{K})} = a$ for each constant a occurring in *chase*($\mathcal{K}$), $A^{can(\mathcal{K})} = \{a \mid A(a) \in chase(\mathcal{K})\}$ for each atomic concept A, and $P^{can(\mathcal{K})} = \{(a_1, a_2) \mid P(a_1, a_2) \in chase(\mathcal{K})\}$ for each atomic role P. The construction of the canonical interpretation guarantees the satisfiability of the positive inclusion assertions in the TBox, and if $\langle \mathcal{SC}(\mathcal{T}), \mathcal{A} \rangle$ is satisfiable, then it also guarantees the satisfiability of the nominal inclusion assertions.

Lemma 3. *For every $i \geq 0$, if $chase_i(\mathcal{K})$ satisfies $\mathcal{SC}(\mathcal{T})$ then $chase_{i+1}(\mathcal{K})$ satisfies $\mathcal{SC}(\mathcal{T})$*

Lemma 4. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite$^o_\mathcal{A}$ KB. Then, can($\mathcal{K}$) is a model of $\mathcal{K}$ iff $\mathcal{K}$ is satisfiable.*

Now we can provide an adapted version of the PerfectRef algorithm presented in [7,6], which takes into account the presence of nominals in *DL-Lite*$^o_\mathcal{A}$. It takes a *DL-Lite*$^o_\mathcal{A}$ TBox $\mathcal{T}$ and a UCQ Q and returns a UCQ Q_r, which we will show to be the FO-rewriting of Q w.r.t. $\mathcal{T}$. In the following, after discussing some preliminary notions, we explain the algorithm PerfectRef, and show its termination and correctness.

Briefly, the algorithm PerfectRef, shown in Figure 1, iterates over the CQs in Q and takes into consideration those assertions in $\mathcal{T}$ that are relevant for the computation of the answers of Q in order to produce Q_r. Note that only the assertions in $\mathcal{T}_p$ and $\mathcal{T}_o$ play a role in the computation of Q_r. Roughly speaking, the algorithm checks the atoms, sees whether we can rewrite them into other atoms in each query in Q using the PIs as rewriting rules, reduces the *restricted*

roles in the queries using the inclusions in $\mathcal{T}_o$, and unifies any resulting unifiable atoms.

We recall here the basic features that our variant of PerfectRef has in common with the original version as presented in [7][2]. We say that a PI I is applicable to an atom g if g is unifiable with the right-hand side of the assertion. The result of the application of a PI $gr(g, I)$on an atom is the atom unifiable with the left-hand side of the assertion. For example, the PI $A \sqsubseteq \exists P$ is applicable to the atoms $P(x, \_)$ and $P(\_, \_)$, with the atoms resulting from this application being $A(x)$ and $A(\_)$, respectively. We use $q[g/g']$ to denote the CQ obtained by replacing every occurrence of the atom g in the query q by the atom g'. The function *remdup* removes from the body of a CQ atoms occurring more than once. We define the *most general unifier* (*mgu*) of two unifiable atoms g_1, g_2 occurring in a CQ q as in [7]. The function *reduce* computes the mgu of its two arguments g_1, g_2, and applies it to the CQ q. Finally, the function τ takes as input a CQ q and replaces every variable occurring only once with the symbol $\_$.

Our addition to PerfectRef is Step (O) which is used to handle assertions involving nominals. This step involves the reduction of restricted roles to concepts. This turns out to be necessary, since if R is a restricted role, i.e., $\exists R \sqsubseteq \{d\}$, then for every model $\mathcal{I}$ we will have that $R^{\mathcal{I}} \subseteq \{d^{\mathcal{I}}\} \times \Delta^{\mathcal{I}}$, which makes R "behave like a concept" w.r.t. rewriting steps. Hence, we can replace an atom $R(x, y)$ occurring in a CQ q by $\exists R^-(y)$, and replace any further occurrence of x in q with d. This is accomplished by the function *reducesingleton*, which makes use of the singleton closure $\mathcal{SC}(\mathcal{T})$.

Example 1. For example, if we have $q(x, y) \leftarrow R_1(x, y), R_2(x, y)$, and we have $R_2^- \sqsubseteq \{d\}$ in the TBox $\mathcal{T}$, then *reducesingleton* will reduce q to $q'(x, d) \leftarrow R_1(x, d), R_2(x, \_)$ (i.e., y is being bound to d in the answers of q). □

It is possible that one role is indirectly restricted to a singleton nominal, e.g., the assertions $A \sqsubseteq B$ and $B \sqsubseteq \{d\}$ induce the restriction $A \sqsubseteq \{d\}$. PerfectRef handles the effect of this interaction of positive inclusion assertions with nominal inclusion assertions by making use of $\mathcal{SC}(\mathcal{T})$, instead of $\mathcal{T}$.

Taking into consideration that the presence of nominals does not lead to an increase of the number of atoms in the rewritten queries, the termination of PerfectRef can be proved similarly to Lemma 34 in [7].

Lemma 5. *Let $\mathcal{T}$ be a DL-Lite$^o_{\mathcal{A}}$ TBox, and Q be a union of conjunctive queries over $\mathcal{T}$. Then, the algorithm* PerfectRef$(Q, \mathcal{T})$ *terminates.*

To compute the answers of Q, given the KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, we need to evaluate the set of conjunctive queries Q_r produced by the algorithm PerfectRef on the ABox $\mathcal{A}$, which can be done in AC^0 in data complexity.

Now we can start observing that query answering over satisfiable knowledge bases can in principle be done by evaluating the query over the model *can($\mathcal{K}$)*.

[2] Extensions of PerfectRef to deal with additional constructs have also been presented in [5,13,8].

Theorem 1. *Let $\mathcal{K}$ be a satisfiable DL-Lite$^o_\mathcal{A}$ KB, and let Q be a union of conjunctive queries over $\mathcal{K}$. Then, $ans(Q, \mathcal{K}) = Q^{can(\mathcal{K})}$*

The proof is done analogously to the proof of Theorem 29 in [7], which states an analogous result for *DL-Lite*$_\mathcal{R}$ and *DL-Lite*$_\mathcal{F}$.

Taking into consideration that *can(K)* is infinite in general, we cannot compute and evaluate queries over it. However, as for *DL-Lite*$_\mathcal{A}$, we can show that instead of computing $Q^{can(\mathcal{K})}$ we can evaluate the UCQ Q_r computed by PerfectRef directly over the ABox, considered as a database. For this purpose, we extend the definition of database interpretation given in [7] so as to handle the presence of nominals. The *database interpretation* $db(\mathcal{T}, \mathcal{A}) = \langle \Delta^{db(\mathcal{T}, \mathcal{A})}, \cdot^{db(\mathcal{T}, \mathcal{A})} \rangle$ of a KB $\langle \mathcal{T}, \mathcal{A} \rangle$ is the interpretation whose domain $\Delta^{db(\mathcal{T}, \mathcal{A})}$ is the non-empty set consisting of all constants appearing in $\mathcal{A}$ and in nominals in $\mathcal{T}$, and such that $a^{db(\mathcal{T}, \mathcal{A})} = a$ for each constant a, and $A^{db(\mathcal{T}, \mathcal{A})} = \{a \mid A(a) \in \mathcal{A}\}$ for each atomic concept A, and $P^{db(\mathcal{T}, \mathcal{A})} = \{(a_1, a_2) \mid P(a_1, a_2) \in \mathcal{A}\}$ for each atomic role P.

Theorem 2. *Let $\mathcal{T}$ be a DL-Lite$^o_\mathcal{A}$ TBox, Q a UCQ over $\mathcal{T}$, and $Q_r = \mathsf{PerfectRef}(Q, \mathcal{T})$ the UCQ returned by* PerfectRef*. Then, for every DL-Lite$^o_\mathcal{A}$ ABox $\mathcal{A}$ such that $\langle \mathcal{T}, \mathcal{A} \rangle$ is satisfiable, we have that $ans(Q, \langle \mathcal{T}, \mathcal{A} \rangle) = Q_r^{db(\mathcal{T}, \mathcal{A})}$.*

Corollary 1. *Query answering in DL-Lite$^o_\mathcal{A}$ is in AC^0 with respect to the data complexity*

5 FO-Rewritability of KB Satisfiability

We now consider KB satisfiablity in *DL-Lite*$^o_\mathcal{A}$, and provide a technique to solve it via query evaluation, hence showing its FO-rewritablity. The first case to be considered is a special case where the TBox has only positive inclusion assertions, in this case the knowledge base is just a *DL-Lite*$_\mathcal{A}$ knowledge base with only positive inclusion assertions which is satisfiable according to lemma 7 in [7].

However, in the general case, we cannot construct the canonical interpretation unless the nominal inclusion assertions are satisfied. According to theorem 3 it suffices to show that the *db(T, A)* satisfies the nominal inclusion assertions in the singleton closure in order to show that *can(K)* satisfies the singleton closure (and hence the nominal inclusion assertions in the TBox).

The ABox $\mathcal{A}$ satisfies $\mathcal{SC}_\mathbf{o}(\mathcal{T})$ when it satisfies all the assignments in $\mathcal{SC}_\mathbf{o}(\mathcal{T})$. An inclusion assertion $A \sqsubseteq \{x\}$ is satisfied when no individual, different than x, is asserted as a member to the concept A in $\mathcal{A}$, whereas the assertion: $A \sqsubseteq \{\neg x\}$ is satisfied when no two different individuals are asserted to the concept A in $\mathcal{A}$. In order to formulate the latter statement, we need to introduce the notion of a (boolean) query associated to a nominal inclusion assertion (which was originally applied only to negative inclusion assertions). This query evaluates false iff the assertion is satisfied. In the following, we make use of CQs and UCQs enriched with inequalities, and express them in the Datalog notation as in [8].

Definition 2. *Given a DL-Lite$^o_{\mathcal{A}}$ nominals inclusion assertion O in $\mathcal{SC}_{\mathbf{o}}(\mathcal{T})$, the form $B \sqsubseteq \{d\}$, such that B is a basic concept and d is an individual, the query associated to O is a boolean conjunctive query q_O of the form:*

$$q_O \leftarrow \exists x : B(x), x \neq d$$

Whereas, assertions of the form: $B \sqsubseteq \{\_\}$ are associated with the following query:

$$q_O \leftarrow \exists x \exists y : B(x), B(y) x \neq y$$

The following Lemma states that satisfiability for a *DL-Lite$^o_{\mathcal{A}}$* knowledge base $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, where $\mathcal{T}$ contains positive inclusions and nominal inclusions, can be reduced to answering a UCQ with inequalities over $\mathcal{A}$.

Lemma 6. *Let $\mathcal{K} = (\mathcal{T}_p \cup T_{\mathcal{O}}, \mathcal{A})$ be a DL-Lite$^o_{\mathcal{A}}$ KB, and $Q_{\mathcal{O}} = \bigvee_{\mathcal{O} \in \mathcal{SC}(T)} q_{\mathcal{O}}$. Then, $\mathcal{K}$ is satisfiable iff $\mathcal{A} \not\models Q_{\mathcal{O}}$.*

Having checked the satisfiabilty of the singleton closure, we can build the canonical interpretation satisfying a TBox containing only positive and nominal inclusion assertions. Now we have to consider negative inclusion assertions and functionality assertions. Functionality restrictions interact with the nominal restrictions, in the sense that if a role is restricted and functional then not only the restricted component but also the other component has to be taken into the singleton closure. We have already handled this situation in our definition of the singleton closure. Finally, we have to check whether the functionality restrictions are fullfilled. This we do by defining the query associated to a functionality restriction.

Definition 3. *Given a DL-Lite$^o_{\mathcal{A}}$ functionality restriction $\mathcal{F}$ of the form* (funct R) *in $\mathcal{T}_f$, the query associated to F is a boolean conjunctive query $q_{\mathcal{F}}$ of the form:*

$$q_{\mathcal{F}} \leftarrow R(x, y), R(x, z), y \neq z$$

Lemma 7. *Let $\mathcal{K} = (\mathcal{T}_p \cup T_{\mathcal{O}} \cup \mathcal{T}_f, \mathcal{A})$ be a DL-Lite$^o_{\mathcal{A}}$ KB, and $Q_{\mathcal{O}} = \bigvee_{\mathcal{O} \in \mathcal{SC}(T)} q_{\mathcal{O}}$ and $Q_{\mathcal{F}} = \bigvee_{\mathcal{F} \in \mathcal{T}_f} q_{\mathcal{F}}$. Then, $\mathcal{K}$ is satisfiable iff $\mathcal{A} \not\models Q_{\mathcal{O}}$ and $\mathcal{A} \not\models Q_{\mathcal{F}}$.*

Notably, since answering of $Q_{\mathcal{F}}$ and $Q_{\mathcal{O}}$ over the ABox $\mathcal{A}$ involves only evaluating $Q_{\mathcal{F}}$ and $Q_{\mathcal{O}}$ over *db($\mathcal{T}, \mathcal{A}$)*, the above lemma also entails that satisfiablity of *DL-Lite$^o_{\mathcal{A}}$* knowledge bases without NIs is FO-rewritable. Now, we can continue the tradition of [8] in using the algorithm PerfectRef, presented in Section 4, to check the satisfiablity of the negative inclusion assertions. Note that PerfectRef only requires the inclusions in $\mathcal{T}_p$ and $\mathcal{T}_o$.

Definition 4. *Given a DL-Lite$^o_{\mathcal{A}}$ negative inclusion assertion N of the form $B_1 \sqsubseteq \neg B_2$ in $\mathcal{T}_n$, the query associated to N is a boolean conjunctive query q_N of the form:*

$$q_N \leftarrow B_1(x), B_2(x)$$

Algorithm Consistent($\mathcal{K}$)
Input: *DL-Lite*$^o_\mathcal{A}$ $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$
Output: *true* if it is unsatisfiable, *false* otherwise

- **Normalize knowledge base**
  $\mathcal{K} = \mathsf{Normalize}(\mathcal{K})$
- **Build singleton closure**
  $\mathcal{SC}(\mathcal{T}) = \mathsf{SingletonClosure}(\mathcal{T})$
- **Check Singleton Closure and functionality restrictions**

$q_\mathcal{O} := \bot$;
foreach $\alpha \in \mathcal{SC}_\mathbf{o}(\mathcal{T}) \cup \mathcal{SC}_\mathcal{F}(\mathcal{T})$
$q_{\mathcal{SC}(\mathcal{T})} = q_{\mathcal{SC}(\mathcal{T})} \vee q_\alpha$;
if $q^{db(\mathcal{T},\mathcal{A})}_{\mathcal{SC}(\mathcal{T})} \neq \emptyset$ **return** *true*;
$q_{\mathcal{T}_n} := \bot$;
foreach $N \in \mathcal{T}_n$
$q_{\mathcal{T}_n} = q_{\mathcal{T}_n} \cup \{q_{\mathcal{T}_n}\}$
$q_{\mathcal{T}_n} = \mathsf{PerfectRef}(q_{\mathcal{T}_n}, \mathcal{T}_p \cup \mathcal{T}_\mathcal{O})$
return $q^{db(\mathcal{T},\mathcal{A})}_{\mathcal{T}_n}$

Fig. 2. The algorithm Consistent

Theorem 3. *Let $\langle \mathcal{T}, \mathcal{A} \rangle$ be a satisfiable DL-Lite$^o_\mathcal{A}$ KB, $\mathcal{T}_n$ a set of NIs, $Q_n = \bigcup_{N \in \mathcal{T}_n} q_N$ the UCQ associated to $\mathcal{T}_n$. Then, $\mathcal{K} = \langle \mathcal{T} \cup \mathcal{T}_n, \mathcal{A} \rangle$ is satisfiable iff $\langle \mathcal{T}, \mathcal{A} \rangle \not\models Q_n$.*

Now, given Lemma 6 and Theorem 3, we can establish the satisfiability of a *DL-Lite*$^o_\mathcal{A}$. This allows us to say that checking the satisfiability of *DL-Lite*$^o_\mathcal{A}$ KB is first-order rewritable, and we can now write an algorithm to check the satisfiabilty of KBs (see Figure 2). Briefly, the algorithm first normalizes the KB, then it checks whether the singleton closure is satisfied. Afterwards, it checks if the funcionality restrictions are satisfied, and finally considers the satisfiablity of the negative inclusion assertions. We will make use of the algorithms Normalize and SingletonClosure, where SingletonClosure is an algorithm for calculating $\mathcal{SC}(\mathcal{T})$. The next Theorem follows directly from the FO-rewritablity.

Theorem 4. *KB satisfiablity in DL-Lite*$^o_\mathcal{A}$ *is in AC^0 with respect to data complexity.*

6 Conclusion

This paper shows that the extending *DL-Lite*$_\mathcal{A}$ with singleton nominals does not have an impact on the data complexity of reasoning. The structure of the paper follows similar ideas to those in [7,8], but is based first define a singleton closure, and build a canonical interpretation based on this closure. Furthermore, we define a modified version of the algorithm PerfectRef to handle the nomianl constructions, which shows the FO-rewritability of query answering for satisfiable knowledge bases, and finally make use of this algorithms to check for the

satisfiability of $DL\text{-}Lite^{o}_{\mathcal{A}}$ knowledge bases using first-order query evaluation. Thus, showing the FO-rewritablity of the satisfiablity problem.

References

1. Artale, A., Calvanese, D., Kontchakov, R., Zakharyaschev, M.: The *DL-Lite* family and relations. J. of Artificial Intelligence Research 36, 1–69 (2009)
2. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In: Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005). pp. 364–369 (2005)
3. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press (2003)
4. Bao, J., et al.: OWL 2 Web Ontology Language document overview (second edition). W3C Recommendation, World Wide Web Consortium (Dec 2012), available at `http://www.w3.org/TR/owl2-overview/`
5. Botoeva, E., Artale, A., Calvanese, D.: Query rewriting in $DL\text{-}Lite^{\mathcal{HN}}_{horn}$. In: Proc. of the 23rd Int. Workshop on Description Logic (DL 2010). CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/`, vol. 573, pp. 267–278 (2010)
6. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Poggi, A., Rodríguez-Muro, M., Rosati, R.: Ontologies and databases: The *DL-Lite* approach. In: Tessaris, S., Franconi, E. (eds.) Reasoning Web. Semantic Technologies for Informations Systems – 5th Int. Summer School Tutorial Lectures (RW 2009), Lecture Notes in Computer Science, vol. 5689, pp. 255–356. Springer (2009)
7. 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. of Automated Reasoning 39(3), 385–429 (2007)
8. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Data complexity of query answering in description logics. Artificial Intelligence 195, 335–360 (2013)
9. Johnson, D.S., Klug, A.C.: Testing containment of conjunctive queries under functional and inclusion dependencies. J. of Computer and System Sciences 28(1), 167–189 (1984)
10. Lutz, C., Walther, D., Wolter, F.: Conservative extensions in expressive description logics. In: Proc. of the 20th Int. Joint Conf. on Artificial Intelligence (IJCAI 2007). pp. 453–458 (2007)
11. Motik, B., Cuenca Grau, B., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C.: OWL 2 Web Ontology Language profiles (second edition). W3C Recommendation, World Wide Web Consortium (Dec 2012), available at `http://www.w3.org/TR/owl2-profiles/`
12. Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Linking data to ontologies. J. on Data Semantics X, 133–173 (2008)
13. Savkovic, O.: Managing Data Types in Ontology-based Data Access. Master's thesis, KRDB Research Centre for Knowledge and Data, Free University of Bozen-Bolzano (2011)
14. Savkovic, O., Calvanese, D.: Introducing datatypes in *DL-Lite*. In: Proc. of the 20th Eur. Conf. on Artificial Intelligence (ECAI 2012) (2012)
15. Seylan, I., Franconi, E., de Bruijn, J.: Effective query rewriting with ontologies over DBoxes. In: Proc. of the 21st Int. Joint Conf. on Artificial Intelligence (IJCAI 2009). pp. 923–925 (2009)

16. Tobies, S.: The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. J. of Artificial Intelligence Research 12, 199–217 (2000)