Tractable Query Answering and Rewriting under Description Logic Constraints

Héctor Pérez-Urbina, Boris Motik, and Ian Horrocks

Computing Laboratory
University of Oxford
Oxford, UK
{hector.perez-urbina,boris.motik,ian.horrocks}@comlab.ox.ac.uk

Abstract. Answering queries over an incomplete database w.r.t. a set of constraints is an important computational task with applications in fields as diverse as information integration and metadata management in the Semantic Web. Description Logics (DL) are constraint languages that have been extensively studied in the past with the goal of providing useful modeling constructs while keeping the query answering problem decidable. For many DLs, query answering under constraints can be reduced to query rewriting: given a conjunctive query Q and a set of DL constraints $\mathcal{T}$, the query Q can be transformed into a datalog query $Q_\mathcal{T}$ that takes into account the semantic consequences of $\mathcal{T}$; then, to obtain answers to Q w.r.t. $\mathcal{T}$ and some (arbitrary) database instance $\mathcal{A}$, one can simply evaluate $Q_\mathcal{T}$ over $\mathcal{A}$ using existing (deductive) database technology, without taking $\mathcal{T}$ into account. In this paper, we present a novel query rewriting algorithm that handles constraints modeled in the DL $\mathcal{ELHIO}^\neg$ and use it to show that answering conjunctive queries in this setting is PTime-complete w.r.t. data complexity. Our algorithm deals with various description logics of the $\mathcal{EL}$ and DL-Lite families and is worst-case optimal w.r.t. data complexity for all of them.

1 Introduction

Answering conjunctive queries over incomplete databases lies at the core of numerous data management problems, such as answering queries using views [29], information integration [28], data exchange [16], and data warehousing [40]. Given a query and an incomplete database consisting of a set of constraints and a partial database instance [1], the problem is to compute the so-called *certain answers*—the tuples that satisfy the query in every database instance that conforms to the partial instance and satisfies the constraints [38].

It is well known that answering conjunctive queries under general first-order constraints is undecidable; therefore, the expressivity of the constraint languages considered has to be restricted in order to achieve decidability. Various decidable constraint languages have been considered in the field of databases. For instance, Calì and Kifer [9] consider first-order constraints derived from a restricted version of F-logic [25]; Simkus and Eiter [37] deal with expressive constraints based

on Answer Set Programming [19]; Calì et al. [10, 6] consider constraints tailored to express entity-relationship schemas; Fuxman and Miller [18] consider key constraints expressed over potentially inconsistent databases; and very recently, Calì et al. [7, 8] proposed a general datalog-based framework for query answering in which constraints are expressed as restricted TGDs (tuple-generating dependencies) and EGDs (equality-generating dependencies) [1].

DLs can be viewed as very expressive but decidable first-order fragments, which makes them natural candidates for constraint languages. DLs are a family of knowledge representation formalisms that can be used to represent a given domain in terms of concepts (unary predicates), roles (binary predicates), and individuals (constants) [3]. A DL Knowledge Base (KB) $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ consists of a *terminological* component $\mathcal{T}$ called the TBox, and an *assertional* component $\mathcal{A}$ called the ABox. In analogy to incomplete databases, the TBox can be seen as a conceptual schema containing a set of constraints, and the ABox as some partial instance of the schema. The use of DLs as constraint languages has already proven to be useful in a variety of scenarios such as ontology-based information integration [14, 28] and the Semantic Web [21].

The DL $\mathcal{ELHIO}^{\neg}$ is an an expressive extension of the basic DL $\mathcal{EL}$ [2]. The language includes (limited) concept and role negation, role inclusions, inverse roles, and nominals—concepts that are to be interpreted as singletons (e.g. {Mexico}). As we show in Section 6, $\mathcal{ELHIO}^{\neg}$ is one of the most expressive Horn logics for which query answering is polynomial w.r.t. data complexity. The approach to query answering that we adopt in this paper is based on *query rewriting*: given a conjunctive query Q and an $\mathcal{ELHIO}^{\neg}$ TBox $\mathcal{T}$, one computes a datalog query $Q_{\mathcal{T}}$—a so-called *rewriting* of Q w.r.t. $\mathcal{T}$—such that, for every ABox $\mathcal{A}$, the answers to Q over $\mathcal{T}$ and $\mathcal{A}$, and the answers to $Q_{\mathcal{T}}$ over $\mathcal{A}$ coincide. Thus, the problem of answering Q over $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ for a specific $\mathcal{A}$ can be reduced to evaluating the datalog query $Q_{\mathcal{T}}$ over $\mathcal{A}$ only. Such an approach to query answering is interesting from a practical perspective because it allows one to reuse existing (deductive) database systems for data storage and query evaluation, thus taking advantage of the extensive body of research in data storage, indexing, and query evaluation.

Various rewriting techniques for DL constraints have been proposed. Motik [31] presented a resolution-based algorithm for reducing very expressive DL KBs to disjunctive datalog programs. Kazakov [24] used saturation-based theorem proving to derive a range of decision procedures for various DLs of the $\mathcal{EL}$ family [2]. These approaches, however, do not consider conjunctive queries. Conjunctive query rewriting under DL constraints has been considered by Calvanese et al. [11] for the DL-Lite family of languages, for which query answering was shown to be in LogSpace w.r.t. data complexity; and by Rosati [35] for $\mathcal{EL}$. Conjunctive query answering for $\mathcal{EL}$ has been shown to be PTime-complete w.r.t. data complexity independently by Rosati [35], Krisnadhi and Lutz [26], and Krotzsch and Rudolph [27]. Finally, another rewriting technique for $\mathcal{EL}$ has been recently proposed by Lutz et al. [30]. However, their approach is quite different from the standard query rewriting approach since their algorithm rewrites both the query

and the ABox w.r.t. the TBox. All the aforementioned techniques are closely related; however, they have been designed to handle different DLs. In contrast, our goal is to obtain a *unified* rewriting algorithm, inspired by the resolution-based techniques presented in [31, 24], that *generalizes* and *extends* the techniques of [11] and [35].

In this paper, we present a rewriting algorithm that takes as input a conjunctive query Q and an $\mathcal{ELHIO}^{\neg}$ TBox $\mathcal{T}$, and computes a datalog query that is a rewriting of Q w.r.t. $\mathcal{T}$. We use the rewriting algorithm to obtain the novel result that conjunctive query answering for $\mathcal{ELHIO}^{\neg}$ is PTime-complete w.r.t. data complexity. Our rewriting algorithm exhibits "pay-as-you-go" behavior: if $\mathcal{T}$ is in $\mathcal{ELHIO}^{\neg}$, then the computed rewriting is a datalog query, as in [33, 35]; if $\mathcal{T}$ is in DL-Lite^{+} [32], then the rewriting consists of a union of conjunctive queries and a linear datalog query, as in [32]; finally, if $\mathcal{T}$ is in DL-Lite$_R$, then the rewriting is a union of conjunctive queries, as in [11]. Therefore, the rewriting algorithm not only deals with various DLs ranging from $\mathcal{ELHIO}^{\neg}$ down to DL-Lite$_{core}$ [11], but it is *optimal* w.r.t. data complexity for all such logics. We derive our rewriting algorithm from a novel query *answering* algorithm that can be used to obtain the certain answers of a conjunctive query Q over an $\mathcal{ELHIO}^{\neg}$ KB $\mathcal{K}$. Such an answering algorithm employs a resolution-based procedure that handles equality in a novel way, which makes our answering algorithm interesting in its own right.

This paper is an extended version of our work presented in [32] and [33], where we considered description logics DL-Lite^{+} and $\mathcal{ELHI}$, respectively.

2 Preliminaries

In this section we introduce all necessary terminology and recapitulate relevant definitions and results.

2.1 Logic Programming

We use the well-known definitions of constants, variables, function symbols, terms, and atoms of first-order logic [17]. With $\vec{t}$ we denote a tuple of terms $\langle t_1, ..., t_n \rangle$. For easier readability, we often write functional terms of the form $f_1(...(f_n(t))...)$ as $f_1...f_n(t)$. The *Herbrand universe* of a first-order signature $\mathcal{L}$, denoted $U_\mathcal{L}$, is the set of all ground terms that can be formed with the functions and constants of $\mathcal{L}$. An *atom* of $\mathcal{L}$ is any expression of the form $R(t_1, ..., t_n)$, where R is an n-place predicate symbol and $t_1, ..., t_n$ are terms. An atom is *ground* if all its terms are ground. The *Herbrand base* of $\mathcal{L}$, denoted $B_\mathcal{L}$, is the set of all ground atoms that can be formed with the predicates of $\mathcal{L}$ and the terms of $U_\mathcal{L}$.

A *Horn clause* is an expression of the form $H \leftarrow B_1 \wedge ... \wedge B_n$, where H and each B_i are atoms. The atom H is called the *head*, and the set of atoms $\{B_i\}$ is called the *body*. With $\square$ we denote the *empty clause*—that is, a clause where $H = \perp$ and the body is empty. A Horn clause C is *safe* if all the variables

occurring in the head also occur in the body. A clause C of the form $H \leftarrow$ is often written as H and is called a *fact*; furthermore, if H is a ground atom, then C is called a *ground fact*. An atom A is *covering* for a clause C if A contains all the variables occurring in C. With $\mathsf{var}(C)$ we denote the number of variables in a clause C. The *depth* of a term t is defined as $\mathsf{depth}(t) = 0$ if t is a constant or a variable, and $\mathsf{depth}(f(s)) = 1 + \mathsf{depth}(s)$ if t is a functional term $f(s)$. The depth of an atom $R(t_1, ..., t_n)$ is defined as $\mathsf{depth}(R(t_1, ..., t_n)) = \max(\mathsf{depth}(t_i))$ for $1 \leq i \leq n$ and of a Horn clause C as $\mathsf{depth}(C) = \max(\mathsf{depth}(H), \max(\mathsf{depth}(B_i)))$ for $1 \leq i \leq n$.

A *logic* program LP is a set of safe Horn clauses. The *extensional database (EDB) predicates* of LP are those that do not occur in the head of any Horn clause in LP; all other predicates are called *intensional database (IDB) predicates*. With each logic program LP we associate the signature $\mathcal{L}(LP)$ that consists of the predicates, functions, and constants occurring in LP; if no constant occurs in LP, we add an arbitrary constant to $\mathcal{L}(LP)$. A *Herbrand interpretation* of a logic program LP is any subset I of the Herbrand base $B_{\mathcal{L}(LP)}$ of $\mathcal{L}(LP)$. A *Herbrand model* of LP is a Herbrand interpretation I such that for each clause $C \in LP$ of the form $H \leftarrow B_1 \wedge ... \wedge B_n$, the interpretation I satisfies the first-order formula $\forall \vec{x}(B_1 \wedge ... \wedge B_n \rightarrow H)$, where $\vec{x}$ is a tuple of all the variables occurring in C. A logic program LP is called a *datalog* program if every Horn clause $C \in LP$ is function-free. A datalog program D is said to be *linear* if each Horn clause $C \in D$ contains at most one IDB predicate in the body.

2.2 Resolution with Free Selection

The Resolution with Free Selection (RFS) family of calculi is a set of resolution-based calculi that can be used to check whether a logic program LP is satisfiable [5]. Each RFS calculus is defined by a *selection function* S that assigns to each Horn clause $C \in LP$ a nonempty set of atoms such that either $S(C)$ is a singleton set containing the head of C, or $S(C)$ is a subset of the body of C. The atoms in $S(C)$ are said to be *selected* by S in C. Every RFS calculus $\mathcal{R}$ consists of the following inference rule only, called *resolution*:

$$\frac{A \leftarrow B_1 \wedge ... \wedge \underline{B_i} \wedge ... \wedge B_n \qquad \underline{C} \leftarrow D_1 \wedge ... \wedge D_m}{A\sigma \leftarrow B_1\sigma \wedge ... \wedge B_{i-1}\sigma \wedge B_{i+1}\sigma \wedge ... \wedge B_n\sigma \wedge D_1\sigma \wedge ... \wedge D_m\sigma}$$

The two clauses above the inference line are called the *premises* and the clause below is called the *resolvent*. W.l.o.g. we make a technical assumption that the premises do not have variables in common. The atoms B_i and C must be selected in the corresponding premises by the selection function S; we usually underline the selecting atoms as shown in the previous inference. Finally, $\sigma = \mathsf{MGU}(B_i, C)$ is the *most general unifier* of B_i and C as defined in [4].

A set of Horn clauses LP is *saturated* by $\mathcal{R}$ if, for every two premises P_1 and P_2 in LP and every resolvent P_R of P_1 and P_2, the set LP contains a clause equivalent to P_R up to variable renaming [4]. A *derivation* by $\mathcal{R}$ from a set of Horn clauses LP is a sequence of sets of Horn clauses $LP_0, LP_1, ...$ such that

Table 1. Semantics of $\mathcal{ELHIO}^{\neg}$

Semantics of concepts and roles:	Semantics of assertions:
$\top^{\mathcal{I}} = \triangle^{\mathcal{I}}$	
$\{a\}^{\mathcal{I}} = \{a^{\mathcal{I}}\}$	
$(B_1 \sqcap B_2)^{\mathcal{I}} = B_1^{\mathcal{I}} \cap B_2^{\mathcal{I}}$	$\mathcal{I} \models A(a)$ iff $a^{\mathcal{I}} \in A^{\mathcal{I}}$
$(\exists R)^{\mathcal{I}} = \{x \mid \exists y.\langle x, y \rangle \in R^{\mathcal{I}}\}$	$\mathcal{I} \models P(a,b)$ iff $\langle a^{\mathcal{I}}, b^{\mathcal{I}} \rangle \in P^{\mathcal{I}}$
$(\exists R.B)^{\mathcal{I}} = \{x \mid \exists y.\langle x, y \rangle \in R^{\mathcal{I}} \wedge y \in B^{\mathcal{I}}\}$	$\mathcal{I} \models B \sqsubseteq C$ iff $B^{\mathcal{I}} \subseteq C^{\mathcal{I}}$
$(P^{-})^{\mathcal{I}} = \{\langle x, y \rangle \mid \langle y, x \rangle \in P^{\mathcal{I}}\}$	$\mathcal{I} \models R \sqsubseteq E$ iff $R^{\mathcal{I}} \subseteq E^{\mathcal{I}}$
$(\neg B)^{\mathcal{I}} = \triangle^{\mathcal{I}} \setminus B^{\mathcal{I}}$	
$(\neg R)^{\mathcal{I}} = \triangle^{\mathcal{I}} \times \triangle^{\mathcal{I}} \setminus R^{\mathcal{I}}$	

$LP_0 = LP$ and for each $i \geq 0$ we have that $LP_{i+1} = LP_i \cup \{C\}$, where C is the resolvent of an inference by $\mathcal{R}$ of a pair of premises in LP_i. The limit $LP_{\mathcal{R}}$ of a fair derivation from a set of Horn clauses LP by $\mathcal{R}$ is defined as $LP_{\mathcal{R}} = \bigcup LP_i$. It is well known that $LP_{\mathcal{R}}$ is saturated by $\mathcal{R}$. A clause C is said to be *derivable* from LP by $\mathcal{R}$ iff $C \in LP_{\mathcal{R}}$. Resolution with free selection is sound and complete; that is, a set of Horn clauses LP is satisfiable iff $\square \notin LP_{\mathcal{R}}$ [5].

2.3 Description Logic $\mathcal{ELHIO}^{\neg}$

Let N_C, N_R, and N_I be countable, infinite, and pairwise disjoint sets of *atomic concepts*, *atomic roles*, and *constants*, respectively. $\mathcal{ELHIO}^{\neg}$ roles are built according to the following syntax rules, where R is called a *basic role*, E is called a *general role*, and $P \in N_R$:

$$R ::= P \mid P^{-} \qquad E ::= R \mid \neg R$$

A basic role of the form P^{-} is called the *inverse* role of P.

$\mathcal{ELHIO}^{\neg}$ concepts are built according to the following syntax rules, where B is called a *basic concept*, C is called a *general concept*, $a \in N_I$, $A \in N_C$, R is a basic role, and B_1 and B_2 are basic concepts:

$$B ::= A \mid \{a\} \mid \top \mid B_1 \sqcap B_2 \mid \exists R.B \qquad C ::= B \mid \neg B$$

An $\mathcal{ELHIO}^{\neg}$ *TBox* is a finite set of *axioms* of the form $B \sqsubseteq C$ (*concept inclusions*) or $R \sqsubseteq E$ (*role inclusions*), where B is a basic concept, C is a general concept, R is a basic role, and E is a general role. An axiom α is called a *negative inclusion* if it contains the symbol $\neg$; otherwise, it is called a *positive inclusion*. Given an $\mathcal{ELHIO}^{\neg}$ TBox $\mathcal{T}$, with $\mathcal{T}_{\text{NI}}$ we denote the set of all negative inclusions of $\mathcal{T}$, and with $\mathcal{T}_{\text{PI}}$ we denote the set of all positive inclusions of $\mathcal{T}$. An *ABox* is a finite set of *membership assertions* of the form $A(a)$ or $P(a,b)$, where $A \in N_C$, $P \in N_R$, $a \in N_I$ and $b \in N_I$. An $\mathcal{ELHIO}^{\neg}$ *Knowledge Base* (KB) $\mathcal{K}$ is a tuple $\langle \mathcal{T}, \mathcal{A} \rangle$, where $\mathcal{T}$ is an $\mathcal{ELHIO}^{\neg}$ TBox and $\mathcal{A}$ is an ABox.

An *interpretation* $\mathcal{I} = (\triangle^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a nonempty interpretation domain $\triangle^{\mathcal{I}}$ and a function $\cdot^{\mathcal{I}}$ that maps each atomic concept $A \in N_C$ to a subset $A^{\mathcal{I}}$ of $\triangle^{\mathcal{I}}$, each atomic role $P \in N_R$ to a subset $P^{\mathcal{I}}$ of $\triangle^{\mathcal{I}} \times \triangle^{\mathcal{I}}$, and each constant

$a \in N_I$ to an element $a^{\mathcal{I}}$ of $\triangle^{\mathcal{I}}$. The function $\cdot^{\mathcal{I}}$ is extended to complex concepts and roles as shown in the left column of Table 1. An interpretation $\mathcal{I}$ is a model of an inclusion or membership assertion α, written $\mathcal{I} \models \alpha$, if $\mathcal{I}$ and α satisfy the conditions shown in the right column of Table 1. An interpretation $\mathcal{I}$ is a *model* of a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, written $\mathcal{I} \models \mathcal{K}$, if $\mathcal{I}$ satisfies every assertion in $\mathcal{T}$ and $\mathcal{A}$. A KB $\mathcal{K}$ is said to be *satisfiable* if it has at least one model. A KB $\mathcal{K}$ *logically implies* an inclusion or membership assertion α, written $\mathcal{K} \models \alpha$, if every model of $\mathcal{K}$ is a model of α.

W.l.o.g. we can restrict our attention only to TBoxes in *normal form* in which all axioms are of the form $A_1 \sqsubseteq \{a\}$, $\{a\} \sqsubseteq A_1$, $A_1 \sqsubseteq (\neg)A_2$, $A_1 \sqcap A_2 \sqsubseteq A_3$, $A_1 \sqsubseteq \exists R_1$, $A_1 \sqsubseteq \exists R_1.A_2$, $\exists R_1 \sqsubseteq A_1$, $\exists R_1.A_1 \sqsubseteq A_2$, or $R_1 \sqsubseteq (\neg)R_2$, where A_1, A_2, and A_3 are in N_C, $a \in N_I$, and R_1 and R_2 are basic roles. Each TBox $\mathcal{T}$ can be transformed into an equisatisfiable TBox $\mathcal{T}'$ in normal form by systematically replacing complex concepts with atomic ones along the lines of [2]. This process can produce axioms of the form $\top \sqsubseteq C$, which are then replaced by $A \sqsubseteq C$, $\{a\} \sqsubseteq A$, and $\exists R \sqsubseteq A$, for every $A \in N_C$, every individual a, and every basic role R occurring in the TBox; it is straightforward to see that this transformation preserves satisfiability of a knowledge base.

Given two DLs $\mathcal{L}_1$ and $\mathcal{L}_2$, we say that $\mathcal{L}_1$ is a *fragment* of $\mathcal{L}_2$ if each axiom of $\mathcal{L}_1$ is an axiom of $\mathcal{L}_2$. $\mathcal{ELHIO}$ is a fragment of $\mathcal{ELHIO}^{\neg}$ obtained by disallowing negative inclusions. $\mathcal{ELHI}$ is a fragment of $\mathcal{ELHIO}$ obtained by disallowing basic concepts of the form $\{a\}$. $\mathcal{ELH}$ is a fragment of $\mathcal{ELHI}$ obtained by disallowing inverse roles. $\mathcal{EL}$ is a fragment of $\mathcal{ELH}$ obtained by disallowing role inclusions. DL-Lite$^+$ is a fragment of $\mathcal{ELH}$ obtained by disallowing basic concepts of the form $B_1 \sqcap B_2$. DL-Lite$_R$ is a fragment of $\mathcal{ELHIO}^{\neg}$ obtained by disallowing basic concepts of the form $\{a\}$, $\top$, and $B_1 \sqcap B_2$, as well as axioms of the form $\exists R.B \sqsubseteq C$. DL-Lite$_{core}$ is a fragment of DL-Lite$_R$ obtained by disallowing role inclusions.

2.4 Queries

A *datalog query* Q is a tuple $\langle Q_P, Q_C \rangle$, where Q_P is a *predicate symbol* and Q_C is a datalog program. Q is a *linear datalog query* if Q_C is a linear datalog program; Q is called a *union of conjunctive queries* if Q_P is the only IDB predicate in Q_C, and the body of each clause in Q_C does not contain Q_P; finally, Q is a *conjunctive query* if it is a union of conjunctive queries and Q_C contains exactly one Horn clause. We may denote a conjunctive query $Q = \langle Q_P, Q_C \rangle$ simply with Q_C. A tuple of constants $\vec{a}$ is a *certain answer* of a datalog query $Q = \langle Q_P, Q_C \rangle$ over a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ iff $\mathcal{K} \cup Q_C \models Q_P(\vec{a})$, where Q_C is considered to be a set of universally quantified implications with the usual first-order semantics. The set of all certain answers of Q over $\mathcal{K}$ is denoted by $\mathsf{ans}(Q, \mathcal{K})$.

3 Approach Overview

Given an $\mathcal{ELHIO}^{\neg}$ KB $\mathcal{K}$ and a conjunctive query $Q = \langle Q_P, Q_C \rangle$ over $\mathcal{K}$, our goal is to compute the answers of Q over $\mathcal{K}$. According to the definition of the

certain answers, we have that $\vec{a}$ is an answer to Q over $\mathcal{K}$ iff $Q_P(\vec{a})$ is a logical consequence of $\mathcal{K} \cup Q_C$. If $\mathcal{K}$ is unsatisfiable, then $\mathsf{ans}(Q, \mathcal{K})$ is trivially the set of all possible n-ary tuples of constants of $\mathcal{K}$, where n is the arity of Q_P. We are only interested in computing meaningful answers; therefore, before proceeding, we first need to check the satisfiability of $\mathcal{K}$.

An $\mathcal{ELHIO}^{\lnot}$ KB may be unsatisfiable due to the presence of negative inclusions that can lead to contradictions. It is possible to reduce the problem of KB satisfiability for $\mathcal{ELHIO}^{\lnot}$ to the problem of query answering along the lines of [11]. The idea is to transform each negative inclusion $\alpha \in \mathcal{T}_{\mathrm{NI}}$ into a boolean conjunctive query Q_α such that $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is satisfiable iff for every $\alpha \in \mathcal{T}_{\mathrm{NI}}$, the query Q_α is false over $\mathcal{K}' = \langle \mathcal{T}_{\mathrm{PI}}, \mathcal{A} \rangle$—that is, if $\mathsf{ans}(Q_\alpha, \mathcal{K}') = \emptyset$. The transformation is essentially the same as the one in [11], so we omit the details for the sake of brevity and just illustrate the transformation with an example.

Example 1. Consider an $\mathcal{ELHIO}^{\lnot}$ KB $\mathcal{K}_0 = \langle \mathcal{T}_0, \mathcal{A}_0 \rangle$ such that $\mathcal{T}_0$ contains the axiom $\alpha_0 = \mathsf{Theist} \sqsubseteq \lnot \mathsf{Atheist}$, and $\mathcal{A}_0$ contains the assertions $\mathsf{Theist}(\mathsf{John})$ and $\mathsf{Atheist}(\mathsf{John})$. The negative inclusion α_0 intuitively says that if an individual is an instance of Theist, then it is not an instance of Atheist. Therefore, if there is an individual that is an instance of both Theist and Atheist according to $\mathcal{K}_0$, then there is a contradiction and $\mathcal{K}_0$ can have no model (i.e., it is unsatisfiable).

We can check if $\mathcal{K}_0$ is unsatisfiable by answering the boolean query

$$Q_{\alpha_0}() \leftarrow \mathsf{Theist}(x) \land \mathsf{Atheist}(x)$$

over $\mathcal{K}_0$. Clearly, $\mathsf{ans}(Q_{\alpha_0}, \mathcal{K}_0) \neq \emptyset$; therefore, $\mathcal{K}_0$ is unsatisfiable.

Since the language of conjunctive queries does not allow for negated atoms, and $\mathcal{ELHIO}^{\lnot}$ does not allow for disjunction, the set $\mathcal{T}_{\mathrm{NI}}$ of negative inclusions can be simply regarded as a set of "constraints" that can only be used to check the consistency of $\mathcal{A}$ w.r.t. $\mathcal{T}$. Therefore, if the input knowledge base is satisfiable, in the same vein as in [11] it can be shown that negative inclusions are not needed for query answering; that is, $\mathsf{ans}(Q, \langle \mathcal{T}, \mathcal{A} \rangle) = \mathsf{ans}(Q, \langle \mathcal{T}_{\mathrm{PI}}, \mathcal{A} \rangle)$ for every satisfiable $\mathcal{ELHIO}^{\lnot}$ knowledge base $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$. Therefore, in the rest of this paper we assume that $\mathcal{T}$ does not contain negative inclusions (i.e., it is expressed in $\mathcal{ELHIO}$).

By the definition of certain answers, $\vec{a} \in \mathsf{ans}(Q, \mathcal{K})$ iff the logic program $LP = \Xi(\mathcal{K}) \cup Q_C \cup \{\bot \leftarrow Q_P(\vec{a})\}$ is unsatisfiable, where $\Xi(\cdot)$ transforms $\mathcal{K}$ into an equisatisfiable logic program $\Xi(\mathcal{K})$. By the well-known relationship between DLs and first-order logic [3], $\Xi(\mathcal{K})$ corresponds to the transformation shown in Table 2. The resulting clauses can contain the equality predicate $\approx$; for example, the $\mathcal{ELHIO}$ TBox axiom $\mathsf{Pope} \sqsubseteq \{\mathsf{BenedictXVI}\}$ is transformed into a clause $x \approx \mathsf{BenedictXVI} \leftarrow \mathsf{Pope}(x)$. To check the satisfiability of $\Xi(\mathcal{K})$ we thus need a calculus capable of effectively dealing with equality. Resolution-based calculi such as paramodulation and superposition [5] have been specially designed to improve efficiency of reasoning with equality. We base our work, however, on Resolution with Free Selection, mainly because we want to extend our previous work [33] where we employed an RFS calculus to handle various sublanguages

Table 2. Translating an $\mathcal{ELHIO}$ KB $\mathcal{K}$ into a set of clauses $\Xi(\mathcal{K})$

$\mathcal{ELHIO}$ clause	$\mathcal{ELHIO}$ axiom
$A(a)$	$A(a), \{a\} \sqsubseteq A$
$P(a,b)$	$P(a,b)$
$x \approx a \leftarrow A(x)$	$A \sqsubseteq \{a\}$
$A_2(x) \leftarrow A_1(x)$	$A_1 \sqsubseteq A_2$
$A_3(x) \leftarrow A_1(x) \wedge A_2(x)$	$A_1 \sqcap A_2 \sqsubseteq A_3$
$P(x, f(x)) \leftarrow A(x)$	$A \sqsubseteq \exists P$
$P(x, f(x)) \leftarrow A_1(x)$	$A_1 \sqsubseteq \exists P.A_2$
$A_2(f(x)) \leftarrow A_1(x)$	
$P(f(x), x) \leftarrow A(x)$	$A \sqsubseteq \exists P^-$
$P(f(x), x) \leftarrow A_1(x)$	$A_1 \sqsubseteq \exists P^-.A_2$
$A_2(f(x)) \leftarrow A_1(x)$	
$A(x) \leftarrow P(x, y)$	$\exists P \sqsubseteq A$
$A_2(x) \leftarrow P(x, y) \wedge A_1(y)$	$\exists P.A_1 \sqsubseteq A_2$
$A(x) \leftarrow P(y, x)$	$\exists P^- \sqsubseteq A$
$A_2(x) \leftarrow P(y, x) \wedge A_1(y)$	$\exists P^-.A_1 \sqsubseteq A_2$
$S(x, y) \leftarrow P(x, y)$	$P \sqsubseteq S, P^- \sqsubseteq S^-$
$S(x, y) \leftarrow P(y, x)$	$P^- \sqsubseteq S, P \sqsubseteq S^-$

Note 1. Each axiom of the form $A \sqsubseteq \exists R.B$ is uniquely associated with a distinct function symbol f.

of $\mathcal{ELHIO}^{\neg}$. Equality can be handled in RFS by treating the equality predicate $\approx$ as just an ordinary predicate and axiomatizing its properties by a set of clauses $E_{\mathcal{T}}$ which ensures that $\mathsf{ans}(Q, \mathcal{K}) = \mathsf{ans}(Q, \Xi(\mathcal{K}) \cup E_{\mathcal{T}})$. It is well known, however, that saturating a set of clauses containing $E_{\mathcal{T}}$ causes the generation of an infinite number of clauses [5]. In order to avoid this problem, we develop an *approximation of equality* $E'_{\mathcal{T}}$ and we show how to compute $\mathsf{ans}(Q, \mathcal{K})$ from $\mathsf{ans}(Q, \Xi(\mathcal{K}) \cup E'_{\mathcal{T}})$.

Our answering algorithm is based on a procedure that decides whether $\vec{a}$ is in $\mathsf{ans}(Q, \Xi(\mathcal{K}) \cup E'_{\mathcal{T}})$ by checking whether the empty clause is derivable from $LP \cup E'_{\mathcal{T}}$ by $\mathcal{R}^{DL}$—a suitable RFS calculus. Note, however, that such an approach allows one only to decide whether some tuple $\vec{a}$ is an answer to Q over $\Xi(\mathcal{K}) \cup E'_{\mathcal{T}}$. In order to compute the entire set $\mathsf{ans}(Q, \Xi(\mathcal{K}) \cup E'_{\mathcal{T}})$, we apply the so-called *answer literal technique* (cf. [20]) and we show that

$$\mathsf{ans}(Q, \Xi(\mathcal{K}) \cup E'_{\mathcal{T}}) = \{Q_P(\vec{a}) \mid Q_P(\vec{a}) \in (\Xi(\mathcal{K}) \cup E'_{\mathcal{T}} \cup Q_C)_{\mathcal{R}^{DL}}\}.$$

Therefore, our answering algorithm amounts to saturating $\Xi(\mathcal{K}) \cup E'_{\mathcal{T}} \cup Q_C$ by $\mathcal{R}^{DL}$ and returning each tuple $\vec{a}$ such that $Q_P(\vec{a}) \in (\Xi(\mathcal{K}) \cup E'_{\mathcal{T}} \cup Q_C)_{\mathcal{R}^{DL}}$. Since RFS is sound and complete, out main challenge is to ensure that the saturation of $\Xi(\mathcal{K}) \cup E'_{\mathcal{T}} \cup Q_C$ by $\mathcal{R}^{DL}$ terminates. We present our answering algorithm in Section 4.

Based on the query answering algorithm, in Section 5 we then present our query rewriting algorithm. The rewriting algorithm takes as input Q and $\mathcal{T}$ and derives the datalog query $\mathsf{rew}(Q, \mathcal{T})$ by saturating $\Xi(\mathcal{T}) \cup E'_{\mathcal{T}} \cup Q_C$ by $\mathcal{R}^{DL}$.

We obtain the rewriting $\mathsf{rew}(Q, \mathcal{T})$ by computing $(\Xi(\mathcal{T}) \cup E'_{\mathcal{T}} \cup Q_C)_{\mathcal{R}^{DL}}$ and then removing all clauses containing functional terms. We also introduce a step that ensures the optimality of $\mathsf{rew}(Q, \mathcal{T})$ for various sublanguages of $\mathcal{ELHIO}$: if $\mathcal{T}$ is in $\mathcal{ELHIO}$, then $\mathsf{rew}(Q, \mathcal{T})$ is a datalog query; if $\mathcal{T}$ is in DL-Lite$^+$, then $\mathsf{rew}(Q, \mathcal{T})$ consists of a union of conjunctive queries and a linear datalog query; and if $\mathcal{T}$ is in DL-Lite$_R$, then $\mathsf{rew}(Q, \mathcal{T})$ is a union of conjunctive queries. We present our rewriting algorithm in Section 5.

4   Resolution-based Query Answering

In this section we present a conjunctive query answering algorithm for $\mathcal{ELHIO}$. Roughly speaking, our algorithm first computes an approximation of equality $E'_{\mathcal{T}}$ for $\Xi(\mathcal{K})$. Next, it computes $\mathsf{ans}(Q, \Xi(\mathcal{K}) \cup E'_{\mathcal{T}})$ by returning every tuple $\vec{a}$ such that $Q_P(\vec{a}) \in (\Xi(\mathcal{K}) \cup E'_{\mathcal{T}} \cup Q_C)_{\mathcal{R}^{DL}}$, where $\mathcal{R}^{DL}$ is an RFS calculus as defined in Section 4.2. Finally, the algorithm computes $\mathsf{ans}(Q, \mathcal{K})$ from $\mathsf{ans}(Q, \Xi(\mathcal{K}) \cup E'_{\mathcal{T}})$. In Section 4.1 we present our approximation of equality $E'_{\mathcal{T}}$ and we show how to compute $\mathsf{ans}(Q, \mathcal{K})$ from $\mathsf{ans}(Q, \Xi(\mathcal{K}) \cup E'_{\mathcal{T}})$ based on the notion of *representatives*. Before formally showing the correctness of our algorithm, we present an example in which we give informal intuitive explanations; the details are provided in the subsequent sections.

Example 2. Consider an $\mathcal{ELHIO}$ KB $\mathcal{K}_1 = \langle \mathcal{T}_1, \mathcal{A}_1 \rangle$ about monotheistic religions. Let $\mathcal{T}_1$ contain the following axioms:

$$\text{Religion} \sqsubseteq \exists \text{hasDevotee}$$
$$\exists \text{hasDevotee}^-.\text{Religion} \sqsubseteq \text{Theist}$$
$$\text{Theist} \sqsubseteq \exists \text{believesIn}.\{\mathsf{God}\}$$
$$\{\mathsf{FSM}\} \sqsubseteq \{\mathsf{God}\}$$

The TBox $\mathcal{T}_1$ states that a religion has at least one devotee, that someone who is a devotee of a religion is a theist, that a theist is someone who believes in God, and that the Flying Spaghetti Monster and God are the same individual.

Let $\mathcal{A}_1$ contain the following assertions:[1]

$$\text{Religion}(\mathsf{Pastafarism}) \tag{1}$$

$$\text{hasDeity}(\mathsf{Pastafarism}, \mathsf{FSM}) \tag{2}$$

$$\text{Mighty}(\mathsf{FSM}) \tag{3}$$

Finally, consider the following query Q_1:

$$Q_1(z) \leftarrow \underline{\text{hasDevotee}(x, y)} \wedge \underline{\text{believesIn}(y, z)} \wedge \underline{\text{hasDeity}(x, z)} \wedge \underline{\text{Mighty}(z)} \tag{4}$$

Note that Pastafarism is a religion, so it has at least one devotee who believes in God. The Flying Spaghetti Monster is mighty, and it is the deity of

---

[1] We underline the atoms selected by the selection function of $\mathcal{R}^{DL}$.

Pastafarism. Therefore, since the Flying Spaghetti Monster and God denote the same individual, we expect that $\{\mathsf{God}, \mathsf{FSM}\} \subseteq \mathsf{ans}(Q_1, \mathcal{K}_1)$.

We now show how our answering algorithm obtains the set $\mathsf{ans}(Q_1, \mathcal{K}_1)$. We start by translating $\mathcal{T}_1$ into clauses and compute $\Xi(\mathcal{T}_1)$. According to Table 2, the set $\Xi(\mathcal{T}_1)$ contains the following clauses:[2]

$$\text{hasDevotee}(x, \text{devoteeOf}(x)) \leftarrow \text{Religion}(x) \tag{5}$$

$$\text{Theist}(x) \leftarrow \text{hasDevotee}(y, x) \wedge \text{Religion}(y) \tag{6}$$

$$\text{believesIn}(x, \text{dietyOf}(x)) \leftarrow \text{Theist}(x) \tag{7}$$

$$A_1(\text{dietyOf}(x)) \leftarrow \text{Theist}(x) \tag{8}$$

$$x \approx \mathsf{God} \leftarrow A_1(x) \tag{9}$$

$$x \approx \mathsf{God} \leftarrow A_2(x) \tag{10}$$

$$A_2(\mathsf{FSM}) \tag{11}$$

According to Definition 2, the approximation of equality $E'_{\mathcal{T}_1}$ for $\mathcal{T}_1$ contains the following clauses:

$$\mathcal{O}(\mathsf{God}) \tag{12}$$

$$\text{Mighty}(y) \leftarrow \text{Mighty}(x) \wedge x \approx y \wedge \mathcal{O}(y) \tag{13}$$

$$\text{hasDeity}(x, z) \leftarrow \text{hasDeity}(x, y) \wedge y \approx z \wedge \mathcal{O}(z) \tag{14}$$

$$\text{believesIn}(x, z) \leftarrow \text{believesIn}(x, y) \wedge y \approx z \wedge \mathcal{O}(z) \tag{15}$$

Clause (12) identifies God as an $\mathcal{O}$-*constant* —a constant that occurs in a clause of the form $x \approx o \leftarrow A(x)$ (cf. clauses (9) and (10)). $\mathcal{O}$-constants play an important role in our approach since they underpin the notion of a *representative*. Intuitively, a representative is an $\mathcal{O}$-constant that "gathers" all the relevant information of all terms that are mutually equal. Clauses (13)–(15) intuitively say that, if an individual is equal to an $\mathcal{O}$-constant, then it can be replaced with such an $\mathcal{O}$-constant. Restricting substitutivity to $\mathcal{O}$-constants reduces the number of clauses generated during the saturation.

As shown in Table 3, the set $(\Xi(\mathcal{K}_1) \cup E'_{\mathcal{T}_1} \cup \{Q_1\})_{\mathcal{R}^{DL}}$ contains the following clauses:

$$\text{hasDeity}(\mathsf{Pastafarism}, \mathsf{God}) \tag{16}$$

$$\text{Mighty}(\mathsf{God}) \tag{17}$$

$$\text{believesIn}(x, \mathsf{God}) \leftarrow \text{Theist}(x) \tag{18}$$

$$\text{believesIn}(\text{devoteeOf}(x), \mathsf{God}) \leftarrow \text{Religion}(x) \tag{19}$$

$$Q_1(\mathsf{God}) \leftarrow \text{hasDevotee}(\mathsf{Pastafarism}, \text{devoteeOf}(x)) \wedge \text{Religion}(x) \tag{20}$$

$$Q_1(\mathsf{God}) \tag{21}$$

---

[2] We only show the clauses that are relevant to derive the answers to Q_1. We introduced A_1 and A_2 in order to normalize $\mathcal{T}_1$ (see Section 2).

Table 3. Inferences on $\Xi(\mathcal{K}_1) \cup E'_{\mathcal{T}_1}$ by $\mathcal{R}^{DL}$ (cf. Example 2)

Clause	$\mathcal{R}^{DL}$ inferences involved
(16)	$[[[(12)+(14)]+(10)]+(2)]+(11)$
(17)	$[[[(12)+(13)]+(10)]+(3)]+(11)$
(18)	$[[[(12)+(15)]+(9)]+(7)]+(8)$
(19)	$[(5)+(6)]+(18)$
(20)	$[[(4)+(16)]+(17)]+(19)$
(21)	$[(20)+(5)]+(1)$

Note 2. $(x)+(y)$ means that the clause was obtained by resolving clauses (x) and (y).

It can be shown that (21) is the only clause of the form $Q_1(\vec{a})$ contained in $(\Xi(\mathcal{K}_1) \cup E'_{\mathcal{T}_1} \cup \{Q_1\})_{\mathcal{R}^{DL}}$; so, $\mathsf{ans}(Q_1, \Xi(\mathcal{K}_1) \cup E'_{\mathcal{T}_1}) = \{\mathsf{God}\}$. In this case, we compute $\mathsf{ans}(Q_1, \mathcal{K}_1)$ by querying $\Xi(\mathcal{K}_1) \cup E'_{\mathcal{T}_1}$ for all individuals that are equal to God; in our example it is possible to show that the only such individual is FSM, so $\mathsf{ans}(Q_1, \mathcal{K}_1) = \{\mathsf{God}, \mathsf{FSM}\}$.

4.1 Approximating Equality

In this section we present our approximation of equality $E'_{\mathcal{T}}$ and a simple procedure to compute $\mathsf{ans}(Q, \mathcal{K})$ from $\mathsf{ans}(Q, \Xi(\mathcal{K}) \cup E'_{\mathcal{T}})$. Therefore, the problem of answering Q over $\mathcal{K}$ can be reduced to the problem of answering Q over $\Xi(\mathcal{K}) \cup E'_{\mathcal{T}}$.

We first recapitulate the well-known *axiomatization of equality*—a set of clauses $E_{\mathcal{T}}$ that encode the properties of the equality relation, typically denoted with $\approx$, such that $\mathsf{ans}(Q, \mathcal{K}) = \mathsf{ans}(Q, \Xi(\mathcal{K}) \cup E_{\mathcal{T}})$.

Definition 1. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be an $\mathcal{ELHIO}$ KB. Let $E_{\mathcal{T}}$ be the set containing exactly the following clauses, where (i) one functional monotonicity clause is instantiated for every functional symbol f occurring in $\Xi(\mathcal{T})$, (ii) one unary substitutivity clause is instantiated for every unary predicate A occurring in $\Xi(\mathcal{T})$, and (iii) one binary substitutivity 1 clause and one binary substitutivity 2 clause are instantiated for every binary predicate P occurring in $\Xi(\mathcal{T})$:*[3]

$$x \approx x \qquad\qquad\qquad\qquad\qquad (reflexivity)$$
$$x \approx y \leftarrow y \approx x \qquad\qquad\qquad\qquad (symmetry)$$
$$x \approx z \leftarrow x \approx y \wedge y \approx z \qquad\qquad\quad (transitivity)$$
$$f(x) \approx f(y) \leftarrow x \approx y \qquad\qquad (functional\ monotonicity)$$
$$A(y) \leftarrow A(x) \wedge x \approx y \qquad\qquad (unary\ substitutivity)$$
$$P(x, z) \leftarrow P(x, y) \wedge y \approx z \qquad\quad (binary\ substitutivity\ 1)$$
$$P(z, x) \leftarrow P(y, x) \wedge y \approx z \qquad\quad (binary\ substitutivity\ 2)$$

---

[3] W.l.o.g. we assume that all the symbols in $\mathcal{A}$ occur in $\mathcal{T}$.

Our approximation of equality $E'_{\mathcal{T}}$ "weakens" the axiomatization of equality $E_{\mathcal{T}}$ in three ways: first, the relation $\approx$ is only required to be (implicitly) symmetric; second, the substitutivity clauses are limited to be applicable to certain constants only ($\mathcal{O}$-constants); and third, functional monotonicity is dropped. We formally define $E'_{\mathcal{T}}$ as follows.

Definition 2. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be an $\mathcal{ELHIO}$ KB. A constant o occurring in $\Xi(\mathcal{K})$ is called an $\mathcal{O}$-constant if it appears in a clause $C \in \Xi(\mathcal{K})$ of the form $x \approx o \leftarrow A(x)$.*

We define the set $E'_{\mathcal{T}}$ as follows. Let $\mathcal{O}$ be a fresh predicate not occurring in $\Xi(\mathcal{T})$. If there is no $\mathcal{O}$-constant o occurring in $\Xi(\mathcal{K})$, then $E'_{\mathcal{T}} = \emptyset$; otherwise let $E'_{\mathcal{T}}$ be the set that contains the following clauses, where (i) one $\mathcal{O}$-constant identification clause is instantiated for every $\mathcal{O}$-constant o occurring in $\Xi(\mathcal{T})$, (ii) one unary substitutivity$_{\mathcal{O}}$ clause is instantiated for every unary predicate A occurring in $\Xi(\mathcal{T})$, (iii) and one binary substitutivity$_{\mathcal{O}}$ 1 clause and one binary substitutivity$_{\mathcal{O}}$ 2 clause are instantiated for every binary predicate P occurring in $\Xi(\mathcal{T})$:

$$\mathcal{O}(o) \qquad\qquad (\mathcal{O}\text{-}constant\ identification)$$
$$A(y) \leftarrow A(x) \wedge x \approx y \wedge \mathcal{O}(y) \qquad\qquad (unary\ substitutivity_{\mathcal{O}})$$
$$P(x, z) \leftarrow P(x, y) \wedge y \approx z \wedge \mathcal{O}(z) \qquad\qquad (binary\ substitutivity_{\mathcal{O}}\ 1)$$
$$P(z, x) \leftarrow P(y, x) \wedge y \approx z \wedge \mathcal{O}(z) \qquad\qquad (binary\ substitutivity_{\mathcal{O}}\ 2)$$

We make the standard assumption of equational theorem proving that $\approx$ is implicitly symmetric; thus, each atom $s \approx t$ should be also read as $t \approx s$. The two forms should be considered interchangeable; for example, both forms should be considered when applying RFS inferences to clauses containing equality atoms.

Our goal is to show that $\mathsf{ans}(Q, \mathcal{K})$ can be computed from $\mathsf{ans}(Q, \Xi(\mathcal{K}) \cup E'_{\mathcal{T}})$. In our proofs we use the notion of *representatives*. Intuitively, the idea is to define a unique representative term for every set of terms that are equal according to $\mathcal{K}$. The role of the representative is to "gather" all the information of all terms that are mutually equal: we ensure that everything that follows from $\mathcal{K}$ for the terms that are mutually euqal also follows from $\Xi(\mathcal{K}) \cup E'_{\mathcal{T}}$ for the corresponding representative term. Therefore, we argue that it is possible to answer queries over $\mathcal{K}$ by answering them over $\Xi(\mathcal{K}) \cup E'_{\mathcal{T}}$, and then "expanding" the answers to the set of represented terms. We formally define the representative $[s]_I$ of a term s as follows.

Definition 3. *Let $\prec_{\mathcal{K}}$ be a total well-founded order on the set of terms occurring in the Herbrand universe of $\Xi(\mathcal{K}) \cup E'_{\mathcal{T}}$ such that, for all terms s and t, we have that $\mathsf{depth}(s) < \mathsf{depth}(t)$ implies $s \prec_{\mathcal{K}} t$, and for every $\mathcal{O}$-constant o and every term s, we have that $o \prec_{\mathcal{K}} s$.*

Let I be the minimal Herbrand model of $\Xi(\mathcal{K}) \cup E'_{\mathcal{T}}$. For all terms s and t occurring in I, we say that s and t are $\approx$-connected w.r.t. I, written $s \approx^*_I t$, if $s = t$ or there are terms $u_1, ..., u_n$ such that $\{s \approx u_1, u_1 \approx u_2, ..., u_n \approx t\} \subseteq I$. For every term s occurring in I, with $\mathsf{min}(s)$ we denote the term such that $s \approx^*_I \mathsf{min}(s)$, and there is no other term t such that $s \approx^*_I t$ and $t \prec_I \mathsf{min}(s)$.

For every term s occurring in the Herbrand universe of $\Xi(\mathcal{K}) \cup E'_{\mathcal{T}}$, we define the representative $[s]_I$ of s w.r.t. I inductively as follows: if s is a constant or if there is an $\mathcal{O}$-constant o such that $s \approx o \in I$, then $[s]_I = \mathsf{min}(s)$; otherwise, s is of the form $f(s')$ and $[s]_I = \mathsf{min}(f([s']_I))$. For $\vec{s} = \langle s_1, ..., s_n \rangle$ a tuple of n terms, with $[\vec{s}]_I$ we denote the tuple $\langle [s_1]_I, ..., [s_n]_I \rangle$.

As we show later in Lemma 9, a tuple of representative constants $[\vec{a}]_I$ is in $\mathsf{ans}(Q, \mathcal{K})$ iff $[\vec{a}]_I$ is in $\mathsf{ans}(Q, \Xi(\mathcal{K}) \cup E'_{\mathcal{T}})$. Moreover, if two constants are implied to be equal by $\mathcal{K}$, then they have the same representative. Therefore, we can compute $\mathsf{ans}(Q, \mathcal{K})$ by replacing every tuple $\langle a_1, ..., a_n \rangle$ in $\mathsf{ans}(Q, \Xi(\mathcal{K}) \cup E'_{\mathcal{T}})$ with $r_{a_1} \times ... \times r_{a_n}$, where r_{a_i} for $1 \leq i \leq n$ is the set of all the constants represented by a_i—that is, $r_{a_i} = \{b \mid [b]_I = a_i\}$. As we will see, for every constant a, constants $o_1, ..., o_m$ exist such that $\Xi(\mathcal{K}) \cup E'_{\mathcal{T}} \models \{a \approx o_1, o_1 \approx o_2, ..., o_m \approx [a]_I\}$ (cf. Lemma 2). Hence, we can obtain every set r_{a_i} by answering the query $Q_{\mathrm{eq}}(x, y) \leftarrow x \approx y$ over $\Xi(\mathcal{K}) \cup E'_{\mathcal{T}}$ and selecting all the constants from which there is a path to a_i (including itself). We illustrate the use of the representative with an example.

Example 3. Consider an $\mathcal{ELHIO}$ TBox $\mathcal{T}_2$ containing the following axioms:

$$\{\mathsf{FSM}\} \sqsubseteq \{\mathsf{God}\}$$
$$\{\mathsf{God}\} \sqsubseteq \{\mathsf{Zeus}\}$$

Consider an ABox $\mathcal{A}_2$ containing the following assertions:

$$\mathrm{Mighty}(\mathsf{FSM})$$
$$\mathrm{Omniscient}(\mathsf{God})$$
$$\mathrm{Omnipotent}(\mathsf{Zeus})$$

Consider the following query Q_2:

$$Q_2(x) \leftarrow \mathrm{Mighty}(x) \wedge \mathrm{Omniscient}(x) \wedge \mathrm{Omnipotent}(x)$$

Clearly, FSM, God and Zeus are equal according to $\mathcal{K}_2 = \langle \mathcal{T}_2, \mathcal{A}_2 \rangle$. Therefore, we have that

$$\mathsf{ans}(Q_2, \mathcal{K}_2) = \{\mathsf{FSM}, \mathsf{God}, \mathsf{Zeus}\}$$

The set $\mathsf{ans}(Q_2, \Xi(\mathcal{K}_2) \cup E'_{\mathcal{T}_2})$ is guaranteed to contain the representative of FSM, God, and Zeus. Therefore, in order to compute $\mathsf{ans}(Q_2, \mathcal{K}_2)$, we only need to substitute each tuple $\langle a \rangle$ in $\mathsf{ans}(Q_2, \Xi(\mathcal{K}_2) \cup E'_{\mathcal{T}_2})$ with the corresponding r_a. It can be shown that $\mathsf{ans}(Q_2, \Xi(\mathcal{K}_2) \cup E'_{\mathcal{T}_2}) = \{\mathsf{God}, \mathsf{Zeus}\}$ and that

$$\{\langle \mathsf{FSM}, \mathsf{God} \rangle, \langle \mathsf{God}, \mathsf{Zeus} \rangle\} \subseteq \mathsf{ans}(Q_{\mathrm{eq}}(x, y) \leftarrow x \approx y, \Xi(\mathcal{K}_2) \cup E'_{\mathcal{T}_2})$$

Therefore, we have that $r_{\mathsf{God}} = r_{\mathsf{Zeus}} = \{\mathsf{God}, \mathsf{FSM}, \mathsf{Zeus}\}$. Clearly, substituting each tuple $\langle a \rangle$ in $\mathsf{ans}(Q_2, \Xi(\mathcal{K}_2) \cup E'_{\mathcal{T}_2})$ with the corresponding r_a yields the set $\{\mathsf{FSM}, \mathsf{God}, \mathsf{Zeus}\}$.

Note that we derived the set $\mathsf{ans}(Q_2, \mathcal{K}_2)$ from $\mathsf{ans}(Q_2, \Xi(\mathcal{K}_2) \cup E'_{\mathcal{T}_2})$ *without knowing* what constant is the actual representative of FSM, God, and Zeus; in fact, we did not need an order $\prec_{\mathcal{K}_2}$ either. The representatives and the associated order are used only to prove that it is possible to compute $\mathsf{ans}(Q_2, \mathcal{K}_2)$ from $\mathsf{ans}(Q_2, \Xi(\mathcal{K}_2) \cup E'_{\mathcal{T}_2})$; the potential implementors of our answering algorithm do not need to worry about the representatives whatsoever.

In the rest of the section we show various properties required to prove the correctness of our approach of deriving $\mathsf{ans}(Q, \mathcal{K})$ from $\mathsf{ans}(Q, \Xi(\mathcal{K}) \cup E'_{\mathcal{T}})$. We start by showing that $E'_{\mathcal{T}}$ ensures the "propagation of information" from every term s to its representative $[s]_I$ (cf. Lemma 4): we show that $A(s) \in I$ implies $A([s]_I) \in I$, and $P(s, t) \in I$ implies $P([s]_I, [t]_I) \in I$. In order to do so, we first prove a property of the binary atoms occurring in the minimal Herbrand model of $\Xi(\mathcal{K}) \cup E'_{\mathcal{T}}$.

Lemma 1. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be an $\mathcal{ELHIO}$ KB, and let I be the minimal Herbrand model of $\Xi(\mathcal{K}) \cup E'_{\mathcal{T}}$. Every binary atom D occurring in I is of the form (i) $P(s, f(s))$, (ii) $P(f(s), s)$, (iii) $P(s, o)$, or (iv) $P(o, s)$.*

Proof. If $D \in \mathcal{A}$, then it is of form (iii) or (iv). Otherwise, by analyzing the type of clauses with binary head in Table 2, it can be readily checked that if D was derived through one of such clauses, then it is of the form (i) or (ii). By analyzing the binary substitutivity$_{\mathcal{O}}$ clauses of $E'_{\mathcal{T}}$, it can be checked that if D was derived through one of such clauses, then it is of the form (iii) or (iv). There is no other type of clause with binary head in $\Xi(\mathcal{K}) \cup E'_{\mathcal{T}}$. The claim follows from these facts and that fact that I is minimal. $\square$

We now prove two properties of the representatives.

Lemma 2. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be an $\mathcal{ELHIO}$ KB, and let I be the minimal Herbrand model of $\Xi(\mathcal{K}) \cup E'_{\mathcal{T}}$. For every term s such that $s \neq \mathsf{min}(s)$, there exist $\mathcal{O}$-constants $o_1, ..., o_n$ such that $\{s \approx o_1, o_1 \approx o_2, ..., o_n \approx \mathsf{min}(s)\} \subseteq I$.*

Proof. We first show that, for all $\mathcal{O}$-constants o and o', if $t \approx o \in I$ and $t \approx o' \in I$ for some term t, then $o \approx o' \in I$. Since I is minimal, the assertion $t \approx o'$ was derived by a clause $x \approx o' \leftarrow A(x)$ and a fact $A(t) \in I$ for some predicate A. We have $t \approx o \in I$; moreover, since o is an $\mathcal{O}$-constant, we have that $\mathcal{O}(o) \in I$. Therefore, the unary substitutivity$_{\mathcal{O}}$ clauses in $E'_{\mathcal{T}}$ then ensures $A(o) \in I$, which implies $o \approx o' \in I$.

Now, since $s \neq \mathsf{min}(s)$, by the definition of $\mathsf{min}(s)$, there exist terms $u_1, ..., u_n$ such that $\Gamma \subseteq I$ for $\Gamma = \{s \approx u_1, u_1 \approx u_2, ..., u_n \approx \mathsf{min}(s)\}$. Given the type of clauses contained in $\Xi(\mathcal{K}) \cup E'_{\mathcal{T}}$, every equality assertion in I is of the form $t \approx o$, where t is a term and o is an $\mathcal{O}$-constant. Assume now that some term u_i is not an $\mathcal{O}$-constant. Then, Γ contains equalities of the form $u_i \approx o$ and $u_i \approx o'$. By the previous paragraph, I then contains $o \approx o'$; hence, u_i can be eliminated from Γ. By successively eliminating all u_j that are not $\mathcal{O}$-constants, we obtain a subset of the equalities of I that satisfy our claim. $\square$

To show the next property we need a notion of a *derivation tree*, as defined next.

Definition 4. *A tree T is a prefix-closed subset of $\mathbb{N}^*$, where the root node is denoted by ϵ, and the i-th child of a node $t \in T$ is denoted by $t.i$.*

Let N be a set of Horn clauses and let I be a Herbrand model of N. A derivation tree Σ for a fact $l \in I$ is a triple $\langle T, \delta, \lambda \rangle$ where T is a finite tree, δ is a function that maps every node $t \in T$ to a fact $\delta(t) \in I$, and λ is a partial function that maps every nonleaf node $t \in T$ with n children to a clause $\lambda(t) \in N$ such that $\delta(t)$ is obtained by resolving each $\delta(t.i)$ with the i-th body literal of $\lambda(t)$ simultaneously. If t is a leaf node, then $\delta(t) \in N$ and $\lambda(t)$ is undefined.

We are ready to show the next property.

Lemma 3. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be an $\mathcal{ELHIO}$ KB, and let I be the minimal Herbrand model of $\Xi(\mathcal{K}) \cup E'_{\mathcal{T}}$. For every term s, every constant o, all predicates A and P, and every $n \geq 0$, we have that*

(i) $A(f_1...f_n(s)) \in I$ implies $A(f_1...f_n([s]_I)) \in I$,
(ii) $P(f_1...f_n(s), f_0 f_1...f_n(s)) \in I$ implies $P(f_1...f_n([s]_I), f_0 f_1...f_n([s]_I)) \in I$,
(iii) $P(f_1...f_n(s), f_2...f_n(s)) \in I$ implies $P(f_1...f_n([s]_I), f_2...f_n([s]_I)) \in I$,
(iv) $P(f_1...f_n(s), o) \in I$ implies $P(f_1...f_n([s]_I), [o]_I) \in I$,
(v) $P(o, f_1...f_n(s)) \in I$ implies $P([o]_I, f_1...f_n([s]_I)) \in I$, and
(vi) $f_1...f_n(s) \approx o$ implies $f_1...f_n([s]_I) \approx o$.

Proof. Let D be an atom of one of the following forms:

$$A(f_1...f_n(s)) \tag{22}$$

$$P(f_1...f_n(s), f_0 f_1...f_n(s)) \tag{23}$$

$$P(f_1...f_n(s), f_2...f_n(s)) \tag{24}$$

$$P(f_1...f_n(s), o) \tag{25}$$

$$P(o, f_1...f_n(s)) \tag{26}$$

$$f_1...f_n(s) \approx o \tag{27}$$

We prove the claim by induction on the height h of the derivation tree of D.

The base case is $h = 0$. Since I is minimal, D is of the form $A(a)$ or $P(a, b)$. We consider the case where D is of the form $P(a, b)$; the other case can be proved analogously. By the definition of $[\cdot]_I$, we have that $[a]_I = \min(a)$ and $[b]_I = \min(b)$. If $a = \min(a)$ and $b = \min(b)$, then the claim trivially follows. We consider the case when $a \neq \min(a)$ and $b \neq \min(b)$; the case where $a \neq \min(a)$ and $b = \min(b)$, and vice versa can be shown analogously. By Lemma 2, $\mathcal{O}$-constants $o_1, ..., o_n$ and $o'_1, ..., o'_m$ exist such that $\{a \approx o_1, o_1 \approx o_2, ..., o_n \approx \min(a)\} \subseteq I$ and $\{b \approx o'_1, o'_1 \approx o'_2, ..., o'_m \approx \min(b)\} \subseteq I$. Due to the substitutivity$_{\mathcal{O}}$ clauses in $E'_{\mathcal{T}}$, we have $P(o_i, b) \in I$ for $1 \leq i \leq n$; since $o_n \approx \min(a)$, we have $P(\min(a), b) \in I$. In an analogous way, we have $P(\min(a), o'_j) \in I$ for $1 \leq j \leq m$. Hence, since $o'_m \approx \min(b)$, we have that $P(\min(a), \min(b)) \in I$.

We now assume the claim holds for h and we prove the claim for $h + 1$. We first consider the case where D is of the form (22). Given the clauses contained in $\Xi(\mathcal{K}) \cup E'_{\mathcal{T}}$, we have that D could have been derived using a clause C of one the following forms:

$$A(x) \leftarrow B(x) \tag{28}$$
$$A(x) \leftarrow B(x) \wedge C(x) \tag{29}$$
$$A(x) \leftarrow P(x, y) \wedge B(y) \tag{30}$$
$$A(x) \leftarrow P(y, x) \wedge B(y) \tag{31}$$
$$A(y) \leftarrow A(x) \wedge x \approx y \wedge \mathcal{O}(y) \tag{32}$$
$$A(f(x)) \leftarrow B(x) \tag{33}$$

- If C is of the form (28) or (29), then by the induction hypothesis we have that the claim holds for every antecedent atom required to apply C, which implies that $A(f_1...f_n([s]_I)) \in I$.
- If C is of the form (30), then $\{P(f_1...f_n(s), t), B(t)\} \subseteq I$ for some term t. Lemma 1 implies that the term t can be of the form o, $f_0 f_1...f_n(s)$, or $f_2...f_n(s)$. In all cases, by the induction hypothesis we have that the claim holds for every antecedent atom required to apply C, which implies that $A(f_1...f_n([s]_I)) \in I$. The proof is analogous if C is of the form (31).
- If C is of the form (32), then we have that D is of the form $A(o)$ for some $\mathcal{O}$-constant o. By the definition of $[\cdot]_I$, we have that $[o]_I = \mathsf{min}(o)$. If $o = \mathsf{min}(o)$, then the claim trivially follows, so we assume that $o \neq \mathsf{min}(o)$. Given Lemma 2 and the substitutivity$_\mathcal{O}$ clauses of $E'_{\mathcal{T}}$, we have that $A(\mathsf{min}(o)) \in I$.
- If C is of the form (33), then we consider two cases: $n > 0$ and $n = 0$. In the former case, we have that $B(f_2...f_n(s)) \in I$. By the induction hypothesis we have $B(f_2...f_n([s]_I)) \in I$, so $A(f_1...f_n([s]_I)) \in I$. For $n = 0$, note that, given the form of C, s is of the form $s = f(s')$ and $B(s') \in I$. We consider the possible forms of $[f(s')]_I$.
  - If an $\mathcal{O}$-constant o exists such that $f(s') \approx o \in I$, then $[f(s')]_I = \mathsf{min}(o)$. Given Lemma 2 and the substitutivity$_\mathcal{O}$ clauses of $E'_{\mathcal{T}}$, we have that $A(\mathsf{min}(o)) \in I$ since $A(f(s')) \in I$.
  - Otherwise, we have $[f(s')]_I = \mathsf{min}(f([s']_I))$. If there is an $\mathcal{O}$-constant o such that $f([s']_I) \approx o \in I$, then $[f(s')]_I = \mathsf{min}(f([s']_I)) = \mathsf{min}(o)$. By the induction hypothesis we have that $B([s']_I) \in I$, so $A(f([s']_I)) \in I$. Therefore, it follows from Lemma 2 and the substitutivity$_\mathcal{O}$ clauses of $E'_{\mathcal{T}}$ that $A(\mathsf{min}(o)) \in I$. Finally, if there is no $\mathcal{O}$-constant o such that $f([s']_I) \approx o \in I$, we have that $[f(s')]_I = \mathsf{min}(f([s']_I)) = f([s']_I)$ and the claim follows since $A(f([s']_I)) \in I$.

We now consider the case where D is of the form (23)–(26). The atom D could have been derived using a clause C of one the following forms:

$$P(x,y) \leftarrow S(x,y) \tag{34}$$
$$P(x,y) \leftarrow S(y,x) \tag{35}$$
$$P(x,z) \leftarrow P(x,y) \wedge y \approx z \wedge \mathcal{O}(z) \tag{36}$$
$$P(z,x) \leftarrow P(y,x) \wedge y \approx z \wedge \mathcal{O}(z) \tag{37}$$
$$P(x,f(x)) \leftarrow A(x) \tag{38}$$
$$P(f(x),x) \leftarrow A(x) \tag{39}$$

- If C is of the form (34) or (35), then by the induction hypothesis we have that the claim holds for every antecedent atom required to apply C, which implies that the claim holds for all possible forms of D.
- If C is of the form (36), then D is of the form (25). Clearly, in order for C to be applied, we have that $\{P(f_1...f_n(s),t), t \approx o, \mathcal{O}(o)\} \subseteq I$ for some term t. Lemma 1 implies that the term t can be of the form o', $f_0 f_1...f_n(s)$, or $f_2...f_n(s)$. In all cases, by the induction hypothesis the claim holds for $P(f_1...f_n(s),t)$ and $t \approx o$, which is enough to derive $P(f_1...f_n([s]_I),o)$ using C. By the definition of $[\cdot]_I$ we have that $[o]_I = \min(o)$. If $o = \min(o)$, then the claim trivially follows, so we assume that $o \neq \min(o)$. By Lemma 2 and the substitutivity$_\mathcal{O}$ clauses of $E'_\mathcal{T}$, we have that $P(f_1...f_n([s]_I), \min(o)) \in I$; the claim can be shown analogously if C is of the form (37).
- If C is of the form (38), then D is of the form (23) and $A(f_1...f_n(s)) \in I$. By the induction hypothesis we have that $A(f_1...f_n([s]_I)) \in I$; therefore, we have that $P(f_1...f_n([s]_I), f_0 f_1...f_n([s]_I)) \in I$; the claim can be shown analogously if C is of the form (39).

Finally, we consider the case where D is of the form (27). The atom D could have been derived using a clause C of the following form:

$$x \approx o \leftarrow A(x) \tag{40}$$

We have that $A(f_1...f_n(s)) \in I$. We have that $A(f_1...f_n([s]_I)) \in I$ by the induction hypothesis; therefore, $f_1...f_n([s]_I) \approx o \in I$. $\qquad \square$

We are now ready to show the desired relationship between a term s and its representative $[s]_I$.

Lemma 4. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be an $\mathcal{ELHIO}$ KB, and let I be the minimal Herbrand model of $\Xi(\mathcal{K}) \cup E'_\mathcal{T}$. For all terms s and t, and all predicates A and P occurring in I, we have that*

(i) $A(s) \in I$ implies $A([s]_I) \in I$, and
(ii) $P(s,t) \in I$ implies $P([s]_I, [t]_I) \in I$.

Proof. Claim (i) follows from Lemma 3. We now consider claim (ii). Let D be an atom of the form $P(s,t)$. In case D is of the form $P(a,b)$, $P(s,a)$, or $P(a,s)$, the

claim follows from Lemma 3. We now consider the case where D is of the form $P(s, f(s))$; the case where D is of the form $P(f(s), s)$ can be shown analogously. We analyze the possible forms of $[f(s)]_I$.

- If there is an $\mathcal{O}$-constant o such that $f(s) \approx o \in I$, then $[f(s)]_I = \min(o)$. By Lemma 3 we have that $f([s]_I) \approx o \in I$ and $P([s]_I, f([s]_I)) \in I$. By Lemma 2 and the substitutivity$_\mathcal{O}$ clauses of $E'_\mathcal{T}$ we have that $P([s]_I, \min(o)) \in I$.
- Otherwise, we have that $[f(s)]_I = \min(f([s]_I))$. If there is an $\mathcal{O}$-constant o such that $f([s]_I) \approx o \in I$, then $[f(s)]_I = \min(f([s]_I)) = \min(o)$, and the claim follows again by Lemma 2 and the substitutivity$_\mathcal{O}$ clauses of $E'_\mathcal{T}$. Finally, if there is no $\mathcal{O}$-constant o such that $f([s]_I) \approx o \in I$, we have that $[f(s)]_I = \min(f([s]_I)) = f([s]_I)$, and the claim follows since we have that $P([s]_I, f([s]_I)) \in I$.

$\square$

We now show that $[\cdot]_I$ "compensates" for the loss of functional monotonicity (cf. Lemma 5): we show that $[s]_I = [t]_I$ implies $[f(s)]_I = [f(t)]_I$. We point out that in order to do so, it suffices to show that $[f(s)]_I = [f([s]_I)]_I$. We illustrate the point with an example.

Example 4. Let us assume that (*) for every term s and every f we have that $[f(s)]_I = [f([s]_I)]_I$. Now, suppose that

$$[\mathsf{FSM}]_I = [\mathsf{Zeus}]_I = \mathsf{God}.$$

Let religionOf be a function symbol. By (*) we have that

$$[\text{religionOf}(\mathsf{FSM})]_I = [\text{religionOf}(\mathsf{God})]_I,$$
$$[\text{religionOf}(\mathsf{Zeus})]_I = [\text{religionOf}(\mathsf{God})]_I.$$

Therefore, it can be seen that if we assume (*) we have that $[\mathsf{FSM}]_I = [\mathsf{Zeus}]_I$ implies $[\text{religionOf}(\mathsf{FSM})]_I = [\text{religionOf}(\mathsf{Zeus})]_I$.

Lemma 5. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be an $\mathcal{ELHIO}$ KB, and let I be the minimal Herbrand model of $\Xi(\mathcal{K}) \cup E'_\mathcal{T}$. For every term s and every function symbol f, we have that $[f(s)]_I = [f([s]_I)]_I$.*

Proof. We show the claim by analyzing the possible forms of $[f(s)]_I$.

- If there is an $\mathcal{O}$-constant o such that $f(s) \approx o \in I$, then $[f(s)]_I = \min(o)$. By Lemma 3 we have that $f([s]_I) \approx o \in I$; therefore, $[f([s]_I)]_I = \min(o)$.
- Otherwise, we have that $[f(s)]_I = \min(f([s]_I))$. If there is an $\mathcal{O}$-constant o such that $f([s]_I) \approx o \in I$, then $\min(f([s]_I)) = \min(o)$ and $[f([s]_I)]_I = \min(o)$. Finally, if there is no $\mathcal{O}$-constant o such that $f([s]_I) \approx o \in I$, then we have that $\min(f([s]_I)) = f([s]_I)$ and $[f([s]_I)]_I = f([s]_I)$.

$\square$

We now concentrate on showing the crucial fact that $[\vec{a}]_I \in \mathsf{ans}(Q, \mathcal{K})$ iff $[\vec{a}]_I \in \mathsf{ans}(Q, \Xi(\mathcal{K}) \cup E'_{\mathcal{T}})$ (cf. Lemma 9). Note that, in order to show this claim, it suffices to show that $\Xi(\mathcal{K}) \cup E_{\mathcal{T}} \models D([\vec{s}]_I)$ iff $\Xi(\mathcal{K}) \cup E'_{\mathcal{T}} \models D([\vec{s}]_I)$ for every atom $D([\vec{s}]_I)$. Moreover, since both sets—$\Xi(\mathcal{K}) \cup E_{\mathcal{T}}$ and $\Xi(\mathcal{K}) \cup E'_{\mathcal{T}}$—are sets of Horn clauses, we can simply show that $D([\vec{s}]_I) \in J$ iff $D([\vec{s}]_I) \in I$, where J is the minimal Herbrand model of $\Xi(\mathcal{K}) \cup E_{\mathcal{T}}$, and I is the minimal Herbrand model of $\Xi(\mathcal{K}) \cup E'_{\mathcal{T}}$. We show this by constructing an interpretation I_∞ from I such that $D([\vec{s}]_I) \in I_\infty$ iff $D([\vec{s}]_I) \in I$, and proving that I_∞ is the minimal Herbrand model of $\Xi(\mathcal{K}) \cup E_{\mathcal{T}}$.

Intuitively, we obtain I_∞ from I by adding to I the set of facts needed to make $\approx$ a *congruence relation*: we propagate every logical consequence for $[s]_I$ to all the terms represented by $[s]_I$; and we ensure that $\approx$ is a transitive, symmetric and reflexive relation in I_∞, that conforms to functional monotonicity. We formally define I_∞ as follows.

Definition 5. *Let I be the minimal Herbrand model of $\Xi(\mathcal{K}) \cup E'_{\mathcal{T}}$. Let $\mathcal{O}_I$ be the set of exactly all the atoms of the form $\mathcal{O}(o)$ contained in I.*

We construct I_∞ as follows. Let I_0 be the smallest set of facts that satisfy the following conditions for all predicates A and P, all terms s and t, and every constant o:

(i) $I \setminus \mathcal{O}_I \subseteq I_0$;
(ii) if $A(s) \in I$, then $A(s') \in I_0$ for every term s' such that $[s']_I = s$;
(iii) if $P(s,t) \in I$, then $P(s',t') \in I_0$ for all terms s' and t' such that $[s']_I = s$ and $[t']_I = t$; and
(iv) if $s \approx o \in I$, then $s' \approx o \in I_0$ for every term s' such that $[s']_I = s$.

Let I_i, for $1 \le i \le \infty$, be the smallest set of facts that satisfy the following conditions for all terms s, t, and u, and every functional symbol f:

(v) $I_{i-1} \subseteq I_i$;
(vi) $s \approx s \in I_i$;
(vii) if $s \approx t \in I_i$, then $t \approx s \in I_i$;
(viii) if $s \approx u \in I_i$ and $u \approx t \in I_i$, then $s \approx t \in I_i$; and
(ix) if $s \approx t \in I_i$, then $f(s) \approx f(t) \in I_i$.

Let $I_\infty = \bigcup I_i$.

We now show that I_∞ is the minimal Herbrand model of $\Xi(\mathcal{K}) \cup E_{\mathcal{T}}$. We do so in two steps: we first show that I_∞ is a model of $\Xi(\mathcal{K}) \cup E_{\mathcal{T}}$ (cf. Lemma 7), and then show that I_∞ is minimal (cf. Lemma 8). In order to do so, we first show an important property of $[\cdot]_I$ in Lemma 6.

Lemma 6. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be an $\mathcal{ELHIO}$ KB, let I be the minimal Herbrand model of $\Xi(\mathcal{K}) \cup E'_{\mathcal{T}}$. For all terms s and t, we have that $s \approx t \in I_\infty$ iff $[s]_I = [t]_I$.*

Proof. We first prove that $s \approx t \in I_\infty$ implies $[s]_I = [t]_I$. We prove the claim by induction on the level i in the construction of I_i for $0 \le i \le \infty$. In case

$i = 0$, it follows from the definition of I_0 that either $s \approx t \in I$ or $[s]_I \approx t \in I$. In both cases, the claim follows from the definition of $[\cdot]_I$. We now assume that the claim holds for i and prove the claim for $i + 1$. The atom $s \approx t$ was derived due to conditions (vi)–(ix). The claim trivially follows for condition (vi). For condition (vii) we have that $t \approx s \in I_i$. By the induction hypothesis we have that $[t]_I = [s]_I$, so the claim holds. For condition (viii) we have that there is a term u such that $s \approx u \in I_i$ and $u \approx t \in I_i$. By the induction hypothesis we have that $[s]_I = [t]_I = [u]_I$; so the claim holds. We now consider condition (ix). Let s be of the form $f(s')$ and t be of the form $f(t')$ for some function symbol f. We have that $s' \approx t' \in I_i$. By the induction hypothesis we have that $[s']_I = [t']_I$, and by Lemma 5, we have that $[f(s')]_I = [f(t')]_I$. We have shown that $s \approx t \in I_i$ implies $[s]_I = [t]_I$ for any $0 \leq i \leq \infty$. The claim for I_∞ follows from the fact that $s \approx t \in I_\infty$ implies that there is an i such that $s \approx t \in I_i$.

We now prove that $[s]_I = [t]_I$ implies $s \approx t \in I_\infty$. It follows from the definition of $[\cdot]_I$ and Lemma 5 that there are three cases for which $[s]_I = [t]_I$: (I) $s = t$; (II) there are terms $u_1, ..., u_m$ such that $\{s \approx u_1, u_1 \approx u_2, ..., u_m \approx t\} \subseteq I$; or (III) s is of the form $f(s')$, the term t is of the form $f(t')$, and $[s']_I = [t']_I$. Let $s = f_1...f_n(s'')$ and $t = f_1...f_n(t'')$ for the longest possible common prefix $f_1...f_n$ of s and t. We prove the claim by induction on n. If $n = 0$, then only cases (I) and (II) apply. The claim follows from conditions (vi) and (viii), respectively, of the definition of I_∞. We now assume that the claim holds for n and prove the claim for $n + 1$. In cases (I) and (II), the claim follows again from conditions (vi) and (viii), respectively. In case (III) we have that s is of the form $f_1...f_{n+1}(s'')$, the term t is of the form $f_1...f_{n+1}(t'')$, and $[f_2...f_{n+1}(s'')]_I = [f_2...f_{n+1}(t'')]_I$. By the induction hypothesis we have that $f_2...f_{n+1}(s'') \approx f_2...f_{n+1}(t'') \in I_\infty$. Therefore, there is an i such that $f_2...f_{n+1}(s'') \approx f_2...f_{n+1}(t'') \in I_i$ and the claim follows due to condition (ix) of the construction of I_∞.   □

We are ready to show that I_∞ is a model of $\Xi(\mathcal{K}) \cup E_\mathcal{T}$.

Lemma 7. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be an $\mathcal{ELHIO}$ KB, let I be the minimal Herbrand model of $\Xi(\mathcal{K}) \cup E'_\mathcal{T}$. Then, I_∞ is a Herbrand model of $\Xi(\mathcal{K}) \cup E_\mathcal{T}$.*

Proof. We first show that I_0 is a model of $\Xi(\mathcal{K})$. The set $\Xi(\mathcal{K})$ contains clauses of the form (28)–(31), (33)–(35), and (38)–(40).

We now show that I_0 is a model of clauses of form (30); the proof is analogous for clauses of form (28), (29), (31), (34), and (35). We show that if $P(s, t) \in I_0$ and $B(t) \in I_0$, then $A(t) \in I_0$. If we have that $P(s, t) \in I$ and $B(t) \in I$, then $A(t) \in I$ because I is a model of $\Xi(\mathcal{K})$. Therefore, $A(t) \in I_0$ since $I \setminus \mathcal{O}_I \subseteq I_0$. Otherwise, we first assume that $P(s, t) \notin I$ and $B(t) \in I$. It follows from the definition of I_0 that $P([s]_I, [t]_I) \in I$; moreover, Lemma 4 implies that $B([t]_I) \in I$. Therefore, we have that $A([t]_I) \in I$ and according to condition (ii) of the definition of I_0, we have that $A(t) \in I_0$; the claim can be shown analogously for the case where $P(s, t) \in I$ and $B(t) \notin I$. Finally, we assume that $P(s, t) \notin I$ and $B(t) \notin I$. It follows from the definition of I_0 that $P([s]_I, [t]_I) \in I$ and $B([t]_I) \in I$; therefore, we have that $A([t]_I) \in I$ and according to condition (ii) of the definition of I_0, we have that $A(t) \in I_0$.

We now show that I_0 is a model of clauses of form (38). We show that if $A(s) \in I_0$, then $P(s, f(s)) \in I_0$. If we have that $A(s) \in I$, then $P(s, f(s)) \in I$ because I is a model of $\Xi(\mathcal{K})$. Therefore, $P(s, f(s)) \in I_0$, since $I \setminus \mathcal{O}_I \subseteq I_0$. Otherwise, it follows from the definition of I_0 that $A([s]_I) \in I$; therefore, we have that $P([s]_I, f([s]_I)) \in I$. By Lemma 4, we have that $P([[s]_I]_I, [f([s]_I)]_I) \in I$. The function $[\cdot]_I$ is idempotent by definition, so we have $P([s]_I, [f([s]_I)]_I) \in I$; moreover, by Lemma 5, we have that $[f([s]_I)]_I = [f(s)]_I$, so $P([s]_I, [f(s)]_I) \in I$. Now, according to condition (iii) of the definition of I_0, we have that $P(s, f(s)) \in I_0$; the proof is analogous for clauses of form (33) and (39).

We finally show that I_0 is a model of clauses of form (40). We show that if $A(s) \in I_0$, then $s \approx o \in I_0$. If we have that $A(s) \in I$, then $s \approx o \in I$ because I is a model of $\Xi(\mathcal{K})$. Therefore, $s \approx o \in I_0$, since $I \setminus \mathcal{O}_I \subseteq I_0$. Otherwise, it follows from the definition of I_0 that $A([s]_I) \in I$; therefore, we have that $[s]_I \approx o \in I$. Now, according to condition (iv) of the definition of I_0, we have that $s \approx o \in I_0$.

Since I_0 is a model of $\Xi(\mathcal{K})$, we have that I_∞ is also a model of $\Xi(\mathcal{K})$ since $I_0 \subseteq I_\infty$, only atoms of the form $s \approx t$ are added to I_i for every $i > 0$, and no clause in $\Xi(\mathcal{K})$ contains the predicate $\approx$ in the body.

We now show that I_∞ is a model of $E_\mathcal{T}$. It follows from the definition of I_i for $i > 0$ that I_∞ is a model of the reflexivity, symmetry, transitivity, and functional monotonicity clauses of $E_\mathcal{T}$. Therefore, we need only consider the substitutivity clauses of $E_\mathcal{T}$. We consider binary substitutivity 1. We show that if $P(s, t) \in I_\infty$ and $t \approx u \in I_\infty$, then $P(s, u) \in I_\infty$. Since no atom of the form $P(s, t)$ is added to I_∞ in the construction of I_i for $i > 0$, we have that $P(s, t) \in I_0$; moreover, it follows from the definition of I_0 that either $P(s, t) \in I$ or $P(s, t) \notin I$. In the former case, by Lemma 4, we have $P([s]_I, [t]_I) \in I$; and in the latter case it follows from the definition of I_0 that $P([s]_I, [t]_I) \in I$. Since $t \approx u \in I_\infty$, Lemma 6 implies that $[t]_I = [u]_I$. Therefore, according to condition (iii) of the definition of I_0 we have that $P(s, u) \in I_0$, which implies that $P(s, u) \in I_\infty$ since $I_0 \subseteq I_\infty$; the claim can be shown analogously for other substitutivity clauses.   □

We are ready to show that I_∞ is minimal.

Lemma 8. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be an $\mathcal{ELHIO}$ KB, let I be the minimal Herbrand model of $\Xi(\mathcal{K}) \cup E'_\mathcal{T}$. We have that every fact $D \in I_\infty$ is derivable from $\Xi(\mathcal{K}) \cup E_\mathcal{T}$.*

Proof. Note that D is of the form $s \approx t$, $A(s)$ or $P(s, t)$. We first consider the case where $D \in I$. Let $\Sigma = \langle T, \delta, \lambda \rangle$ be the derivation tree of D. We transform Σ into a new derivation tree Σ' as follows: for every node $n \in T$, if $\delta(n)$ is of the form $\mathcal{O}(o)$, then remove n from T; and if $\lambda(n)$ contains an atom of the form $\mathcal{O}(y)$, then remove such an atom from $\lambda(n)$. Given the substitutivity clauses contained in $E_\mathcal{T}$ it follows that Σ' is a derivation tree for D such that for every node $n \in T'$ we have that $\lambda(n) \in \Xi(\mathcal{K}) \cup E_\mathcal{T}$. Therefore, D is derivable from $\Xi(\mathcal{K}) \cup E_\mathcal{T}$.

We now consider the case where $D \notin I$. In case D is of the form $s \approx t$, we have that it was added to I_∞ due to condition (iv) of the definition of I_0, or conditions (vi)–(ix) of the definition of I_i for $i > 0$. For condition (iv), it follows from the definition of I_0 that $[s]_I \approx t \in I$; and since $I \setminus \mathcal{O}_I \subseteq I_\infty$, we have $[s]_I \approx t \in I_\infty$.

Since clearly $[s]_I = [[s]_I]_I$, Lemma 6 implies that $s \approx [s]_I \in I_\infty$. Therefore, the atom $s \approx t$ is derivable from $\Xi(\mathcal{K}) \cup E_{\mathcal{T}}$ given the transitivity clause of $E_{\mathcal{T}}$. For conditions (vi)–(ix), it is easy to see that $s \approx t$ is derivable from $\Xi(\mathcal{K}) \cup E_{\mathcal{T}}$ given the reflexivity, symmetry, transitivity, and functional monotonicity clauses of $E_{\mathcal{T}}$, respectively. We now assume that D is of the form $P(s,t)$; the unary case can be shown analogously. Since no atom of the form $P(s,t)$ is added to I_∞ in the construction of I_i for $i > 0$, we have that $P(s,t) \in I_0$; moreover, it follows from the definition of I_0 that either $P(s,t) \in I$ or $P(s,t) \notin I$. In the former case, by Lemma 4, we have $P([s]_I, [t]_I) \in I$; and in the latter case it follows from the definition of I_0 that $P([s]_I, [t]_I) \in I$. As already shown, there is a derivation tree Σ' for $P([s]_I, [t]_I)$. Since clearly $[s]_I = [[s]_I]_I$ and $[t]_I = [[t]_I]_I$, Lemma 6 implies that $s \approx [s]_I \in I_\infty$ and $t \approx [t]_I \in I_\infty$. We obtain a derivation tree Σ'' for $P(s,t)$ from Σ' by applying the binary substitutivity clauses of $E_{\mathcal{T}}$ twice. Therefore, $P(s,t)$ is derivable from $\Xi(\mathcal{K}) \cup E_{\mathcal{T}}$. □

By Lemma 7 we have that I_∞ is a Herbrand model of $\Xi(\mathcal{K}) \cup E_{\mathcal{T}}$, and by Lemma 8 we have that every fact $D \in I_\infty$ is derivable from $\Xi(\mathcal{K}) \cup E_{\mathcal{T}}$. Therefore, I_∞ is the minimal Herbrand model of $\Xi(\mathcal{K}) \cup E_{\mathcal{T}}$. Lemma 9 follows from this fact, since by definition of I_∞, we have that $D([\vec{s}]_I) \in I_\infty$ iff $D([\vec{s}]_I) \in I$.

Lemma 9. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be an $\mathcal{ELHIO}$ KB, let I be the minimal Herbrand model of $\Xi(\mathcal{K}) \cup E'_{\mathcal{T}}$, and let $Q = \langle Q_P, Q_C \rangle$ be a conjunctive query. We have that $[\vec{a}]_I \in \mathsf{ans}(Q, \mathcal{K})$ iff $[\vec{a}]_I \in \mathsf{ans}(Q, \Xi(\mathcal{K}) \cup E'_{\mathcal{T}})$.*

Lemma 9 implies that $\mathsf{ans}(Q, \Xi(\mathcal{K}) \cup E'_{\mathcal{T}}) \subseteq \mathsf{ans}(Q, \mathcal{K})$. As previously mentioned, in order to compute the whole set $\mathsf{ans}(Q, \mathcal{K})$, one needs to expand $\mathsf{ans}(Q, \Xi(\mathcal{K}) \cup E'_{\mathcal{T}})$ by expanding every tuple $\vec{a}$ into all tuples of terms $\vec{b}$ such that each a_i is equal to the corresponding b_i. We formalize this process as follows.

Definition 6. *Let $\mathsf{ans}(Q, \Xi(\mathcal{K}) \cup E'_{\mathcal{T}}) = \{\langle a_{1,1}, ..., a_{1,m} \rangle, ..., \langle a_{n,1}, ..., a_{n,m} \rangle\}$ for a conjunctive query Q and an $\mathcal{ELHIO}$ KB $\mathcal{K}$. With $\mathsf{ans}'(Q, \Xi(\mathcal{K}) \cup E'_{\mathcal{T}})$ we denote the set $\bigcup_{i=1}^{n} \times_{j=1}^{m} r_{i,j}$, where $r_{i,j}$ is the set containing $a_{i,j}$ and every constant to which there is a path from $a_{i,j}$ in $\mathsf{ans}(Q_{eq}(x,y) \leftarrow x \approx y, \Xi(\mathcal{K}) \cup E'_{\mathcal{T}})$.*

By Lemma 9 we have that $[\vec{a}]_I \in \mathsf{ans}(Q, \mathcal{K})$ iff $[\vec{a}]_I \in \mathsf{ans}(Q, \Xi(\mathcal{K}) \cup E'_{\mathcal{T}})$. Moreover, it follows from the definition of $[\cdot]_I$ and Lemma 2 that if there is a path from a to b in the relation $\mathsf{ans}(Q_{\mathrm{eq}}(x,y) \leftarrow x \approx y, \Xi(\mathcal{K}) \cup E'_{\mathcal{T}})$, then $[a]_I = [b]_I$. Furthermore, by Lemma 6 we have that $[a]_I = [b]_I$ implies $a \approx b \in I_\infty$. This implies Lemma 10 since, by Lemma 7 and Lemma 8, we have that I_∞ is the minimal Herbrand model of $\Xi(\mathcal{K}) \cup E_{\mathcal{T}}$.

Lemma 10. *For $\mathcal{K}$ an $\mathcal{ELHIO}$ KB, and $Q = \langle Q_P, Q_C \rangle$ a conjunctive query, we have that $\mathsf{ans}(Q, \mathcal{K}) = \mathsf{ans}'(Q, \Xi(\mathcal{K}) \cup E'_{\mathcal{T}})$.*

4.2   Computing $\mathsf{ans}(Q, \Xi(\mathcal{K}) \cup E'_{\mathcal{T}})$ using Resolution

As mentioned in Section 3, it follows from the definition of certain answers that a tuple $\vec{a}$ is a certain answer to a conjunctive query $Q = \langle Q_P, Q_C \rangle$ over $\Xi(\mathcal{K}) \cup E'_{\mathcal{T}}$

iff the logic program $\Xi(\mathcal{K}) \cup E'_{\mathcal{T}} \cup Q_C \cup \{\bot \leftarrow Q_P(\vec{a})\}$ is unsatisfiable. Joyner [22] has established the fundamental principles for deciding a first-order fragment $\mathcal{L}$ by resolution: first, one selects a sound and complete clausal calculus $\mathcal{R}$; second, one identifies a set of clauses $\mathcal{N}$ such that (i) $\mathcal{N}$ is finite for a finite signature, and (ii) the translation of each formula $\alpha \in \mathcal{L}$ into clauses produces only clauses from $\mathcal{N}$; and third, one demonstrates that $\mathcal{N}$ is closed under $\mathcal{R}$— that is, one shows that applying an inference of $\mathcal{R}$ to clauses from $\mathcal{N}$ produces a clause in $\mathcal{N}$. This is sufficient to obtain a refutation decision procedure for $\mathcal{L}$: given any formula $\alpha \in \mathcal{L}$, a saturation by $\mathcal{R}$ of the clauses corresponding to α will, in the worst case, derive all clauses of $\mathcal{N}$.

Note, however, that Joyner's principles allow us only to check whether some tuple $\vec{a}$ is an answer to Q over $\Xi(\mathcal{K}) \cup E'_{\mathcal{T}}$. In order to compute the entire set $\mathsf{ans}(Q, \Xi(\mathcal{K}) \cup E'_{\mathcal{T}})$, we make use of the *answer literal* technique [20]: instead of saturating $\Xi(\mathcal{K}) \cup E'_{\mathcal{T}} \cup Q_C \cup \{\bot \leftarrow Q_P(\vec{a})\}$, we saturate $\Xi(\mathcal{K}) \cup E'_{\mathcal{T}} \cup Q_C$ by $\mathcal{R}^{DL}$—a suitably parameterized RFS calculus. The following lemma shows that by doing so we shall compute all answers to Q over $\Xi(\mathcal{K}) \cup E'_{\mathcal{T}}$.

Lemma 11. *Let $\mathcal{K}$ be an $\mathcal{ELHIO}$ KB, and $Q = \langle Q_P, Q_C \rangle$ a conjunctive query. We have that $\vec{a} \in \mathsf{ans}(Q, \Xi(\mathcal{K}) \cup E'_{\mathcal{T}})$ iff $Q_P(\vec{a}) \in (\Xi(\mathcal{K}) \cup E'_{\mathcal{T}} \cup Q_C)_{\mathcal{R}^{DL}}$.*

Proof. Clearly, $\vec{a} \in \mathsf{ans}(Q, \Xi(\mathcal{K}) \cup E'_{\mathcal{T}})$ iff $\Xi(\mathcal{K}) \cup E'_{\mathcal{T}} \cup Q_C \cup \{\bot \leftarrow Q_P(\vec{a})\}$ is unsatisfiable. We show that the latter is the case iff $Q_P(\vec{a})$ is contained in $(\Xi(\mathcal{K}) \cup E'_{\mathcal{T}} \cup Q_C)_{\mathcal{R}^{DL}}$. The ($\Leftarrow$) direction is trivial. For the ($\Rightarrow$) direction, note that $\Xi(\mathcal{K}) \cup E'_{\mathcal{T}} \cup Q_C$ does not contain clauses with the empty head, so a saturation of $\Xi(\mathcal{K}) \cup E'_{\mathcal{T}} \cup Q_C$ by $\mathcal{R}^{DL}$ cannot derive the empty clause. Furthermore, the predicate Q_P does not occur in the body of any clause in $\Xi(\mathcal{K}) \cup E'_{\mathcal{T}} \cup Q_C$; thus, $\mathcal{R}^{DL}$ can derive the empty clause from $\Xi(\mathcal{K}) \cup E'_{\mathcal{T}} \cup Q_C \cup \{\bot \leftarrow Q_P(\vec{a})\}$ only if $Q_P(\vec{a}) \in (\Xi(\mathcal{K}) \cup E'_{\mathcal{T}} \cup Q_C)_{\mathcal{R}^{DL}}$. Since any RFS calculus is sound and complete [5], the claim of this lemma follows. $\qquad\square$

We now apply Joyner's principles and the answer literal technique to show how to compute $\mathsf{ans}(Q, \Xi(\mathcal{K}) \cup E'_{\mathcal{T}})$ using resolution. With $\mathcal{N}^{DL}$ we denote the clause set defined in Definition 7. Clearly, $\mathcal{N}^{DL}$ is finite assuming that Q and $\Xi(\mathcal{K})$ are finite; moreover, it can be verified that $\Xi(\mathcal{K}) \cup E'_{\mathcal{T}} \cup Q_C \subseteq \mathcal{N}^{DL}$. In the rest of this section we present $\mathcal{R}^{DL}$ and we show that $\mathcal{N}^{DL}$ is closed under $\mathcal{R}^{DL}$. We formally define $\mathcal{N}^{DL}$ and $\mathcal{R}^{DL}$ as follows.

Definition 7. *Let $\mathcal{K}$ be an $\mathcal{ELHIO}$ KB and $Q = \langle Q_P, Q_C \rangle$ a conjunctive query. The set of $\mathcal{ELHIO}$ clauses $\mathcal{N}^{DL}$ is the set of all clauses of types shown in Table 4 constructed using the symbols in Q and $\Xi(\mathcal{K})$ such that for every clause $C \in \mathcal{N}^{DL}$ the following properties hold:*

- *(i) C is safe.*
- *If C is of type 1.1, then*
    - *(ii) $\mathsf{var}(C) = 1$, and*
    - *(iii) the head of C is of the form $A(a)$, $A(x)$, $A(f(x))$, $P(a,b)$, $P(a,x)$, $P(x,a)$, $P(a,f(x))$, $P(f(x),a)$, $P(x,f(x))$, or $P(f(x),x)$.*

Table 4. Clause Set $\mathcal{N}^{DL}$ for $Q = \langle Q_P, Q_C \rangle$ and $\Xi(\mathcal{K})$

Type	$\mathcal{ELHIO}$ clause
1.1	$D(\vec{s}) \leftarrow \bigwedge A_i(a_i) \wedge \bigwedge B_j(x) \wedge \bigwedge C_k(f(x))$
1.2	$\mathcal{O}(o)$
2.1	$A(x) \leftarrow \underline{P(x,y)} \wedge [B(y)]$
2.2	$A(x) \leftarrow \underline{P(y,x)} \wedge [B(y)]$
2.3	$P(x,a) \leftarrow \underline{P(x,y)} \wedge A(y)$
2.4	$P(a,x) \leftarrow \underline{P(y,x)} \wedge A(y)$
2.5	$P(x,y) \leftarrow \underline{S(x,y)}$
2.6	$P(x,y) \leftarrow \underline{S(y,x)}$
3.1	$\underline{x \approx a} \leftarrow A(x)$
3.2	$A(y) \leftarrow A(x) \wedge x \approx y \wedge \underline{\mathcal{O}(y)}$
3.3	$P(x,z) \leftarrow P(x,y) \wedge y \approx \underline{z} \wedge \underline{\mathcal{O}(z)}$
3.4	$P(z,x) \leftarrow P(y,x) \wedge y \approx z \wedge \underline{\mathcal{O}(z)}$
3.5	$A(a) \leftarrow A(x) \wedge \underline{x \approx a}$
3.6	$P(x,a) \leftarrow P(x,y) \wedge \underline{y \approx a}$
3.7	$P(a,x) \leftarrow P(y,x) \wedge \underline{y \approx a}$
4	$Q_P(\vec{s}) \leftarrow \bigwedge D_i(\vec{t_i})$

Note 3. D (possibly with subscripts) denotes a unary or a binary predicate; x, y, and z denote variables; a (possibly with subscripts) denotes a constant; s and t (possibly with subscripts) denote terms; and $[D(s)]$ denotes a possible occurrence of the atom $D(s)$ in the clause.

- If C is of type 4, then
    - *(iv)* $\mathsf{var}(C) \leq \mathsf{var}(Q_C)$,
    - *(v)* $\mathsf{depth}(C) \leq \max(1, \mathsf{var}(Q_C) - \mathsf{var}(C))$, *and*
    - *(vi) if a variable x occurs in a functional term in C, then x occurs in every functional term in C.*

With $\mathcal{R}^{DL}$ we denote the RFS calculus parameterized with the selection function S defined as follows.

- If $C \in \mathcal{N}^{DL}$ *is of type 1.1, then S selects the head atom H if the body is empty or if the depth of H is greater than the maximal depth of the body; otherwise, S selects all deepest covering body atoms.*
- If $C \in \mathcal{N}^{DL}$ *is of type 1.2, 2.1–2.6, or 3.1–3.7, then S selects the atom underlined in Table 4.*
- If $C \in \mathcal{N}^{DL}$ *is of type 4, then S selects the head atom H if the body is empty or if H contains functional terms; otherwise, S selects all deepest body atoms.*

Since $\mathcal{R}^{DL}$ is sound and complete [5], in order to obtain a decision procedure we only need to show that each saturation terminates. We do this in Lemma 12 by showing that $\mathcal{N}^{DL}$ is closed under $\mathcal{R}^{DL}$.

Lemma 12. *For every two clauses C_1 and $C_2 \in \mathcal{N}^{DL}$, and every resolvent C_r of C_1 and C_2 by $\mathcal{R}^{DL}$, we have that $C_r \in \mathcal{N}^{DL}$.*

Table 5. Inferences of $\mathcal{R}^{DL}$ on $\mathcal{N}^{DL}$

3.2 + 1.2 = 3.5:
$$\frac{A(y) \leftarrow A(x) \wedge x \approx y \wedge \mathcal{O}(y) \quad \mathcal{O}(o)}{A(o) \leftarrow A(x) \wedge x \approx o}$$
3.3 + 1.2 = 3.6 and **3.4 + 1.2 = 3.7** are analogous.

3.5 + 3.1 = 1.1:
$$\frac{A(o) \leftarrow A(x) \wedge \underline{x \approx o} \quad \underline{x \approx o} \leftarrow B(x)}{A(o) \leftarrow A(x) \wedge B(x)}$$
3.6 + 3.1 = 2.3 and **3.7 + 3.1 = 2.4** are analogous.

2.1 + 1.1 = 1.1:
$$\frac{A(x) \leftarrow P(x,y) \wedge [B(y)] \quad P(s,t) \leftarrow \bigwedge A_i(a_i) \wedge \bigwedge B_j(x)}{A(s) \leftarrow \bigwedge A_i(a_i) \wedge \bigwedge B_j(x) \wedge [B(t)]}$$
2.2–2.6 + 1.1 = 1.1 are analogous.

1.1 + 1.1 = 1.1:
$$\frac{D(\vec{s}) \leftarrow \bigwedge A_i(a_i) \wedge A(b) \quad A(b)}{D(\vec{s}) \leftarrow \bigwedge A_i(a_i)}$$
(a) $D(\vec{s})$ is of the form $A(a)$ or $P(a,b)$.

1.1 + 1.1 = 1.1:
$$\frac{D(\vec{s}) \leftarrow \bigwedge A_i(a_i) \wedge \bigwedge B_j(x) \wedge \underline{B(x)} \quad \underline{B(f(x))} \leftarrow \bigwedge A'_i(a_i) \wedge \bigwedge B'_j(x)}{D(\vec{s})\sigma \leftarrow \bigwedge A_i(a_i) \wedge \bigwedge B_j(f(x)) \wedge \bigwedge A'_i(a_i) \wedge \bigwedge B'_j(x)}$$
(a) $D(\vec{s})$ is of the form $A(a)$, $A(x)$, $P(a,b)$, $P(a,x)$, or $P(x,a)$.
(b) $\sigma = \{x \mapsto f(x)\}$

1.1 + 1.1 = 1.1:
$$\frac{D(\vec{s}) \leftarrow \bigwedge A_i(a_i) \wedge \bigwedge B_j(x) \wedge \bigwedge C_k(f(x)) \wedge \underline{C(f(x))} \quad \underline{C(f(x))} \leftarrow \bigwedge A'_i(a_i) \wedge \bigwedge B'_j(x)}{D(\vec{s}) \leftarrow \bigwedge A_i(a_i) \wedge \bigwedge B_j(x) \wedge \bigwedge C_k(f(x)) \wedge \bigwedge A'_i(a_i) \wedge \bigwedge B'_j(x)}$$
(a) $D(\vec{s})$ is of the form $A(a)$, $A(x)$, $A(f(x))$, $P(a,b)$, $P(a,x)$, $P(x,a)$, $P(a,f(x))$,
$P(f(x),a)$, $P(x,f(x))$, or $P(f(x),x)$.

---

4 + 1.1 = 4:
$$\frac{Q_P(\vec{s}) \leftarrow \underline{D(\vec{t})} \wedge \bigwedge D_i(\vec{u_i}) \quad \underline{D(\vec{v})} \leftarrow \bigwedge A_i(a_i) \wedge \bigwedge B_j(x)}{Q_P(\vec{s})\sigma \leftarrow A_i(a_i)\sigma \wedge \bigwedge B_j(x)\sigma \wedge \bigwedge D_i(\vec{u_i})\sigma}$$
(a) $\sigma = \mathsf{MGU}(D(\vec{t}), D(\vec{v}))$

Note 4. The notation $X + Y = Z$ denotes that resolving a clause of type X with a clause of type Y produces a clause of type Z.

Proof. The possible inferences are summarized in Table 5. If neither C_1 nor C_2 is of type 4, it is straightforward to check that $C_r \in \mathcal{N}^{DL}$; hence, we assume that C_1 is of type 4. For resolution to be possible, C_2 must be of type 1.1 and the selection function S must select the head $D(\vec{v})$ of C_2. By the definition of S, if $D(\vec{v})$ is of the form $A(a)$ or $P(a, b)$, then the body of C_2 is empty. Therefore, the inference only reduces the number of body atoms of C_1, so C_r is of type 4. We now consider the case where $D(\vec{v})$ is of the form $P(\alpha, f(x))$ for α a constant a or the variable x; the proof for the other forms of $D(\vec{v})$ is analogous. Let $D(\vec{t})$ be of the form $P(s, u)$. For unification to be possible, the term u must be a variable x_u or a functional term of the form $f(u')$.

- Let u be a variable x_u. If α is also a variable x, then $\sigma = \{x \mapsto s, x_u \mapsto f(s)\}$. If α is a constant a, then $\sigma = \{s \mapsto a, x_u \mapsto f(x)\}$. In both cases, due to the occurs-check in unification, x_u cannot occur in s. The inference thus decreases the number of variables by one, so C_r satisfies condition (iv). Moreover, since C_1 satisfies condition (vi), x_u does not occur in any functional term in C_1 (because it does not occur in s). Hence, even though x_u is mapped to a functional term, we have that $\mathsf{depth}(C_r) \leq \mathsf{depth}(C_1) + 1$, so C_r satisfies condition (v). Finally, since every occurrence of x_u is replaced with the same term, the resolvent C_r satisfies condition (vi) as well.
- Let u be a functional term $f(u')$. If α is a variable x, then we have that either $s = u'$ and $\sigma = \{x \mapsto u'\}$; s is a variable x_s (not occurring in u') and $\sigma = \{x \mapsto u', x_s \mapsto u'\}$; or u' is a variable x'_u and $\sigma = \{x \mapsto s, x'_u \mapsto s\}$. If α is a constant a, then $\sigma = \{s \mapsto a, x \mapsto u'\}$. In all cases, the inference does not increase the number of variables or functional terms; therefore, C_r satisfies conditions (iv) and (v). Furthermore, the inference does not introduce new functional terms, so C_r satisfies condition (vi) as well.

$\square$

Theorem 1 follows from Lemma 11 and Lemma 12.

Theorem 1. *For $\mathcal{K}$ an $\mathcal{ELHIO}$ KB, and $Q = \langle Q_P, Q_C \rangle$ a conjunctive query, we have that* $\mathsf{ans}(Q, \Xi(\mathcal{K}) \cup E'_{\mathcal{T}}) = \{\vec{a} \mid Q_P(\vec{a}) \in (\Xi(\mathcal{K}) \cup E'_{\mathcal{T}} \cup Q_C)_{\mathcal{R}^{DL}}\}$.

5   Resolution-based Query Rewriting

In this section we present a query rewriting algorithm for $\mathcal{ELHIO}$. We derive it from the answering algorithm presented in Section 4: we show that in order to answer Q over $\Xi(\mathcal{K}) \cup E'_{\mathcal{T}}$, one can *first* saturate $\Xi(\mathcal{T}) \cup E'_{\mathcal{T}} \cup Q_C$ with $\mathcal{R}^{DL}$ to produce a rewriting $\mathsf{rew}(Q, \mathcal{T})$ of Q w.r.t. $\mathcal{T}$, and *then* evaluate $\mathsf{rew}(Q, \mathcal{T})$ over $\mathcal{A}$. Our goal is to obtain a worst-case optimal rewriting for various sublanguages of $\mathcal{ELHIO}$—that is, if $\mathcal{T}$ is in $\mathcal{ELHIO}$, then $\mathsf{rew}(Q, \mathcal{T})$ should be a datalog query; if $\mathcal{T}$ is in DL-Lite$^+$, then $\mathsf{rew}(Q, \mathcal{T})$ should consist of a union of conjunctive queries and a linear datalog query; finally, if $\mathcal{T}$ is in DL-Lite$_R$, then $\mathsf{rew}(Q, \mathcal{T})$ should be a union of conjunctive queries.

In a nutshell, our algorithm takes as input Q and $\mathcal{T}$, and proceeds in two steps: (i) the set $\Xi(\mathcal{T}) \cup E'_{\mathcal{T}} \cup Q_C$ is saturated using $\mathcal{R}^{DL}$, and (ii) the resulting logic program $(\Xi(\mathcal{T}) \cup E'_{\mathcal{T}} \cup Q_C)_{\mathcal{R}^{DL}}$ is transformed into a datalog query $\mathsf{rew}(Q, \mathcal{T})$ of optimal form. By Lemma 12, we know that every saturation by $\mathcal{R}^{DL}$ terminates; hence, the main challenge in computing the rewriting lies in ensuring that $(\Xi(\mathcal{T}) \cup E'_{\mathcal{T}} \cup Q_C)_{\mathcal{R}^{DL}}$ can always be transformed into an optimal rewriting. In the rest of this section, we first show how to convert $(\Xi(\mathcal{T}) \cup E'_{\mathcal{T}} \cup Q_C)_{\mathcal{R}^{DL}}$ into a nonoptimal datalog rewriting, and then we present an optimization step to obtain rewritings of optimal form.

5.1   Elimination of Function Symbols

We have devised $\mathcal{R}^{DL}$ such that transforming $(\Xi(\mathcal{T}) \cup E'_{\mathcal{T}} \cup Q_C)_{\mathcal{R}^{DL}}$ into a datalog program simply amounts to discarding clauses containing function symbols. Intuitively, the reason is that by saturating $\Xi(\mathcal{T}) \cup E'_{\mathcal{T}} \cup Q_C$ we obtain new clauses that act as "inference shortcuts" rendering clauses containing function symbols unnecessary to derive facts. We illustrate this point with an example.

Example 5. Consider an $\mathcal{ELHIO}$ TBox $\mathcal{T}_3$ containing the following axioms:

$$\text{Catholic} \sqsubseteq \exists\text{believesIn.Deity}$$

$$\exists\text{believesIn.Deity} \sqsubseteq \text{Theist}$$

The TBox $\mathcal{T}_3$ states that a catholic believes in a deity, and that someone who believes in a deity is a theist. Clearly, we have that $\mathcal{T}_3 \models \text{Catholic} \sqsubseteq \text{Theist}$.

The set $\Xi(\mathcal{T}_3)$ contains the following clauses:

$$\underline{\text{believesIn}(x, \text{deityOf}(x))} \leftarrow \text{Catholic}(x) \tag{41}$$

$$\underline{\text{Deity}(\text{deityOf}(x))} \leftarrow \text{Catholic}(x) \tag{42}$$

$$\text{Theist}(x) \leftarrow \underline{\text{believesIn}(x, y)} \wedge \text{Deity}(y) \tag{43}$$

Resolving (41) with (43) produces

$$\text{Theist}(x) \leftarrow \underline{\text{Deity}(\text{deityOf}(x))} \wedge \text{Catholic}(x). \tag{44}$$

Resolving (42) with (44) produces

$$\text{Theist}(x) \leftarrow \text{Catholic}(x). \tag{45}$$

Therefore, the saturation of $\Xi(\mathcal{T}_3)$ will create a new *function-free* "shortcut" clause (45) that explicitly represents the fact that $\mathcal{T}_3 \models \text{Catholic} \sqsubseteq \text{Theist}$.

According to the definition of the certain answers, in order to answer conjunctive queries, we are only interested in tuples of *constants* (not of functional terms). Moreover, since it is possible to obtain $\mathsf{rew}(Q, \mathcal{T})$ by saturating $\Xi(\mathcal{T}) \cup E'_{\mathcal{T}} \cup Q_C$ with $\mathcal{R}^{DL}$, every relevant implicit logical consequence (such as (45)) that can be used to derive new facts will be made explicit by the saturation. Therefore, we can safely get rid of every clause containing function symbols in $(\Xi(\mathcal{T}) \cup E'_{\mathcal{T}} \cup Q_C)_{\mathcal{R}^{DL}}$. The following definition summarizes the first step of the algorithm.

Definition 8. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be an $\mathcal{ELHIO}$ KB and $Q = \langle Q_P, Q_C \rangle$ a conjunctive query. With $\mathsf{ff}(Q, \mathcal{T})$ we denote the set that contains exactly every function-free clause contained in $(\Xi(\mathcal{T}) \cup E'_{\mathcal{T}} \cup Q_C)_{\mathcal{R}^{DL}}$.*

We now show that $\mathsf{ff}(Q, \mathcal{T})$ is a rewriting of Q w.r.t. $\Xi(\mathcal{T}) \cup E'_{\mathcal{T}}$, albeit not necessarily an optimal one. To this end, we first prove that in order to compute the answers to Q over an $\mathcal{ELHIO}$ KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, we can always "postpone" the inferences involving ground facts of the form $A(a)$ or $P(a, b)$ in the saturation of $\Xi(\mathcal{K}) \cup E'_{\mathcal{T}} \cup Q_C$.

Lemma 13. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be an $\mathcal{ELHIO}$ KB and $Q = \langle Q_P, Q_C \rangle$ a conjunctive query. For every clause C of type 4, if $C \in (\Xi(\mathcal{K}) \cup E'_{\mathcal{T}} \cup Q_C)_{\mathcal{R}^{DL}}$, then there is a clause C' of type 4 such that $C' \in (\Xi(\mathcal{T}) \cup E'_{\mathcal{T}} \cup Q_C)_{\mathcal{R}^{DL}}$ and there is a derivation tree for C from $\{C'\} \cup \mathsf{g}(\mathsf{ff}(Q, \mathcal{T}), \mathcal{A})$, where $\mathsf{g}(\mathsf{ff}(Q, \mathcal{T}), \mathcal{A})$ is the set of exactly all clauses of the form $A(a)$ or $P(a, b)$ contained in $(\mathsf{ff}(Q, \mathcal{T}) \cup \mathcal{A})_{\mathcal{R}^{DL}}$.*

Proof. We prove the claim by induction on the height of a derivation tree of C. If the derivation tree has height zero, then C' is the only clause contained in Q_C, and the claim trivially follows. We assume that the claim holds for each clause derived by a derivation tree of height n, and consider a clause C derived by a derivation tree of height $n + 1$. Let C_1 and C_2 be the clauses that are resolved to obtain C. W.l.o.g. we assume that C_1 is of type 4 and C_2 is of type 1.1. Hence, by the induction hypothesis, there is a clause C'_1 of type 4 such that $C'_1 \in (\Xi(\mathcal{T}) \cup E'_{\mathcal{T}} \cup Q_C)_{\mathcal{R}^{DL}}$ and C_1 can be derived from $\{C'_1\} \cup \mathsf{g}(\mathsf{ff}(Q, \mathcal{T}), \mathcal{A})$.

The selection function S selects a deepest covering body atom in C_1 and the head $D(\vec{s})$ of C_2, which is of the form $A(a)$, $A(f(x))$, $R(a, b)$, $R(a, f(x))$, $R(f(x), a)$, $R(x, f(x))$, or $R(f(x), x)$.

If $D(\vec{s})$ is ground, it follows from the definition of $\mathcal{R}^{DL}$ that the body of C_2 is empty. Given the fact that S selects the deepest atoms in clauses of type 1.1, and that functional terms do not unify with constants, we have that every ground clause has function-free premises. Therefore, we have that $C_2 \in \mathsf{g}(\mathsf{ff}(Q, \mathcal{T}), \mathcal{A})$, and the claim follows for $C' = C'_1$.

We now consider the case where $D(\vec{s})$ is not ground. We show the claim for the case where $D(\vec{s})$ is of the form $R(\alpha, f(x))$, for α a constant a or the variable x; the proof for the other cases is analogous. Let C_2 be a clause of the form (46). Note that every clause of type 1.1 is safe and contains at most one variable; therefore, since S selects the covering atoms of this type of clauses, we have that if $D(\vec{s})$ is not ground, then $C_2 \in (\Xi(\mathcal{T}) \cup E'_{\mathcal{T}})_{\mathcal{R}^{DL}}$.

We know there is a derivation tree for C_1 from $\{C'_1\} \cup \mathsf{g}(\mathsf{ff}(Q, \mathcal{T}), \mathcal{A})$. Therefore, given the fact that $\mathsf{g}(\mathsf{ff}(Q, \mathcal{T}), \mathcal{A})$ contains only ground unit clauses, we have that a set $\{D_k(\vec{a_k})\} \subseteq \mathsf{g}(\mathsf{ff}(Q, \mathcal{T}), \mathcal{A})$ exists such that resolving C'_1 on body atoms $\{D_k(\vec{r_k})\}$ with the atoms of $\{D_k(\vec{a_k})\}$ produces C_1. Furthermore, all such resolution inferences just remove body atoms; hence, since C_1 is to be resolved with C_2, the clause C'_1 is of the form (47), and C_1 of the form (48), where δ maps the variables occurring in $\{D_k(\vec{r_k})\}$ to some constants occurring in $\{D_k(\vec{a_k})\}$. Note that no inference used to derive C_1 changes the number of function symbols of C'_1; therefore, since $R(s, t)\delta$ is deepest in C_1, we have that $R(s, t)$ is deepest in

C_1'. Moreover, each variable of C_1' that is replaced by δ with a constant clearly does not occur in $R(s,t)\delta$; hence, the substitutions δ and σ have disjoint domains, and $\sigma = \delta\sigma$. Resolving C_1 and C_2 produces C, a clause of the form (49), where $\sigma = \mathsf{MGU}(R(s,t)\delta, R(\alpha, f(x)))$.

$$C_2 = \underline{R(\alpha, f(x))} \leftarrow \bigwedge A_i(t_i) \tag{46}$$

$$C_1' = Q_P(\vec{u}) \leftarrow \underline{R(s,t)} \wedge \bigwedge D_j(\vec{w_j}) \wedge \bigwedge D_k(\vec{r_k}) \tag{47}$$

$$C_1 = Q_P(\vec{u})\delta \leftarrow \underline{R(s,t)\delta} \wedge \bigwedge D_j(\vec{w_j})\delta \tag{48}$$

$$C = Q_P(\vec{u})\delta\sigma \leftarrow \bigwedge A_i(t_i)\sigma \wedge \bigwedge D_j(\vec{w_j})\delta\sigma \tag{49}$$

We now transform this derivation into a derivation in which all inferences with clauses in $\mathsf{g}(\mathsf{ff}(Q, \mathcal{T}), \mathcal{A})$ are performed in the end. Let C' be the clause obtained by resolving C_1' and C_2. Given the form of C_1' and C_2, the clause C' is of the form (50), where $\sigma' = \mathsf{MGU}(R(s,t), R(\alpha, f(x)))$.

$$C' = Q_P(\vec{u})\sigma' \leftarrow \bigwedge A_i(t_i)\sigma' \wedge \bigwedge D_j(\vec{w_j})\sigma' \wedge \bigwedge D_k(\vec{r_k})\sigma' \tag{50}$$

Since $R(s,t)$ is deepest in C_1', the inference between C_1' and C_2 satisfies the selection function of $\mathcal{R}^{DL}$. Moreover, since both C_1' and C_2 are derivable from $\Xi(\mathcal{T}) \cup E_{\mathcal{T}}' \cup Q_C$, the clause C' is derivable from $\Xi(\mathcal{T}) \cup E_{\mathcal{T}}' \cup Q_C$ as well. Let D now be the clause obtained by resolving $\{D_k(\vec{r_k})\sigma'\}$ in C' with atoms of $\mathsf{g}(\mathsf{ff}(Q, \mathcal{T}), \mathcal{A})$. The clause D has form (51), where δ' maps some variables of C' to constants.

$$D = Q_P(\vec{u})\sigma'\delta' \leftarrow \bigwedge A_i(t_i)\sigma'\delta' \wedge \bigwedge D_j(\vec{w_j})\sigma'\delta' \tag{51}$$

We now prove that $C = D$. Given the forms of C and D, and since $\sigma = \delta\sigma$, it suffices to show that $\delta\sigma = \sigma'\delta'$. We first assume that $\alpha = x$, so C_2 has head $R(x, f(x))$. Let y be a variable that occurs in $R(s,t)$ that is replaced by δ with a constant. Clearly, σ does not contain such y; therefore, w.l.o.g. we can assume that (*) if σ maps a variable z to a constant $y\delta$, then σ' maps z to y. Note that σ' may still map a variable y_1 occurring in δ to another variable y_2 occurring in δ, which implies that $y_1\delta = y_2\delta$. Due to (*), σ and σ' have the same domain modulo variables such as y_1; therefore, $\{D_k(\vec{r_k})\}$ and $\{D_k(\vec{r_k})\sigma'\}$ are not necessarily the same. However, note that since σ' maps variables such as y_1 to other *variables* such as y_2, the atoms $\{D_k(\vec{r_k})\sigma'\}$ can be resolved with the ground atoms $\{D_k(\vec{a_k})\}$ via δ'. So, we know that σ and σ' have the same domain modulo variables such as y_1; moreover, due to the possible variable renaming of σ' on variables such as y_1, we have that δ and δ' are the same modulo the mappings concerning variables such as y_1. Therefore, since $y_2\delta' = y_2\delta = y_1\delta$, we have that $\delta\sigma = \sigma'\delta'$.

We now assume that $\alpha = a$, so C_2 has head $R(a, f(x))$. We consider the forms of $R(s,t)\delta$. For the inference between C_1 and C_2 to be possible, the term $s\delta$

cannot be a functional term, and $t\delta$ cannot be a constant. Therefore, $R(s,t)\delta$ can be of the form (i) $R(x_s, x_t)$, (ii) $R(a, x_t)$, (iii) $R(x_s, f(t'\delta))$, or (iv) $R(a, f(t'\delta))$. W.l.o.g. we assume that if $R(s,t)\delta$ is of form (i), $\sigma = \{x_s \mapsto a, x_t \mapsto f(x)\}$; if it is of form (ii), $\sigma = \{x_t \mapsto f(x)\}$; if it is of form (iii), $\sigma = \{x_s \mapsto a, x \mapsto t'\delta\}$; and if it is of form (iv), $\sigma = \{x \mapsto t'\delta\}$. Since $t\delta$ cannot be a constant, x_t is not in the domain of δ. In case x_s is not in the domain of δ either, w.l.o.g. we can assume that σ' and σ have the same domain. As can be seen, such a domain is disjoint from the domain of δ; therefore, we have that $\sigma = \sigma'\delta$ and $\delta = \delta'$. Therefore, $\delta\sigma = \sigma'\delta'$ since $\sigma = \delta\sigma$. In case x_s is in the domain of δ, we have that δ has to map x_s to a in order for the inference between C_1 and C_2 to be possible. Moreover, $R(s,t)\delta$ is of form (ii) or (iv). W.l.o.g. we assume that σ and σ' have the same domain modulo x_s. Moreover, since x_s is the only variable of δ that is mapped to a term (namely, a) by σ', we have that δ and δ' are the same modulo the mapping concerning x_s. Furthermore, both δ and σ' map x_s to the same constant (namely, a); therefore, $\delta\sigma = \sigma'\delta'$. $\qquad\square$

We now demonstrate the desired relationship between $\mathsf{ans}(Q, \Xi(\mathcal{K}) \cup E'_{\mathcal{T}})$ and $\mathsf{ans}(\mathsf{ff}(Q, \mathcal{T}), \mathcal{A})$. By Theorem 1, it follows that $\vec{a} \in \mathsf{ans}(Q, \Xi(\mathcal{K}) \cup E'_{\mathcal{T}})$ iff $Q_P(\vec{a}) \in (\Xi(\mathcal{K}) \cup E'_{\mathcal{T}} \cup Q_C)_{\mathcal{R}^{DL}}$. By Lemma 13, we have that there is a clause $C' \in (\Xi(\mathcal{T}) \cup E'_{\mathcal{T}} \cup Q_C)_{\mathcal{R}^{DL}}$ of type 4, such that there is a derivation tree for $Q_P(\vec{a})$ from $\{C'\} \cup \mathsf{g}(\mathsf{ff}(Q, \mathcal{T}), \mathcal{A})$. Since $Q_P(\vec{a})$ does not contain function symbols, C' does not contain function symbols either, so $C' \in \mathsf{ff}(Q, \mathcal{T})$. Hence, by the definition of $\mathsf{g}(\mathsf{ff}(Q, \mathcal{T}), \mathcal{A})$, it follows that there is a derivation tree for $Q_P(\vec{a})$ from $\mathsf{ff}(Q, \mathcal{T}) \cup \mathcal{A}$. Lemma 14 follows immediately.

Lemma 14. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be an $\mathcal{ELHIO}$ KB, and $Q = \langle Q_P, Q_C \rangle$ a conjunctive query. We have that $\mathsf{ans}(Q, \Xi(\mathcal{K}) \cup E'_{\mathcal{T}}) = \mathsf{ans}(\mathsf{ff}(Q, \mathcal{T}), \mathcal{A})$.*

According to Lemma 14, the datalog program $\mathsf{ff}(Q, \mathcal{T})$ is a rewriting of Q w.r.t. $\Xi(\mathcal{T}) \cup E'_{\mathcal{T}}$. We note, however, that $\mathsf{ff}(Q, \mathcal{T})$ is not necessarily optimal for DL-Lite$^+$. We illustrate the point with an example.

Example 6. Consider a DL-Lite$^+$ TBox $\mathcal{T}_4$ consisting of the following axioms:

$$\exists\mathrm{believesIn.Deity} \sqsubseteq \mathrm{Theist}$$
$$\mathrm{praysTo} \sqsubseteq \mathrm{believesIn}$$

Since $\mathcal{T}_4$ does not contain $\mathcal{O}$-constants, then $E'_{\mathcal{T}_4} = \emptyset$; therefore $\Xi(\mathcal{T}_4) \cup E'_{\mathcal{T}_4}$ consists of the following clauses:

$$\mathrm{Theist}(x) \leftarrow \underline{\mathrm{believesIn}(x,y)} \wedge \mathrm{Deity}(y)$$
$$\mathrm{believesIn}(x,y) \leftarrow \underline{\mathrm{praysTo}(x,y)}$$

Consider the query Q_4: $Q_4(x) \leftarrow \mathrm{Theist}(x)$. It is not difficult to verify that $\Xi(\mathcal{T}_4) \cup E'_{\mathcal{T}_4} \cup \{Q_4\} = \mathsf{ff}(Q_4, \mathcal{T}_4)$. This is so because such a set does not contain function symbols, and it is already saturated by $\mathcal{R}^{DL}$. In the case of DL-Lite$^+$, a worst-case optimal rewriting consists of a union of conjunctive queries and a *linear* datalog program [32]. In this case, however, predicates Theist and believesIn

are both IDB predicates; therefore, the datalog program $\mathsf{ff}(Q_4, \mathcal{T}_4) \setminus \{Q_4\}$ is not linear.

We now introduce a further step that transforms $\mathsf{ff}(Q, \mathcal{T})$ into a datalog program of optimal form.

5.2 Optimizing the Rewriting

The second step of the algorithm is based on the notion of *unfolding* that is formally defined as follows.

Definition 9. *Let C_1 be a clause of the form $D_1(\vec{r}) \leftarrow \bigwedge D_i(\vec{s_i})$ and let C_2 be a clause of the form $D_2(\vec{t}) \leftarrow D_1(\vec{u}) \wedge \bigwedge D_j(\vec{v_j})$. The unfolding of C_1 in C_2 is the clause $D_2(\vec{t})\sigma \leftarrow \bigwedge D_i(\vec{s_i})\sigma \wedge \bigwedge D_j(\vec{v_j})\sigma$, where $\sigma = \mathsf{MGU}(D_1(\vec{r}), D_1(\vec{u}))$. The clause C_1 is said to have been unfolded into C_2.*

Let R and U be sets of safe Horn clauses. With R_U we denote the smallest set such that $R \subseteq R_U$, and for every unfolding C_r of a clause $C_1 \in R \cap U$ in a clause $C_2 \in R$, we have that there is a clause $C'_r \in R_U$ that is equivalent to C_r up to variable renaming. The unfolding of R w.r.t. U is defined as $\mathsf{unfold}(R, U) = R_U \setminus U$.

We show that unfolding clauses does not change the set of "relevant" consequences (ground facts) of a datalog program.

Lemma 15. *Let R and U be sets of safe Horn clauses. For any set of facts A and for any predicate D that does not occur in U, we have $R \cup A \models D(\vec{a})$ iff $\mathsf{unfold}(R, U) \cup A \models D(\vec{a})$.*

Proof. For the ($\Leftarrow$) direction, note that $R \models R_U$ and $\mathsf{unfold}(R, U) \subseteq R_U$; therefore, for each clause C, if $\mathsf{unfold}(R, U) \cup A \models C$ then $R \cup A \models C$.

For the ($\Rightarrow$) direction, let $\Sigma = \langle T, \delta, \lambda \rangle$ be the derivation tree of $D(\vec{a})$. We now inductively define a function $\sigma(t)$ as follows: starting from the leaves upwards, for each $t \in T$, we set $\sigma(t)$ to be the clause obtained from $\lambda(t)$ by unfolding each $\sigma(t.i)$ in the i-th body atom of $\lambda(t)$ provided that $\sigma(t.i) \notin U$ or $\delta(t.i) \in A$; furthermore, we call t a *surviving* node iff $\sigma(t) \notin U$ or $\delta(t) \in A$. We say that a node t_2 is the *closest surviving node* to t_1 if t_2 is a surviving node, if it is a descendent of t_1, and no node on the path between t_1 and t_2 is a surviving node. By the inductive definition of σ, it can be seen that, for each node t, the fact $\delta(t)$ can be derived by resolving $\sigma(t)$ with the set of facts $\{\delta(t_1), \ldots, \delta(t_n)\}$ in one step, where $t_1, \ldots, t_n$ are exactly all the closest surviving nodes to t. Note that, for every node $t \in T$, we have $\sigma(t) \in R_U$. Moreover, if t is a surviving node, then $\sigma(t) \in \mathsf{unfold}(R, U)$. Therefore, if t is a surviving node, the fact $\delta(t)$ can be derived from $\mathsf{unfold}(R, U) \cup A$. Since the predicate D does not occur in U, we have $D(\vec{a}) \notin U$. Furthermore, $\delta(\epsilon) = D(\vec{a})$, so the clause $\sigma(\epsilon)$ contains D in the head, and $\sigma(\epsilon) \notin U$. Thus, ϵ is a surviving node, so $\delta(\epsilon)$ can be derived from $\mathsf{unfold}(R, U) \cup A$. $\qquad\square$

In order to obtain the rewriting $\mathsf{rew}(Q,\mathcal{T})$, the idea is to unfold certain types of clauses in $\mathsf{ff}(Q,\mathcal{T})$ such that the resulting set is an optimal rewriting for the sublanguage of $\mathcal{ELHIO}$ used to model $\mathcal{T}$. We are ready to define the rewriting $\mathsf{rew}(Q,\mathcal{T})$ in terms of $\mathsf{ff}(Q,\mathcal{T})$.

Definition 10. *Let* $Q = \langle Q_P, Q_C \rangle$ *be a conjunctive query, and* $\mathcal{T}$ *an* $\mathcal{ELHIO}$ *TBox. Let* $\mathsf{rew}(Q,\mathcal{T}) = \langle Q_P, \mathsf{unfold}(\mathsf{ff}(Q,\mathcal{T}), U)\rangle$, *where* U *is the set that contains every clause* $C \in \mathcal{N}^{DL}$ *of one of the following forms:*

$$A(x) \leftarrow B(x) \tag{52}$$
$$A(x) \leftarrow R(x,y) \tag{53}$$
$$A(x) \leftarrow R(y,x) \tag{54}$$
$$R(x,y) \leftarrow S(x,y) \tag{55}$$
$$R(x,y) \leftarrow S(y,x) \tag{56}$$

Theorem 2 follows from Lemma 14 and Lemma 15, since w.l.o.g. we can assume that Q_P does not occur in any of the clauses that are to be unfolded to obtain $\mathsf{rew}(Q,\mathcal{T})$.

Theorem 2. *Let* $\mathcal{K} = \langle \mathcal{T}, \mathcal{A}\rangle$ *be an* $\mathcal{ELHIO}$ *KB, and* $Q = \langle Q_P, Q_C \rangle$ *a conjunctive query. We have that* $\mathsf{ans}(Q, \Xi(\mathcal{K}) \cup E'_\mathcal{T}) = \mathsf{ans}(\mathsf{rew}(Q,\mathcal{T}), \mathcal{A})$.

5.3   Structure of the Rewriting

We now consider the form of the rewriting. We show important properties of the structure of $\mathsf{rew}(Q,\mathcal{T})$ that are used in Section 6 to prove complexity results about query answering in $\mathcal{ELHIO}^\neg$.

Lemma 16. *Let* $\mathcal{T}$ *be an* $\mathcal{ELHIO}$ *TBox,* $Q = \langle Q_P, Q_C\rangle$ *be a conjunctive query, and let* $\mathsf{rew}(Q,\mathcal{T}) = \langle Q_P, Q'_C\rangle$. *We have that (i)* $\mathsf{rew}(Q,\mathcal{T})$ *is a datalog query; (ii) if* $\mathcal{T}$ *is a DL-Lite$^+$ TBox, then* Q'_C *consists of a union of conjunctive queries and a linear datalog program; and (iii) if* $\mathcal{T}$ *is a DL-Lite$_R$ TBox, then* $\mathsf{rew}(Q,\mathcal{T})$ *is a union of conjunctive queries.*

Proof. We first consider claim (i). It is immediate to see that, since $\mathsf{ff}(Q,\mathcal{T})$ does not contain functional terms, we have that $\mathsf{rew}(Q,\mathcal{T})$ does not contain functional terms either. Therefore, the claim holds.

We now consider claim (ii). It follows from Table 2 and the definition of DL-Lite$^+$ that $E'_\mathcal{T} = \emptyset$, and $\Xi(\mathcal{T})$ only contains clauses of type 2.5, 2.1, or 1.1 of the form $B(x) \leftarrow A(x)$, $P(x, f(x)) \leftarrow A(x)$, or $B(f(x)) \leftarrow A(x)$. By analyzing the inferences shown in Table 5, one can see that saturating a set of clauses of these forms by $\mathcal{R}^{DL}$ produces only clauses of the same forms plus clauses of the form $A(x) \leftarrow B(f(x)) \wedge C(x)$. Therefore, $\mathsf{ff}(Q,\mathcal{T})$ contains only clauses of type 4 or of the form (52), (53), (55), or $A(x) \leftarrow R(x,y) \wedge B(y)$. Hence, the datalog program Q'_C only contains clauses of type 4, and of the form $A(x) \leftarrow R(x,y) \wedge B(y)$ or $A(x) \leftarrow R(x,y) \wedge S(y,z)$ (which are obtained by unfolding a clause of the

form (53) into a clause of the form $A(x) \leftarrow R(x, y) \wedge B(y)$). Clearly, every binary predicate in Q'_C is an EDB predicate. Moreover, even though there could be unary IDB predicates in Q'_C, we have that there can be at most one unary atom in the clauses of Q'_C which are not of type 4. Therefore Q'_C consists of a union of conjunctive queries and a linear datalog program.

We finally consider claim (iii). It follows from Table 2 and the definition of DL-Lite$_R$ that $E'_{\mathcal{T}} = \emptyset$, and $\Xi(\mathcal{T})$ only contains clauses of type 2.1 with one body atom, 2.2 with one body atom, 2.5, 2.6, or 1.1 of the form $B(x) \leftarrow A(x)$, $P(x, f(x)) \leftarrow A(x)$, $P(f(x), x) \leftarrow A(x)$ or $B(f(x)) \leftarrow A(x)$. By analyzing the inferences shown in Table 5, one can see that saturating a set of clauses of these forms by $\mathcal{R}^{DL}$ produces only clauses of the same forms. Therefore, $\mathsf{ff}(Q, \mathcal{T})$ contains only clauses of type 4 or of the form (52)–(56). Clearly, the datalog program Q'_C only contains clauses of type 4; hence, the claim follows.   □

5.4   Size of the Rewriting

We now provide estimates on the maximal number of clauses M generated by our rewriting algorithm and the maximal size $|\mathsf{rew}(Q, \mathcal{T})|$ of the rewriting.

Lemma 17. *Let $\mathcal{T}$ be an $\mathcal{ELHIO}$ TBox, and Q be a conjunctive query. We have that (i) the maximal number M of clauses generated by our rewriting algorithm is $O(2^{t^q})$ and (ii) the maximal size $|\mathsf{rew}(Q, \mathcal{T})|$ of the rewriting is $O(2^{tq})$.*

Proof. We first consider claim (i). Clearly, we have that $M = |\mathcal{N}^{DL}| + F$, where $|\mathcal{N}^{DL}|$ is the size of the clause set and F is the maximal number of clauses produced in the unfolding step. We provide upper bounds for $|\mathcal{N}^{DL}|$ and F.

We first consider $|\mathcal{N}^{DL}|$. We analyze clauses of type 4 since they are the longest and deepest clauses in $\mathcal{N}^{DL}$. Let p be the number of predicate names occurring in $\Xi(\mathcal{T})$, f the number of constants and function symbols occurring in $\Xi(\mathcal{T})$, v a bound on the number of variables for the clauses of $\Xi(\mathcal{T})$, d a bound on the maximal depth for the clauses of $\Xi(\mathcal{T})$, and b be the number of body atoms of Q_C. It can be readily verified that it is possible to build $pf^d v$ unary atoms and $p(f^d v)^2$ binary atoms with the symbols of $\Xi(\mathcal{T})$. For clauses of type 4, however, both d and v are bounded by $\mathsf{var}(Q_C)$ and consequently by b. Therefore, we can build no more than $U = pf^b b$ unary and $B = p(f^b b)^2$ binary atoms for clauses of type 4. Moreover, it is easy to see that we can build no more than $H = (f^b b)^a$ head atoms for clauses of type 4, where a is the arity of Q_C.

We now determine the *length* of the longest clause of type 4—that is, we determine the maximal number of *occurrences* of unary and binary body atoms in every clause of type 4. The number of unary body atoms of a clause of type 4 may be augmented by resolving it with a clause of type 1.1; however, no inference augments the number of binary atoms, which are bounded by b. Therefore, the longest clause C in $\mathcal{N}^{DL}$ contains no more than one head atom, U unary body atoms and b binary body atoms.

Given the possible atoms that can be built for clauses of type 4 and the length of the longest clause of type 4, it can be readily verified that there can be

no more than $H2^U B^b$ clauses of type 4 in $\mathcal{N}^{DL}$. Note that p and f depend on $\mathcal{T}$, and a and b depend on Q. Let $t = |\mathcal{T}|$ and $q = |Q|$. We have that $H = O((t^q q)^q)$, $U = O(tt^q q)$, and $B = O((t(t^q q)^2)^q)$; therefore, the number of clauses of type 4 contained in $\mathcal{N}^{DL}$ is

$$O((t^q q)^q 2^{tt^q q}(t(t^q q)^2)^q) = O(t^{3q^2+q}q^{3q}2^{qt^{q+1}}) = O(2^{t^q}).$$

Since clauses of type 4 are the longest and deepest clauses in $\mathcal{N}^{DL}$, we have that $|\mathcal{N}^{DL}| = O(2^{t^q})$.

We now consider F. Note that only the function-free clauses of $\mathcal{N}^{DL}$ are unfolded; therefore, we first need to determine the size $|\mathsf{ff}(Q,\mathcal{T})|$ of $\mathsf{ff}(Q,\mathcal{T})$. We do so by analyzing the number of possible clauses of type 4 in $\mathcal{N}^{DL}$ with a depth equal to 0. For such clauses, we can build no more than $H_0 = (fb)^a$ head atoms, $U_0 = pfb$ unary body atoms, and $B_0 = p(fb)^2$ binary body atoms. The length of the longest clause of type 4 with depth 0 is the same as before; therefore, there can be no more than $H_0 2^{U_0} B_0^b$ clauses of type 4 in $\mathsf{ff}(Q,\mathcal{T})$. Since p and f depend on t, and a and b depend on q, we have that $H_0 = O((tq)^q)$, $U_0 = O(ttq)$, and $B_0 = O((t(tq)^2)^q)$; therefore, the number of clauses of type 4 contained in $\mathsf{ff}(Q,\mathcal{T})$ is

$$O((tq)^q 2^{t^2 q}(t(tq)^2)^q) = O(t^{4q}q^{3q}2^{t^2 q}) = O(2^{tq}).$$

The unfolding step may augment the number of binary body atoms in clauses of type 4. Each unary atom of a clause $C \in \mathsf{ff}(Q,\mathcal{T})$ can be replaced with a binary atom. Therefore, for every such C, the unfolding step produces no more than p^{U_0} clauses. Hence, the number of clauses of type 4 produced in the unfolding step is the number of clauses of this type contained in $\mathsf{ff}(Q,\mathcal{T})$ multiplied by p^{U_0}. Consequently, the number of clauses of type 4 produced in the unfolding step is

$$O(2^{tq}t^{ttq}) = O(2^{tq}).$$

Since clauses of type 4 are the longest and deepest clauses produced in the unfolding step, we have that $F = O(2^{tq})$.

We conclude that the maximal number of clauses M generated by our rewriting algorithm is

$$O(|\mathcal{N}^{DL}| + F) = O(2^{t^q} + 2^{tq}) = O(2^{t^q}).$$

We now consider claim (ii). The number of clauses of type 4 contained in $\mathsf{rew}(Q,\mathcal{T})$ is the number of clauses of this type contained in $\mathsf{ff}(Q,\mathcal{T})$ plus the new clauses of this type produced in the unfolding step, that is

$$O(2^{tq} + 2^{tq}) = O(2^{tq}).$$

Since clauses of type 4 are the longest and deepest clauses in $\mathsf{ff}(Q,\mathcal{T})$ and in the set of clauses produced in the unfolding step, we have that $|\mathsf{rew}(Q,\mathcal{T})| = O(2^{tq})$.

□

Note that according to Lemma 17, even though the number of clauses generated by our algorithm is doubly exponential w.r.t. $|\mathcal{T}|$ and $|Q|$, the number of clauses in the computed rewriting $\mathsf{rew}(Q, \mathcal{T})$ is only exponential w.r.t. $|\mathcal{T}|$ and $|Q|$.

6 Data Complexity Analysis

Query answering is typically considered the most important reasoning task in scenarios where the size of the ABox is considerably larger than the size of the TBox. In this kind of scenario, it is usual to measure the complexity of query answering w.r.t. the size of the ABox only. This notion of complexity is known as *data complexity*, and has been extensively studied in the database community [39]. The data complexity of query answering over DLs has been studied for a large variety of DLs of the DL-Lite and $\mathcal{EL}$ families (see for instance, [13, 36, 26, 27]). We now determine the data complexity of answering conjunctive queries over $\mathcal{ELHIO}^{\neg}$ KBs, and show that $\mathsf{rew}(Q, \mathcal{T})$ is optimal w.r.t. data complexity for various sublanguages of $\mathcal{ELHIO}^{\neg}$.

Theorem 3. *Let* $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ *be an* $\mathcal{ELHIO}$ *KB, and* $Q = \langle Q_P, Q_C \rangle$ *a conjunctive query. Deciding* $\vec{a} \in \mathsf{ans}(Q, \Xi(\mathcal{K}) \cup E'_{\mathcal{T}})$ *is* PTime-*complete w.r.t.* $|\mathcal{A}|$.

Proof. It has been shown in [13] that the problem of instance checking is already PTime-hard if the language in which $\mathcal{K}$ is modeled allows for TBox axioms of the form $\exists P.A \sqsubseteq B$ and $A \sqcap B \sqsubseteq C$. Moreover, it is well known that, given a datalog program DP, deciding whether $DP \models P(\vec{a})$ can be done in PTime w.r.t. to the number of facts in DP [15]. According to Lemma 16 we have that $\mathsf{rew}(Q, \mathcal{T})$ is a datalog query; therefore, since its size does not depend on $\mathcal{A}$, deciding whether $\vec{a} \in \mathsf{ans}(\mathsf{rew}(Q, \mathcal{T}), \mathcal{A})$ is PTime-complete w.r.t. $|\mathcal{A}|$. The claim follows from this fact given Theorem 2. $\square$

As explained in Section 3, verifying whether an $\mathcal{ELHIO}^{\neg}$ $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is unsatisfiable can be reduced to answering a set of queries (that depend on $\mathcal{T}$) over $\langle \mathcal{T}_{\mathrm{PI}}, \mathcal{A} \rangle$. Therefore, as a corollary of Theorem 3, we have that verifying whether an $\mathcal{ELHIO}^{\neg}$ KB $\mathcal{K}$ is unsatisfiable can be done in PTime w.r.t. data complexity. Moreover, as also explained in Section 3, if an $\mathcal{ELHIO}^{\neg}$ KB is satisfiable, then its negative inclusions are not relevant for query answering. Therefore, as another corollary of Theorem 3, we have that given a conjunctive query Q and a satisfiable $\mathcal{ELHIO}^{\neg}$ KB $\mathcal{K}$, deciding $\vec{a} \in \mathsf{ans}(Q, \Xi(\mathcal{K}) \cup E'_{\mathcal{T}})$ is PTime-complete w.r.t. data complexity.

It is well-known that adding other common DL features to $\mathcal{ELHIO}^{\neg}$, such as disjunction or qualified universal quantification, would cause query answering to be coNP-hard w.r.t. data complexity [13]. Therefore, $\mathcal{ELHIO}^{\neg}$ is one of the most expressive languages for which query answering remains tractable w.r.t. data complexity.

We now consider the optimality of the rewriting. Before proceeding, we show that a datalog program DP consisting of a union of conjunctive queries and a

linear datalog program can be evaluated in NLogSpace w.r.t. to the number of facts in DP.

Lemma 18. *Let $Q = \langle Q_P, Q_C \rangle$ be a union of conjunctive queries, DP a linear datalog program. Deciding $DP \cup Q_C \models Q_P(\vec{a})$ can be performed in NLogSpace w.r.t. $|A|$, where A is the set of facts contained in DP.*

Proof. If $DP \cup Q_C \models Q_P(\vec{a})$, then $Q_P(\vec{a})$ can be derived from the set of clauses $DP \cup Q_C$ using SLD resolution [5]. First, we nondeterministically choose a query $Q_i \in Q_C$ and ground it by nondeterministically choosing a set of constants from A. We then initialize the *goal* G to be the resolvent of Q_i and $\leftarrow Q_P(\vec{a})$; if resolution is not possible, the algorithm halts. Then, we start the following loop. We first eliminate all atoms with EDB predicates in G by resolving them with facts in A; if some atom cannot be resolved, the algorithm halts. If G has an empty body, the algorithm accepts. Otherwise, we nondeterministically choose a rule $R \in DP$ and generate its grounding R' by nondeterministically choosing a set of constants from A. Finally, we set our goal G to be the resolvent between R' and G; if this is not possible, the algorithm halts. We now repeat the loop. To ensure termination, we maintain a counter that is initialized in the beginning to the number of ground clauses of DP and A multiplied by the number of the queries in Q_C. We decrease the counter after each pass through the loop, and we terminate the loop if the counter reaches zero. Clearly, if the algorithm accepts, then SLD resolution for $Q_P(\vec{a})$ from $DP \cup Q_C$ exists. Conversely, if an SLD resolution exists, then we can assume that each ground instance of a rule is used only once, so an accepting run of our algorithm exists. It is possible to make such an assumption since w.l.o.g. we can assume that every ground fact occurs on each branch of a derivation tree only once: multiple occurrences of a ground fact would clearly lead to unnecessary cyclic computations.

Since we are interested in in determining the data complexity of this procedure, the number of predicates p and their arity r is bounded. Hence, if A contains c constants, we can describe each ground atom in $p \cdot r \cdot \lceil \log(c) \rceil$ bits. The number of atoms in G depends on the number of clauses in $DP \setminus A \cup Q_C$, so storing G requires $k_1 \lceil \log(c) \rceil$ bits for k_1 a constant that does not depend on c. Finally, the number of ground clauses depends polynomially on c, so we can store the counter using $k_2 \lceil \log(c) \rceil$ bits for k_2 a constant that does not depend on c. Clearly, the algorithm requires $k \lceil \log(c) \rceil$ bits of space in total for k a constant that does not depend on c. The algorithm is nondeterministic, so it can be implemented in NLogSpace. □

Since negative inclusions are not taken into account by our rewriting algorithm, it follows from Lemma 16 that if $\mathcal{T}$ is an $\mathcal{ELHIO}^\neg$ TBox, then $\mathsf{rew}(Q, \mathcal{T})$ is a datalog query; if $\mathcal{T}$ is a DL-Lite$^+$ TBox, then $\mathsf{rew}(Q, \mathcal{T})$ consists of a union of conjunctive queries and a linear datalog query; and if $\mathcal{T}$ is a DL-Lite$_R$ TBox, then $\mathsf{rew}(Q, \mathcal{T})$ is a union of conjunctive queries. A datalog query can be evaluated in PTime w.r.t. data complexity [15]; a union of conjunctive queries with a linear datalog query, in NLogSpace w.r.t. data complexity (Lemma 18); and

a union of conjunctive queries, in LogSpace w.r.t. data complexity [23]. Therefore, our rewriting algorithm not only deals with various DLs from $\mathcal{ELHIO}^{\neg}$ down to DL-Lite$_{core}$, but it is worst-case optimal w.r.t. data complexity for all such logics, which makes it a generalization and an extension of the rewriting approaches of [11], [35], [32], and [33].

Even though data complexity is the most significant complexity measure in scenarios where the size of the ABox dominates, complexity of query answering can also be measured w.r.t. the TBox only, the query only, and all inputs combined. Upper bounds concerning these types of complexity may be derived from our rewriting algorithm. We present these complexity results in Lemma 19.

Lemma 19. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be an $\mathcal{ELHIO}$ KB, and $Q = \langle Q_P, Q_C \rangle$ a conjunctive query. Deciding $\vec{a} \in \mathsf{ans}(Q, \Xi(\mathcal{K}) \cup E'_{\mathcal{T}})$ is in $2\mathrm{EXPTIME}$ w.r.t. $|\mathcal{T}|$, w.r.t. $|Q|$, and w.r.t. $|\mathcal{K}| + |Q|$.*

Proof. According to Lemma 17, the maximal number of clauses produced by our algorithm is $O(2^{t^q})$. Therefore, computing $\mathsf{rew}(Q, \mathcal{T})$ takes an exponential number of steps w.r.t. $|\mathcal{T}|$, and a doubly exponential number of steps w.r.t. $|Q|$, and w.r.t. $|\mathcal{K}| + |Q|$. It follows from [15] that $\mathsf{rew}(Q, \mathcal{T})$ can be evaluated over $\mathcal{A}$ in an exponential number of steps w.r.t. $|\mathsf{rew}(Q, \mathcal{T})|$. By Lemma 17, we have that $|\mathsf{rew}(Q, \mathcal{T})| = O(2^{qt})$. Therefore, evaluating $\mathsf{rew}(Q, \mathcal{T})$ over $\mathcal{A}$ can be done in a doubly exponential number of steps w.r.t. $|\mathcal{T}|$, w.r.t. $|Q|$, and w.r.t. $|\mathcal{K}| + |Q|$. The claim follows from these facts given Theorem 2. $\square$

We believe that can tighten the upper complexity bounds stated by Lemma 19 by adjusting the definition of $\mathcal{N}^{DL}$ to make it as small as possible. Nevertheless, our complexity analysis suggests that our rewriting-based technique is more likely to be useful in practice when dealing with relatively small TBoxes and queries. It is reasonable to assume that queries are small in practice; moreover, applications dealing with enormous databases and small schemas are quite common. Therefore, we expect our rewriting-based query answering technique to be useful in practice for many real scenarios.

7 Future Work

We plan to conduct a more thorough complexity analysis of our technique to derive tighter bounds w.r.t. TBox, query, and combined complexity. Moreover, we plan to develop optimization techniques to reduce the size of the rewriting, and reduce/eliminate recursion when possible. Additionally, we plan to conduct an empirical evaluation of our algorithm. We have already conducted some preliminary experiments with very encouraging results. We have established promising relationships with researchers at the universities of Newcastle and Aberdeen who are using rewriting techniques to answer queries over various different KBs. We plan to use such KBs in our empirical evaluation.

References

1. S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.
2. F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ Envelope. In L. P. Kaelbling and A. Saffiotti, editors, *IJCAI*, pages 364–369. Professional Book Center, 2005.
3. F. Baader and W. Nutt. *Basic Description Logics*, chapter 2, pages 47–100. Cambridge University Press, 2003.
4. F. Baader and W. Snyder. Unification Theory. In Robinson and Voronkov [34], chapter 8, pages 445–532.
5. L. Bachmair and H. Ganzinger. Resolution Theorem Proving. In Robinson and Voronkov [34], chapter 2, pages 19–99.
6. A. Calì. Querying incomplete data with logic programs: Er strikes back. In C. Parent, K.-D. Schewe, V. C. Storey, and B. Thalheim, editors, *ER*, volume 4801 of *Lecture Notes in Computer Science*, pages 245–260. Springer, 2007.
7. A. Calì, G. Gottlob, and M. Kifer. Taming the infinite chase: Query answering under expressive relational constraints. In G. Brewka and J. Lang, editors, *KR*, pages 70–80. AAAI Press, 2008.
8. A. Calì, G. Gottlob, and T. Lukasiewickz. A general datalog-based framework for tractable query answering over ontologies. In *PODS '09: Proceedings of the twenty-seventh ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 69–78, New York, NY, USA, 2009. ACM.
9. A. Calì and M. Kifer. Containment of conjunctive object meta-queries. In U. Dayal, K.-Y. Whang, D. B. Lomet, G. Alonso, G. M. Lohman, M. L. Kersten, S. K. Cha, and Y.-K. Kim, editors, *VLDB*, pages 942–952. ACM, 2006.
10. A. Calì, T. Lukasiewicz, L. Predoiu, and H. Stuckenschmidt. Tightly integrated probabilistic description logic programs for representing ontology mappings. In S. Hartmann and G. Kern-Isberner, editors, *FoIKS*, volume 4932 of *Lecture Notes in Computer Science*, pages 178–198. Springer, 2008.
11. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable Reasoning and Efficient Query Answering in Description Logics: The DL-Lite Family. *Journal of Automated Reasoning*, 9:385–429, 2007.
12. D. Calvanese, E. Franconi, V. Haarslev, D. Lembo, B. Motik, A.-Y. Turhan, and S. Tessaris, editors. *Proceedings of the 2007 International Workshop on Description Logics (DL2007), Brixen-Bressanone, near Bozen-Bolzano, Italy, 8-10 June, 2007*, volume 250 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2007.
13. D. Calvanese, G. D. Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Data Complexity of Query Answering in Description Logics. In P. Doherty, J. Mylopoulos, and C. A. Welty, editors, *KR*, pages 260–270. AAAI Press, 2006.
14. D. Calvanese, G. D. Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Description Logic Framework for Information Integration. In A. G. Cohn, L. K. Schubert, and S. C. Shapiro, editors, *Principles of Knowledge Representation and Reasoning*, pages 2–13. Morgan Kaufmann, 1998.
15. E. Dantsin, T. Eiter, G. Gottlob, and A. Voronkov. Complexity and Expressive Power of Logic Programming. In *IEEE Conference on Computational Complexity*, pages 82–101, 1997.
16. R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data Exchange: Semantics and Query Answering. In D. Calvanese, M. Lenzerini, and R. Motwani, editors, *ICDT*, volume 2572 of *Lecture Notes in Computer Science*, pages 207–224. Springer, 2003.

17. M. Fitting. *First-order logic and automated theorem proving (2nd ed.).* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1996.
18. A. Fuxman and R. J. Miller. First-order query rewriting for inconsistent databases. *J. Comput. Syst. Sci.*, 73(4):610–635, 2007.
19. M. Gelfond and V. Lifschitz. The Stable Model Semantics for Logic Programming. In *Proc. of the 5th Int. Conf. on Logic Programming (ICLP '88)*, pages 1070–1080, Seattler, WA, USA, August 15–19 1988. MIT Press.
20. C. Green. Theorem proving by resolution as a basis for question-answering systems. In B. Meltzer and D. Michie, editors, *4th Annual Machine Intelligence Workshop*, page 183208. Edinburgh University Press, 1969.
21. J. Heflin and J. Hendler. A portrait of the semantic web in action. *IEEE Intelligent Systems*, 16(2):54–59, 2001.
22. W. H. Joyner. Resolution Strategies as Decision Procedures. *J. ACM*, 23(3):398–417, 1976.
23. P. C. Kanellakis, G. M. Kuper, and P. Z. Revesz. Constraint Query Languages. In *Proceedings of the Ninth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, April 2-4, 1990, Nashville, Tennessee*, pages 299–313. ACM Press, 1990.
24. Y. Kazakov. *Saturation-Based Decision Procedures for Extensions of the Guarded Fragment.* PhD thesis, Universität des Saarlandes, Saarbrücken, Germany, March 2006.
25. M. Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and frame-based languages. *J. ACM*, 42(4):741–843, 1995.
26. A. Krisnadhi and C. Lutz. Data complexity in the $\mathcal{EL}$ family of dls. In Calvanese et al. [12].
27. M. Krotzsch and S. Rudolph. Conjunctive Queries for $\mathcal{EL}$ with Role composition. In Calvanese et al. [12].
28. M. Lenzerini. Data Integration: a theoretical perspective. In L. Popa, editor, *PODS '02: Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 233–246, New York, NY, USA, 2002. ACM Press.
29. A. Y. Levy. Answering Queries Using Views: A Survey. In *Very Large Databases Journal*, volume 10, pages 270–294, 2001.
30. C. Lutz, D. Toman, and F. Wolter. Conjunctive Query Answering in the Description Logic EL using a Relational Database System. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence IJCAI09*. AAAI Press, 2009.
31. B. Motik. *Reasoning in Description Logics using Resolution and Deductive Databases.* PhD thesis, Universität Karlsruhe (TH), Karlsruhe, Germany, January 2006.
32. H. Pérez-Urbina, B. Motik, and I. Horrocks. Rewriting Conjunctive Queries over Description Logic Knowledge Bases. In *Proceedings of the International Workshop on Semantics in Data and Knowledge Bases*, March 2008.
33. H. Pérez-Urbina, B. Motik, and I. Horrocks. Rewriting Conjunctive Queries under Description Logic Constraints. In *Proceedings of the International Workshop on Logics in Databases*, May 2008.
34. J. A. Robinson and A. Voronkov, editors. *Handbook of Automated Reasoning (in 2 volumes).* Elsevier and MIT Press, 2001.
35. R. Rosati. On conjunctive query answering in $\mathcal{EL}$. In Calvanese et al. [12].

36. R. Rosati. The Limits of Querying Ontologies. In T. Schwentick and D. Suciu, editors, *ICDT*, volume 4353 of *Lecture Notes in Computer Science*, pages 164–178. Springer, 2007.
37. M. Simkus and T. Eiter. Dnc: Decidable non-monotonic disjunctive logic programs with function symbols. In N. Dershowitz and A. Voronkov, editors, *LPAR*, volume 4790 of *Lecture Notes in Computer Science*, pages 514–530. Springer, 2007.
38. R. van der Meyden. Logical Approaches to Incomplete Information: A Survey. In J. Chomicki and G. Saake, editors, *Logics for Databases and Information Systems*, pages 307–356. Kluwer, 1998.
39. M. Y. Vardi. The Complexity of Relational Query Languages (Extended Abstract). In H. R. Lewis, B. B. Simons, W. A. Burkhard, and L. Landweber, editors, *Proc. of the 14th annual ACM Symposium on Theory of Computing (STOC '82)*, pages 137–146, San Francisco, CA, USA, May 5–7 1982. ACM Press.
40. J. Widom. Research Problems in Data Warehousing. In *CIKM '95, Proceedings of the 1995 International Conference on Information and Knowledge Management, November 28 - December 2, 1995, Baltimore, Maryland, USA*, pages 25–30, 1995.