Small Datalog Query Rewritings for $\mathcal{EL}$[⋆]

Giorgio Stefanoni, Boris Motik, and Ian Horrocks

Department of Computer Science, University of Oxford

1 Introduction

Description Logics are a key technology in data management scenarios such as Ontology-Based Data Access (OBDA), a paradigm in which a DL ontology is used to provide a conceptual view of the data [1]. An OBDA system transforms a conjunctive query over the ontology into a query over the data sources [2]. This transformation is independent of the data, so the OBDA approach can thus be used in settings where the data sources provide read-only access to the data, and where the data changes frequently.

Most existing OBDA systems are based on the DL-Lite family of *lightweight* Description Logics [1], which is also the basis for the QL profile of the OWL 2 ontology language. Logics in this family have been designed to allow a conjunctive query posed over the ontology to be rewritten as a first order query over the data sources—that is, queries are first-order rewritable. The query rewriting procedure is independent of the data, and the resulting queries can be evaluated using highly scalable relational database technology. To achieve this, however, the expressive power of DL-Lite is very restrictive. This prevents the OBDA approach from being applied in the life science domain, where many ontologies use DLs from the $\mathcal{EL}$ family [3, 4]. This family provides the basis for the EL profile of OWL2, and many prominent ontologies, such as SNOMED-CT, were developed using this language.

The problem of answering conjunctive queries in $\mathcal{EL}$ has already been studied in the literature, and two orthogonal approaches have been proposed. First, Rosati proposed a pure query rewriting technique which transforms an $\mathcal{EL}$ TBox $\mathcal{T}$ and a conjunctive query q into a Datalog program $P_{\mathcal{T},q}$ [5]. Second, Lutz et al. introduced a "combined" approach [6, 7]. This technique first materializes certain facts entailed by the ontology in a precomputation step. Then, each user query is rewritten into a polynomial first-order query that, when evaluated over the materialized facts, computes the answers to the user's query.

Unfortunately, these two approaches exhibit several shortcomings when applied in the context of OBDA. In particular, Rosati's rewriting technique computes for each user query a fresh Datalog program whose size depends on both the query and the terminology, which could be very inefficient when dealing with large scale ontologies. The approach by Lutz et al. produces smaller first order rewritings, but the use of materialization means that the technique is only applicable when the data sources provide read/write access to the data; furthermore, materialization can be inefficient if the data changes frequently.

⋆ This work was supported by EPSRC and Alcatel-Lucent.

In this paper, we present a pure query rewriting technique to query answering in $\mathcal{EL}$. Our approach reinterprets the combined approach proposed by Lutz and colleagues in terms of Datalog. Our rewriting procedure consists of two distinct steps. The first step rewrites a TBox $\mathcal{T}$ into a Datalog program $P_{\mathcal{T}}$, whose size depends linearly on the size of $\mathcal{T}$. Then, at query time, the conjunctive query q is rewritten into a Datalog query $\langle Q_P, Q_C \rangle$, whose size depends polynomially on q. The two rewriting steps are such that, given an ABox $\mathcal{A}$, deciding whether $Q_P(a_1, \ldots, a_k)$ follows from $P_{\mathcal{T}} \cup Q_C \cup \mathcal{A}$ is equivalent to deciding whether $\langle a_1, \ldots, a_k \rangle$ is a certain answer to q over a knowledge base $\langle \mathcal{T}, \mathcal{A} \rangle$.

At last, we summarize our main contributions. First, our rewriting approach, unlike Rosati's, separates the rewriting of the TBox and the query into two distinct steps, thus reducing inefficiency when dealing with large ontologies. Second, our technique does not require the materialization of entailed facts, hence our solution is in the spirit of OBDA and it avoids the problems associated with the materialization of large models. Finally, we set the stage for assessing the utility and the applicability to $P_{\mathcal{T}} \cup Q_C \cup \mathcal{A}$ of optimized Datalog evaluation techniques, such as *magic sets* and *SLG resolution* [8, 9]. Indeed, heuristic-based evaluation strategies significantly reduce the number of facts needed to answer a query, thus potentially improving the performance of our rewriting approach. In this paper we provide only proof sketches, and refer the reader to [10] for full proofs.

2 Preliminaries

Description Logic $\mathcal{EL}$

Let N_C, N_R, N_I be pairwise disjoint infinite sets of atomic concepts, atomic roles, and individuals. Together, the sets N_C, N_R, and N_I form the *signature* of an $\mathcal{EL}$ language. Whenever the distinction between atomic concepts and atomic roles is immaterial, we call an element of $N_C \cup N_R$ a *predicate*. The set of $\mathcal{EL}$ *concept expressions* is inductively defined starting from atomic concepts $A \in N_C$ and atomic roles $R \in N_R$ according to the syntax rule: $C \rightarrow A \mid C_1 \sqcap C_2 \mid \exists R.C \mid \top$.

An $\mathcal{EL}$ TBox $\mathcal{T}$ is a finite set of *concept inclusions* of the form $C \sqsubseteq D$; an $\mathcal{EL}$ ABox $\mathcal{A}$ is a finite set of *assertions* of the form $A(a)$ or $R(a, b)$ with a and b individuals; and an $\mathcal{EL}$ *knowledge base* (KB) is a tuple $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, where $\mathcal{T}$ is an $\mathcal{EL}$ TBox and $\mathcal{A}$ is an $\mathcal{EL}$ ABox. We denote with $\mathsf{Ind}(\mathcal{A})$ the set of all individuals occurring in the ABox $\mathcal{A}$. Furthermore, for $\mathcal{E}$ either a TBox or an ABox, $\mathsf{Pred}(\mathcal{E})$ is the set of all predicates occurring in $\mathcal{E}$.

Semantics is given as usual in terms of first-order interpretations $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$, where $\Delta^{\mathcal{I}}$ is a nonempty *domain* and $\cdot^{\mathcal{I}}$ is an *interpretation function*; please refer to [3] for details. In the following, we will extensively use the notion of an *unraveling of an interpretation w.r.t. an ABox.* Consider an interpretation $\mathcal{I}$ and an ABox $\mathcal{A}$ over an arbitrary $\mathcal{EL}$ signature. A *path* p in $\mathcal{I}$ w.r.t. $\mathcal{A}$ is a nonempty finite sequence $c_1 \cdot R_2 \cdot c_2 \cdots R_{n-1} \cdot c_{n-1} \cdot R_n \cdot c_n$ such that $c_1 \in \{a^{\mathcal{I}} \mid a \in \mathsf{Ind}(\mathcal{A})\}$ and for all $1 \leq i \leq n-1$ we have that $\langle c_i, c_{i+1} \rangle \in R_{i+1}^{\mathcal{I}}$ for $R_{i+1} \in N_R$. We say that a path p has *depth* n and we write $\mathsf{dep}(p) = n$; furthermore, $\mathsf{tail}(p)$ is the last domain element c_n in p. Let $\mathsf{paths}_{\mathcal{A}}(\mathcal{I})$ denote the set of all paths w.r.t. $\mathcal{A}$ occurring in $\mathcal{I}$. The unraveling $\mathcal{J}$ of $\mathcal{I}$ w.r.t. $\mathcal{A}$ is the following interpretation.

$$\begin{aligned}
\Delta^{\mathcal{J}} &= \mathsf{paths}_{\mathcal{A}}(\mathcal{I}) \\
a^{\mathcal{J}} &= a^{\mathcal{I}} \\
A^{\mathcal{J}} &= \{p \in \mathsf{paths}_{\mathcal{A}}(\mathcal{I}) \mid \mathsf{tail}(p) \in A^{\mathcal{I}}\} \\
R^{\mathcal{J}} &= \{\langle a^{\mathcal{J}}, b^{\mathcal{J}}\rangle \mid R(a,b) \in \mathcal{A}\} \cup \{\langle p,\ p \cdot R \cdot c\rangle \mid \{p,\ p \cdot R \cdot c\} \subseteq \mathsf{paths}_{\mathcal{A}}(\mathcal{I})\}
\end{aligned}$$

In this paper, we deal only with normalized $\mathcal{EL}$ TBoxes. Let A_1, A, and B be arbitrary concepts from $N_C \cup \{\top\}$. We say that an $\mathcal{EL}$ TBox $\mathcal{T}$ is in *normal form* if each axiom in $\mathcal{T}$ is in one of the following forms: $A \sqsubseteq B, A \sqcap A_1 \sqsubseteq B$, $A \sqsubseteq \exists R.B$, or $\exists R.A \sqsubseteq B$. Given an arbitrary $\mathcal{EL}$ TBox $\mathcal{T}$, we can compute a normalized TBox $\mathcal{T}_{norm}$ of $\mathcal{T}$ in linear time [3].

Querying $\mathcal{EL}$ KBs

Let N_V be an infinite set of *variables* disjoint from N_I. Together, N_V and N_I form the set N_T of *terms*. A *first-order query* q is a first-order formula constructed from the terms in N_T and the predicates from $N_C \cup N_R$ [8]. In general, we write $q = \psi(\vec{x})$ to express that q is the FO formula ψ whose *answer variables* are $\vec{x} = \{x_1, \ldots, x_k\}$. A query with k answer variables is a *k-ary query.* A *conjunctive query* (CQ) is a FO query of the form $q = \exists \vec{y}.\psi(\vec{x}, \vec{y})$, where ψ is a conjunction of unary atoms $A(s)$ and binary atoms $R(s,t)$ with s and t terms. The variables $\vec{y}$ are the *quantified variables* of q. In the following, $\mathsf{avar}(q)$ is the set of answer variables of q, and $\mathsf{qvar}(q)$ is the set of quantified variables. Finally, $N_V(q)$ is the set of all variables occurring in q, and $N_T(q)$ is the set of all terms occurring in q. Let $q = \psi(\vec{x})$ be a k-ary FO query with $\vec{x} = \langle x_1, \ldots, x_k\rangle$ and let $\mathcal{I}$ be an interpretation. We say that a k-ary tuple of individuals $\langle a_1, \ldots, a_k\rangle$ is an answer to q in $\mathcal{I}$, written $\mathcal{I} \models q[a_1, \ldots, a_k]$, if $\mathcal{I}$ satisfies q under the mapping π which sets $\pi(x_i) = a_i$ for all $1 \leq i \leq k$. We call π a *match* for q in $\mathcal{I}$ witnessing $\langle a_1, \ldots, a_k\rangle$, written $\mathcal{I} \models^{\pi} q$. We say that $\langle a_1, \ldots, a_k\rangle$ is a *certain answer* to q over $\mathcal{K}$ if $\mathcal{I} \models q[a_1, \ldots, a_k]$, for all models $\mathcal{I}$ of $\mathcal{K}$. We denote the set of all certain answers to q over $\mathcal{K}$ with $\mathsf{cert}(q, \mathcal{K})$. Rosati in [5] showed that deciding whether a tuple of individuals is a certain answer to q over $\mathcal{K}$ is PTIME-complete w.r.t. data-complexity (i.e., w.r.t. the size of the ABox); PTIME-complete w.r.t. KB complexity (i.e., w.r.t. the size of $\mathcal{K}$); and, NP-complete w.r.t. combined complexity (i.e., w.r.t. the size of both $\mathcal{K}$ and q).

Datalog

Let N_B be a nonempty set of *built-in predicates* [11]. Then, a DATALOG *rule* r is an expression of the form

$$S(\vec{u}) \leftarrow S_1(\vec{u}_1), \ldots, S_n(\vec{u}_n), B_{n+1}(\vec{u}_{n+1}) \ldots, B_m(\vec{u}_m),$$

where $n, m \geq 0$, $\{S, S_1, \ldots, S_n\} \subseteq N_C \cup N_R$, $\{B_{n+1}, \ldots, B_m\} \subseteq N_B$, and $\vec{u}$ and $\vec{u}_i$ are tuples of terms of suitable length. A rule is *safe* if each variable occurring in $\vec{u} \cup \vec{u}_{n+1} \cup \ldots \cup \vec{u}_m$ also occurs in $\vec{u}_1 \cup \ldots \cup \vec{u}_n$. Atom $S(\vec{u})$ is the *head* of the rule, and atoms $S_1(\vec{u}_1), \ldots, B_m(\vec{u}_m)$ constitute the *body* of the rule. Whenever the body of a rule r is empty, we call r a *fact*, and we write the rule as $S(\vec{u})$. A Datalog *program* P is a set of safe Datalog rules. Finally, $sch(P)$ is the set of predicates occurring in P.

Next, we define the semantics of a Datalog program P using *Herbrand interpretations* [8]. The Herbrand Universe of P is the set of all individuals occurring in P. The Herbrand Base of P is the set of all facts that can be constructed from the predicates in $N_C \cup N_R$ and the individuals in the universe of P. A Herbrand interpretation I of P is a subset of the Herbrand Base of P. Note that I does not interpret built-in predicates. As usual, we assume that these predicates are evaluated over a fixed, possibly infinite Herbrand interpretation B [12]. Then I is a model of P w.r.t. B if, for all the rules r in P, we have that

$$I \cup \mathsf{B} \models \forall \vec{x}(B_m(\vec{u}_n) \wedge \ldots \wedge B_{n+1}(\vec{u}_{n+1}) \wedge S_n(\vec{u}_n) \wedge \ldots \wedge S_1(\vec{u}_1) \rightarrow S(\vec{u})),$$

where $\vec{x}$ is a tuple consisting of all variables occurring in the rule. The semantics of a Datalog program P is defined as the minimal Herbrand interpretation I satisfying P w.r.t. B, written $\mathsf{M}_{\mathsf{B}}(P)$. Whenever the program does not contain built-in predicates, we do not consider the interpretation B and we simply write $\mathsf{M}(P)$. The semantics of Datalog programs can be defined also by means of a fixpoint construction. Then, T_P is the *immediate consequence operator* that maps instances $\mathbf{I}$ over $sch(P)$ to instances over $sch(P)$ as follows. For each rule r in P, if there exists a match π for $S_1(\vec{u}_1) \wedge \ldots \wedge S_n(\vec{u}_n) \wedge B_{n+1}(\vec{u}_{n+1}) \wedge \ldots \wedge B_m(\vec{u}_m)$ in $\mathbf{I} \cup \mathsf{B}$, then $S(a_1, \ldots, a_k)$ is contained in $T_P(\mathbf{I})$ with $a_i = \pi(u_i)$ for each $u_i \in \vec{u}$. One can prove that T_P has a minimum fixpoint T_P^ω such that $T_P^\omega = \mathsf{M}_{\mathsf{B}}(P)$ [8].

Finally, a Datalog query is a tuple $\langle Q_P, Q_C \rangle$ where Q_P is a predicate symbol and Q_C is a Datalog program. A tuple of individuals $\langle a_1, \ldots, a_k \rangle$ is an *answer* to $\langle Q_P, Q_C \rangle$ over Datalog program P if $P \cup Q_C \models Q_P(a_1, \ldots, a_k)$.

3 Datalog Rewriting for $\mathcal{EL}$ TBoxes

In this section, we show how to transform an $\mathcal{EL}$ TBox $\mathcal{T}$ into a Datalog program $P_\mathcal{T}$ whose size depends linearly on $\mathcal{T}$. The transformation is such that, for an arbitrary $\mathcal{EL}$ ABox $\mathcal{A}$, we can use the unraveling of $\mathsf{M}(P_\mathcal{T} \cup \mathcal{A})$ to compute the answers to conjunctive queries over $\langle \mathcal{T}, \mathcal{A} \rangle$. Let $\mathcal{T}$ be a TBox over an arbitrary $\mathcal{EL}$ signature. Intuitively, for each axiom α occurring in $\mathcal{T}$, the program $P_\mathcal{T}$ contains a set of Datalog rules which encode the constraint imposed by α. To achieve this, we have to overcome two issues.

First, $\mathcal{EL}$ concept inclusions of the form $A \sqsubseteq \exists R.B$ require the use of either existential quantifications or Skolem terms in rule heads; however, Datalog does not allow neither of the two. In order to solve this issue, we use a technique that has been introduced for representing *canonical models* of $\mathcal{EL}$ knowledge bases [3]. That is, for each atomic concept B occurring in $\mathcal{T}$ we introduce a fresh

Axiom α		**Set of rules** $\Theta(\alpha)$
$A \sqsubseteq B$	$\rightsquigarrow$	$B(X) \leftarrow A(X)$
$A_1 \sqcap A_2 \sqsubseteq B$	$\rightsquigarrow$	$B(X) \leftarrow A_1(X), A_2(X)$
$\exists R.A \sqsubseteq B$	$\rightsquigarrow$	$B(X) \leftarrow R(X,Y), A(Y)$
$A \sqsubseteq \exists R.B$	$\rightsquigarrow$	$R(X, o_B) \leftarrow A(X)$
		$B(o_B) \leftarrow A(X)$

Fig. 1. Transformation of $\mathcal{EL}$ Axioms into Rules.

auxiliary individual o_B, which represents the class of existentially quantified individuals of type B. Then, for each axiom of the above form, the program $P_\mathcal{T}$ contains the following two rules:

$$R(X, o_B) \leftarrow A(X); \qquad B(o_B) \leftarrow A(X).$$

Second, $\mathcal{EL}$ allows for $\top$ to occur in concept expressions. Hence, we need to define in $P_\mathcal{T}$ a unary predicate $\top$, whose extension—given an ABox $\mathcal{A}$—coincides with the Herbrand universe of $P_\mathcal{T} \cup \mathcal{A}$. To achieve this, we restrict our study to a subset of all $\mathcal{EL}$ ABoxes. In particular, we consider only those ABoxes $\mathcal{A}$ such that $\mathsf{Pred}(\mathcal{A}) \subseteq \mathsf{Pred}(\mathcal{T})$. That is, each predicate occurring in the ABox $\mathcal{A}$ must occur also in the TBox $\mathcal{T}$. Then, in our Datalog program, for each atomic concept A and for each atomic role R occurring in $\mathcal{T}$, we add the following rules:

$$\top(X) \leftarrow A(X); \qquad \top(X) \leftarrow R(X,Y); \qquad \top(Y) \leftarrow R(X,Y).$$

This is only one of the several ways in which we can encode such a predicate. In fact, another possibility would be—as suggested by Rosati in [5]—to assume that each ABox $\mathcal{A}$ contains an assertion $\top(a)$ for each individual $a \in \mathsf{Ind}(\mathcal{A})$. We believe that in the context of OBDA—where the focus is to provide access to arbitrary data-sources—it is important to make as few assumptions as possible on the physical realization of the ABox. For this reason, we prefer the option presented above.

Next, we formalize the transformation of a TBox $\mathcal{T}$ into a Datalog program $P_\mathcal{T}$. Let $\mathbf{Aux} = \{o_A \mid A \in N_C\} \cup \{o_\top\}$ be a set of *auxiliary individuals* distinct from N_I. Then, the program $P_\mathcal{T}$ is constructed from terms in $N_T \cup \mathbf{Aux}$ and predicates in $N_C \cup N_R \cup \{\top\}$ as follows. The transformation uses the function Θ, shown in Figure 1, to transform each axiom in the (normalized) TBox $\mathcal{T}$ into a set of Datalog rules. The Datalog program $P_\mathcal{T}$ is then defined as follows.

$$\begin{aligned} P_\mathcal{T} = & \textstyle\bigcup_{\alpha \in \mathcal{T}} && \Theta(\alpha) \\ & \textstyle\bigcup_{A \in \mathsf{Pred}(\mathcal{T}) \cap N_C} && \top(X) \leftarrow A(X) \\ & \textstyle\bigcup_{R \in \mathsf{Pred}(\mathcal{T}) \cap N_R} && \top(X) \leftarrow R(X,Y),\ \top(Y) \leftarrow R(X,Y) \end{aligned}$$

The following result readily follows from the definition of the program.

Proposition 1. *For an arbitrary $\mathcal{EL}$ TBox $\mathcal{T}$,* Datalog *program $P_\mathcal{T}$ can be computed in time linear in the size of $\mathcal{T}$.*

Consider an arbitrary $\mathcal{EL}$ ABox $\mathcal{A}$. Next, we prove that the unraveling $\mathcal{U}$ w.r.t. $\mathcal{A}$ of $\mathsf{M}(P_{\mathcal{T}} \cup \mathcal{A})$ can be used to answer conjunctive queries over $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$. We do so in two distinct steps. First, we introduce the notion of chase of an $\mathcal{EL}$ knowledge base $\mathcal{K}$. Second, we show that the chase of $\mathcal{K}$ is isomorphic to $\mathcal{U}$.

The *chase* of an $\mathcal{EL}$ knowledge base $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, written $\mathsf{chase}(\mathcal{K})$, is a possibly infinite Herbrand interpretation defined inductively by starting from $\mathcal{A}$ and then *applying* axioms occurring in the TBox to assertions occurring in the ABox. In our definition of the chase, we use function terms to denote existentially quantified individuals. Hence, the definition of ABox assertion is extended in a natural way to accommodate for assertions over function terms. We denote with u and w terms that can be either individuals or function terms. Next, we define an operator $\Gamma_{\mathcal{T}}$ that chases an ABox by applying the axioms occurring in the TBox $\mathcal{T}$. In the definition, we use assertions of the form $\top(u)$ to assert that u is a member of the $\mathcal{EL}$ *concept expression* $\top$. For $\mathcal{S}$ an arbitrary ABox, $\Gamma_{\mathcal{T}}(\mathcal{S})$ is the smallest ABox containing $\mathcal{S}$ and closed under the following *chasing rules.*

(cr1) If $\{A(u)\} \subseteq \mathcal{S}$ and $A \sqsubseteq B \in \mathcal{T}$, then $\{B(u)\} \subseteq \Gamma_{\mathcal{T}}(\mathcal{S})$.
(cr2) If $\{A_1(u), A_2(u)\} \subseteq \mathcal{S}$ and $A_1 \sqcap A_2 \sqsubseteq B \in \mathcal{T}$, then $\{B(u)\} \subseteq \Gamma_{\mathcal{T}}(\mathcal{S})$.
(cr3) If $\{R(u,w), A(w)\} \subseteq \mathcal{S}$ and $\exists R.A \sqsubseteq B \in \mathcal{T}$, then $\{B(u)\} \subseteq \Gamma_{\mathcal{T}}(\mathcal{S})$.
(cr4) If $\{A(u)\} \subseteq \mathcal{S}$ and $A \sqsubseteq \exists R.B \in \mathcal{T}$, then

$$\{R(u, f(u,R,B)), B(f(u,R,B))\} \subseteq \Gamma_{\mathcal{T}}(\mathcal{S}).$$

(cr5) If u occurs in $\mathcal{S}$, then $\{\top(u)\} \subseteq \Gamma_{\mathcal{T}}(\mathcal{S})$.

We now define an infinite sequence of finite ABoxes $\mathcal{A}_i$ for $i \in \mathbb{N}$.

$$\begin{aligned} \mathcal{A}_0 &= \mathcal{A} \\ \mathcal{A}_{i+1} &= \Gamma_{\mathcal{T}}(\mathcal{A}_i) \end{aligned}$$

Finally, the chase of $\mathcal{K}$ is the infinite union of all such ABoxes $\mathcal{A}_i$.

$$\mathsf{chase}(\mathcal{K}) = \bigcup_{i \in \mathbb{N}} \mathcal{A}_i$$

It is clear that our construction of the chase of $\mathcal{K}$ is *fair*. In fact, for each $i \in \mathbb{N}$ we have that $\mathcal{A}_{i+1}$ is the result of exhaustively applying—to all possible assertions occurring in $\mathcal{A}_i$—all applicable axioms in $\mathcal{T}$. At last, we want to point out that $\mathsf{chase}(\mathcal{K})$ can be used to compute the certain answers to a CQ q over $\mathcal{K}$.

Proposition 2 ([5]). *Let $\mathcal{K}$ be an $\mathcal{EL}$ knowledge base. Further, let q be a k-ary conjunctive query. Then, for each k-ary tuple of individuals $\langle a_1, \ldots, a_k \rangle$, we have*

$$\langle a_1, \ldots, a_k \rangle \in \mathit{cert}(q, \mathcal{K}) \text{ if and only if } \mathsf{chase}(\mathcal{K}) \models q[a_1, \ldots, a_k].$$

So, by proving that the unraveling $\mathcal{U}$ of $\mathsf{M}(P_{\mathcal{T}} \cup \mathcal{A})$ is isomorphic to $\mathsf{chase}(\mathcal{K})$, we establish that $\mathcal{U}$ can be used to answer conjunctive queries over $\mathcal{K}$. To prove the structural equivalence of $\mathcal{U}$ and $\mathsf{chase}(\mathcal{K})$, we define a function h mapping paths occurring in $\mathcal{U}$ to terms in $\mathsf{chase}(\mathcal{K})$. We define h by induction on the depth of paths occurring in $\mathcal{U}$ as follows.

Base Case. Consider an arbitrary $p \in \Delta^{\mathcal{U}}$ with $\mathsf{dep}(p) = 1$. We set $h(p) := p$.

Inductive Step. Let $p = t_1 \cdot R_2 \cdot t_2 \cdots t_{n-1} \cdot R_n \cdot t_n$ be a path occurring in $\mathcal{U}$ such that $h(p)$ has not been defined yet, but $h(t_1 \cdots R_{n-1} \cdot t_{n-1}) = u$. We distinguish between two cases depending on the type of the individual t_n.

1. If t_n occurs in the ABox, we set $h(p) := t_n$.
2. If t_n is of the form o_B, we set $h(p) := f(u, R_n, B)$.

Theorem 1 shows that h is an isomorphism between the two structures. Intuitively, for the only-if direction, we show that h is an injective homomorphism from $\mathcal{U}$ to $\mathsf{chase}(\mathcal{K})$ by induction on the depth of paths occurring in $\mathcal{U}$; for the if-direction, by induction on the construction of $\mathsf{chase}(\mathcal{K})$ we prove that h is a surjective function and that it is a homomorphism from $\mathsf{chase}(\mathcal{K})$ to $\mathcal{U}$.

Theorem 1. *Function h is an isomorphism from $\mathcal{U}$ to $\mathsf{chase}(\mathcal{K})$.*

Since the unraveling of $\mathsf{M}(P_{\mathcal{T}} \cup \mathcal{A})$ is generally infinite, this result alone does not provide us with an algorithm for answering queries in $\mathcal{EL}$. In the next section, we show how to rewrite a user query q into a Datalog query $\langle Q_P, Q_C \rangle$ such that $P_{\mathcal{T}} \cup \mathcal{A} \cup Q_C \models Q_P(a_1, \ldots, a_k)$ if and only if $\langle a_1, \ldots, a_k \rangle \in \mathsf{cert}(q, \mathcal{K})$ and thus solve the problem.

4 Polynomial Query Rewriting in Datalog

In the previous section, we have seen that for an arbitrary $\mathcal{EL}$ KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ evaluating a conjunctive query q over the unraveling of the Herbrand model of $P_{\mathcal{T}} \cup \mathcal{A}$ is equivalent to computing the certain answers to q over $\mathcal{K}$. In this section, we develop a query rewriting procedure that reduces the computation of $\mathsf{cert}(q, \mathcal{K})$ to the problem of evaluating a suitably constructed Datalog query over $P_{\mathcal{T}} \cup \mathcal{A}$. We achieve this in two steps. First, we present an interesting property of a certain class of interpretations. Second, we show how this result can be used to develop a query rewriting procedure in our Datalog setting.

We use the notions of $\mathcal{A}$-connected and split interpretations from [6, 13]. Let $\mathcal{I}$ be an interpretation and let $\mathcal{A}$ be an ABox over an arbitrary $\mathcal{EL}$ signature. We say that $\mathcal{I}$ is *$\mathcal{A}$-connected* if, for each domain element $c \in \Delta^{\mathcal{I}}$, there exists a path $p \in \mathsf{paths}_{\mathcal{A}}(\mathcal{I})$ such that $\mathsf{tail}(p) = c$. Furthermore, $\mathcal{I}$ is a *split* interpretation if, for all domain elements $c, c' \in \Delta^{\mathcal{I}}$, we have that $c \notin \{a^{\mathcal{I}} \mid a \in \mathsf{Ind}(\mathcal{A})\}$ and $\langle c, c' \rangle \in R^{\mathcal{I}}$ imply $c' \notin \{a^{\mathcal{I}} \mid a \in \mathsf{Ind}(\mathcal{A})\}$. Intuitively, in an $\mathcal{A}$-connected interpretation $\mathcal{I}$, for each domain element c_n it is always possible to find a path $a^{\mathcal{I}} \cdot R_2 \cdot c_2 \cdots R_n \cdot c_n$ such that $a \in \mathsf{Ind}(\mathcal{A})$. Furthermore, if $\mathcal{I}$ is split, then each domain element that is not the image of an individual can be related by a role only with elements that themselves do not interpret individuals.

Then, let $\mathcal{J}$ be the unraveling w.r.t. $\mathcal{A}$ of a split and $\mathcal{A}$-connected interpretation $\mathcal{I}$ and let q be a conjunctive query. Lutz et al. in [6, 13] showed that it is possible to reduce the problem of answering q in $\mathcal{J}$ to evaluating a first-order query rewriting q^* of q over $\mathcal{I}$. Roughly speaking, the query rewriting q^* rules out some spurious answers for q in $\mathcal{I}$ that cannot be reproduced in $\mathcal{J}$. More specifically, we have to ensure that the answer variables of q, the variables of q mapped to cyclic portions of $\mathcal{I}$, and the variables of q mapped to nontree portions of $\mathcal{I}$ are all matched only to the domain elements in $\{a^{\mathcal{I}} \mid a \in \mathsf{Ind}(\mathcal{A})\}$.

We now briefly outline how we can construct such an FO rewriting q^* for q [6]. Let $\sim_q$ be the smallest equivalence relation over $N_T(q)$ that is closed under the following rule: if $R(s,t)$ and $R(s',t')$ occur in q and $t \sim_q t'$, then $s \sim_q s'$. Then, for each equivalence class ζ of $\sim_q$, we let $t_\zeta \in \zeta$ be an arbitrary, but fixed, representative of the class. Also, for each such equivalence class ζ and for each atomic role R occurring in q, we let $\mathsf{Pred}(\zeta, R)$ be the following set.

$$\mathsf{Pred}(\zeta, R) = \{t \in N_T(q) \mid R(t,t') \text{ occurs in } q \text{ with } t' \in \zeta\}$$

Next, we define three sets of terms that correspond to the above mentioned cases.
– $\mathsf{Fork}_=$ is the set of all pairs $\langle \mathsf{Pred}(\zeta, R), t_\zeta \rangle$ such that ζ is an equivalence class of $\sim_q$ and $|\mathsf{Pred}(\zeta, R)| > 1$.
– $\mathsf{Fork}_{\neq}$ is the set of all quantified variables $v \in \mathsf{qvar}(q)$ for which atoms $R(s,v)$ and $S(s',t)$ exist in q such that $R \neq S$ and $v \sim_q t$.
– Cyc is the set of all variables $v \in \mathsf{qvar}(q)$ for which atoms

$$R_0(t_0, t'_0), \ldots, R_m(t_m, t'_m), \ldots, R_n(t_n, t'_n)$$

exist in q such that $m, n \geq 0$; for some $i \leq n$ we have that $t_i \sim_q v$; for each $j < n$ we have that $t'_j \sim_q t_{j+1}$; and $t'_n \sim_q t_m$.

We are now ready to formally specify the FO query rewriting q^*. In the definition, we assume that Aux is a fresh predicate not occurring in q and $\mathcal{K}$ and that every interpretation $\mathcal{I}$ interprets Aux as $\Delta^{\mathcal{I}} \setminus \{a^{\mathcal{I}} \mid a \in \mathsf{Ind}(\mathcal{A})\}$. Then, formulae q_1 and q_2 are defined as follows.

$$q_1 = \bigwedge_{v \in \mathsf{avar}(q) \cup \mathsf{Fork}_{\neq} \cup \mathsf{Cyc}} \neg \mathsf{Aux}(v)$$

$$q_2 = \bigwedge_{\langle \mathsf{Pred}(\zeta,R), t_\zeta \rangle \in \mathsf{Fork}_=} \neg \mathsf{Aux}(t_\zeta) \vee \bigwedge_{t,t' \in \mathsf{Pred}(\zeta,R)} (t = t')$$

Finally, we set $q^* = q \wedge q_1 \wedge q_2$. It turns out that q^* can be computed in polynomial time w.r.t. q [6]. In the same paper, Lutz et al. prove the following result.

Proposition 3. *Let $\mathcal{A}$ be an arbitrary $\mathcal{EL}$ ABox, let $\mathcal{I}$ be a split and $\mathcal{A}$-connected interpretation, and let $\mathcal{J}$ be the unraveling of $\mathcal{I}$ w.r.t. $\mathcal{A}$. Then, for every k-tuple of individuals $\langle a_1, \ldots, a_k \rangle$, we have that*

$$\mathcal{I} \models q^*[a_1 \ldots, a_k] \text{ if and only if } \mathcal{J} \models q[a_1 \ldots, a_k].$$

This result applies to our Datalog rewriting of $\mathcal{EL}$ TBoxes. Indeed, for an arbitrary $\mathcal{EL}$ KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, we have that $\mathsf{M}(P_{\mathcal{T}} \cup \mathcal{A})$ is a split and $\mathcal{A}$-connected interpretation. The intuition behind the argument is as follows. We show that $\mathsf{M}(P_{\mathcal{T}} \cup \mathcal{A})$ is split by noticing that rules encoded in $P_{\mathcal{T}}$ do not allow for the derivation of facts of the form $R(o_B, a)$ for $a \in \mathsf{Ind}(\mathcal{A})$ and $o_B \in \mathbf{Aux}$. To see that $\mathsf{M}(P_{\mathcal{T}} \cup \mathcal{A})$ is $\mathcal{A}$-connected, we just recall that $\mathsf{M}(P_{\mathcal{T}} \cup \mathcal{A})$ is minimal and, hence, all the derived facts must be "grounded" w.r.t. the facts in $\mathcal{A}$.

Theorem 2. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be an $\mathcal{EL}$ knowledge base. Then, $\mathsf{M}(P_\mathcal{T} \cup \mathcal{A})$ is a split and $\mathcal{A}$-connected interpretation.*

Hence, for an arbitrary k-ary CQ q and for each k-tuple of individuals $\langle a_1, \ldots, a_k \rangle$, we have that $\mathsf{M}(P_\mathcal{T} \cup \mathcal{A}) \models q^*[a_1 \ldots, a_k]$ if and only if $\langle a_1, \ldots, a_k \rangle \in \mathsf{cert}(q, \mathcal{K})$.

Note that q^* is a first-order query, and we are unaware of systems capable of evaluating first-order queries over DATALOG programs. Therefore, we next show how to transform q^* into a DATALOG query $\langle Q_P, Q_C \rangle$ such that $\langle a_1, \ldots, a_k \rangle \in \mathsf{cert}(q, \mathcal{K})$ if and only if $P_\mathcal{T} \cup \mathcal{A} \cup Q_C \models Q_P(a_1 \ldots, a_k)$. We construct such a DATALOG query $\langle Q_P, Q_C \rangle$ by applying to q^* a simplified version of the Lloyd-Topor transformation [14, 15].

Definition 1 (Datalog Rewriting). *Let $q(\vec{x})$ be a k-ary CQ whose quantified variables are among $\vec{y}$; let Cyc, $\mathsf{Fork}_{\neq}$, and $\mathsf{Fork}_{=}$ be as specified above; let $\langle \mathsf{Pred}(\zeta^1, R^1), t^1_\zeta \rangle$, ..., $\langle \mathsf{Pred}(\zeta^n, R^n), t^n_\zeta \rangle$ be an arbitrary enumeration of $\mathsf{Fork}_{=}$; let $p_0, p_1, \ldots, p_n$ be fresh predicates; and let Named be a built-in with a predetermined, possibly infinite Herbrand interpretation $\mathsf{N} = \{\mathsf{Named}(a) \mid a \in N_I\}$. Query Q_C then contains the following safe DATALOG rules:*

$$p_0(\vec{x}, \vec{y}) \leftarrow q, \bigwedge_{v \in \mathit{avar}(q) \cup \mathsf{Fork}_{\neq} \cup \mathsf{Cyc}} \mathsf{Named}(v) \tag{1}$$

$$p_i(\vec{x}, \vec{y}) \leftarrow p_{i-1}(\vec{x}, \vec{y}), \mathsf{Named}(t^i_\zeta) \qquad \textit{for } 1 \leq i \leq n \tag{2}$$

$$p_i(\vec{x}, \vec{y}) \leftarrow p_{i-1}(\vec{x}, \vec{y}), \bigwedge_{t, t' \in \mathsf{Pred}(\zeta^i, R^i)} t = t' \qquad \textit{for } 1 \leq i \leq n \tag{3}$$

$$Q_P(\vec{x}) \leftarrow p_n(\vec{x}, \vec{y}) \tag{4}$$

One may think that the recursive definition of predicates p_i for $1 \leq i \leq n$ could be simplified by writing $Q_P(\vec{x}) \leftarrow p_0(\vec{x}, \vec{y}) \ldots p_n(\vec{x}, \vec{y})$ and by defining each p_i as:

$$p_i(\vec{x}, \vec{y}) \leftarrow \mathsf{Named}(t^i_\zeta) \qquad p_i(\vec{x}, \vec{y}) \leftarrow \textstyle\bigwedge_{t, t' \in \mathsf{Pred}(\zeta^i, R^i)} t = t'$$

Unfortunately, these rules are not safe. Safe rules, on the one hand, provide us with a clear and unambiguous semantics. On the other hand, unsafe rules are also computationally more expensive for bottom-up computation, since each variable in the head may be bound to an arbitrary individual in the universe of the program. For this reason, we prefer our, slightly more involved, solution.

Proposition 4. *For an arbitrary k-ary conjunctive query q, query $\langle Q_P, Q_C \rangle$ can be computed in polynomial time w.r.t. the size of q.*

Proof. We note that $\sim_q$ can be computed in polynomial time w.r.t. the size of q [6] and, therefore, also the sets Cyc, $\mathsf{Fork}_{\neq}$, and $\mathsf{Fork}_{=}$ can be computed in polynomial time w.r.t. q. Furthermore, the size of the body of rule $p_0(\vec{x}, \vec{y})$ depends linearly on the size of q, Cyc, and $\mathsf{Fork}_{\neq}$. Also, for each pair $\langle \mathsf{Pred}(\zeta, R), t_\zeta \rangle$ in $\mathsf{Fork}_{=}$, the program Q_C contains exactly two rules. The size of these two rules depends linearly on the size of $\langle \mathsf{Pred}(\zeta, R), t_\zeta \rangle$. Thus, we conclude that $\langle Q_P, Q_C \rangle$ can be computed in polynomial time with respect to the size of q. □

In [10], we prove that our rewriting is correct—that is, that answering $\langle Q_P, Q_C \rangle$ over $P_\mathcal{T} \cup \mathcal{A}$ is equivalent to computing $\mathsf{cert}(q, \mathcal{T}, \mathcal{A})$. This follows from Proposition 3 and the fact that our Datalog query is the result of transforming the FO rewriting q^* along the lines of the Lloyd-Topor transformation.

Theorem 3. *Let $\mathcal{K}$ be an $\mathcal{EL}$ knowledge base and let q be a k-ary CQ over $\mathcal{K}$. Then, for every k-tuple of individuals $\langle a_1, \ldots, a_k \rangle$, we have that*

$$\langle a_1, \ldots, a_k \rangle \in \mathsf{cert}(q, \mathcal{K}) \text{ if and only if } P_\mathcal{T} \cup \mathcal{A} \cup Q_C \models Q_P(a_1, \ldots, a_k).$$

Finally, we investigate the complexity of our rewriting procedure.

Theorem 4. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be an $\mathcal{EL}$ KB, let q be a k-ary CQ, and let $\langle a_1, \ldots, a_k \rangle$ be a tuple of individuals. We can decide $P_\mathcal{T} \cup \mathcal{A} \cup Q_C \models Q_P(a_1, \ldots, a_k)$ in polynomial time w.r.t. the size of $\mathcal{K}$ and in non-deterministic polynomial time with respect to the size of both $\mathcal{K}$ and q.*

Proof. We have already argued that the size of Datalog program $P_\mathcal{T}$ depends linearly on the size of the TBox $\mathcal{T}$ and that the Datalog rewriting $\langle Q_P, Q_C \rangle$ can be computed in Ptime w.r.t. q. Also, we note that the arity of predicates and the number of variables occurring in $P_\mathcal{T} \cup \mathcal{A} \cup Q_C$ do not depend on $\mathcal{K}$. Finally, from an implementation point-of-view (as suggested in [12]), the built-in predicate Named can be considered as an assertion in the ABox $\mathcal{A}$ with a different physical realization: it is not directly stored in the ABox but it is implemented as a procedure which is evaluated during the execution of the program. Clearly, such a procedure can be implemented to run in time polynomial in $\mathcal{K}$. It follows that we can compute the minimal Herbrand model of $P_\mathcal{T} \cup \mathcal{A} \cup Q_C$ in time polynomial in the size of $\mathcal{K}$ [8]. The membership in NP follows directly from the considerations above and from the fact that we can guess and check in nondeterministic polynomial time a match π for Q_P in $\mathsf{M}(P_\mathcal{T} \cup \mathcal{A} \cup Q_C)$. □

5 Conclusions

In this paper, we introduce a query rewriting approach to query answering in $\mathcal{EL}$. In our approach, the process of computing the certain answers to a CQ q over an $\mathcal{EL}$ KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is divided into two distinct steps. A first preprocessing step in which the TBox $\mathcal{T}$ is transformed into a Datalog program $P_\mathcal{T}$, whose size is linear in $\mathcal{T}$. Then, at query time, the query q is independently rewritten into a Datalog query $\langle Q_P, Q_C \rangle$, whose size is polynomial in q. Finally, computing $\mathsf{cert}(q, \mathcal{K})$ amounts to evaluating the Datalog query $\langle Q_P, Q_C \rangle$ over $P_\mathcal{T} \cup \mathcal{A}$.

In future, we plan to extend our approach to deal with $\mathcal{ELH}^{dr}_{\bot}$. Lutz et al. have already proposed a combined approach for query answering in this DL [13]. However, differently from their solution, we would like the Datalog rewriting $\langle Q_P, Q_C \rangle$ to be independent from the role inclusions contained in the TBox. Additionally, we plan to extend our work to cover nominals, which raises the question on how to efficiently handle equality in Datalog [2].

References

1. 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.: Semantic Technologies for Informations Systems – 5th Int. Reasoning Web Summer School (RW 2009). Volume 5689. (2009) 255–356
2. Pérez-Urbina, H., Motik, B., Horrocks, I.: Tractable query answering and rewriting under description logic constraints. J. Applied Logic **8**(2) (2010) 186–209
3. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In Kaelbling, L.P., Saffiotti, A., eds.: IJCAI, Professional Book Center (2005) 364–369
4. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope further. In Clark, K., Patel-Schneider, P.F., eds.: In Proceedings of the OWLED 2008 DC Workshop on OWL: Experiences and Directions. (2008)
5. Rosati, R.: On conjunctive query answering in $\mathcal{EL}$. In Calvanese, D., Franconi, E., Haarslev, V., Lembo, D., Motik, B., Turhan, A.Y., Tessaris, S., eds.: Description Logics. Volume 250 of CEUR Workshop Proceedings., CEUR-WS.org (2007)
6. Lutz, C., Toman, D., Wolter, F.: Conjunctive query answering in $\mathcal{EL}$ using a database system. In: Proceedings of the OWLED 2008 Workshop on OWL: Experiences and Directions. (2008)
7. Kontchakov, R., Lutz, C., Toman, D., Wolter, F., Zakharyaschev, M.: The combined approach to ontology-based data access. In: IJCAI, AAAI Press (2011)
8. Abiteboul, S., Hull, R., Vianu, V.: Foundations of databases. Addison-Wesley (1995)
9. Chen, W., Warren, D.S.: Query evaluation under the well-founded semantics. In: Proceedings of the twelfth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems. PODS '93, New York, NY, USA, ACM (1993) 168–179
10. Stefanoni, G., Motik, B., Horrocks, I.: Small datalog query rewriting for $\mathcal{EL}$. Technical report, University of Oxford (2012) Available at http://www.cs.ox.ac.uk/people/giorgio.stefanoni/pubs/2012/tr/dl2012.pdf.
11. Ullman, J.D.: Principles of database and knowledge-base systems, Vol. I. Computer Science Press, Inc., New York, NY, USA (1988)
12. Ceri, S., Gottlob, G., Tanca, L.: What you always wanted to know about datalog (and never dared to ask). IEEE Trans. Knowl. Data Eng. **1**(1) (1989) 146–166
13. Lutz, C., Toman, D., Wolter, F.: Conjunctive query answering in the description logic $\mathcal{EL}$ using a relational database system. In Boutilier, C., ed.: IJCAI. (2009) 2070–2075
14. Lloyd, J., Topor, R.: Making prolog more expressive. The Journal of Logic Programming **1**(3) (1984) 225 – 240
15. Decker, S.: Semantic web methods for knowledge managmement. PhD thesis, University of Karlsruhe (2002)