Rewriting Conjunctive Queries under Description Logic Constraints
(Technical Report)

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. We consider the problems of conjunctive query answering and rewriting under Description Logic constraints. We present a query rewriting algorithm for \(\mathcal{ELHI}\) knowledge bases, and use it to show that query answering in this setting is \(\text{PTime}\)-complete w.r.t. data complexity. We show that our algorithm is worst-case optimal for languages with data complexity of query answering ranging from \(\text{LogSpace}\) to \(\text{PTime}\)-complete.

1 Introduction

Query answering under constraints is the problem of computing the answers to a query over an incomplete database w.r.t. a set of constraints [19]. Since an incomplete database is only partially specified, the task is to compute the tuples that satisfy the query in every database that conforms to the partial specification and satisfies the constraints. Answering conjunctive queries under constraints is also relevant in several other contexts, including information integration [14], data exchange [9], and data warehousing [20].

Query answering under constraints can be solved via query rewriting under constraints: given a query \(Q\) over an incomplete database \(D\), consisting of a set of extensions \(E\) and a set of constraints \(C\), we can compute a query \(Q'\) (which depends on \(Q\) and \(C\)), such that for every set of extensions \(E\), the answers of \(Q\) over \(D\), and the answers of \(Q'\) over \(E\) coincide. This problem has been tackled by several authors (see for example [5]), who have considered standard database constraints, such as inclusion dependencies, functional dependencies, and so on. It is well known that rewriting queries under general constraints is undecidable; therefore, the expressivity of the constraint languages considered is typically restricted in order to achieve decidability.

Description Logics (DLs) [2] 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 represent a given domain in terms of concepts (unary predicates), roles (binary predicates), and individuals (constants). A DL Knowledge Base (KB) consists of a
terminological component \( \mathcal{T} \) called the TBox, and an assertional component \( \mathcal{A} \) called the ABox. In analogy to databases, the TBox can be seen as a conceptual schema and the ABox as a (partial) instantiation of the schema. DL constraints are not required to be acyclic in order for query answering to remain decidable. The use of DLs as constraint languages has already proven to be useful in a variety of scenarios, such as the Semantic Web [16].

Various rewriting techniques under DL constraints have been proposed. Motik [15] presented a resolution-based algorithm for reducing very expressive DL KBs to disjunctive datalog programs. Similarly, Kazakov [13] used saturation-based theorem proving to derive a range of decision procedures for various DLs of the \( \mathcal{EL} \) family of languages [1]. These approaches, however, do not handle conjunctive queries. Conjunctive query rewriting under DL constraints has been considered by Calvanese et al. for the DL-Lite family of languages, for which query answering was shown to be in \( \text{LogSpace} \) w.r.t. data complexity [7]. Similarly, Rosati presented a rewriting algorithm for the \( \mathcal{EL} \) family of languages, and showed that query answering in \( \mathcal{EL} \) and \( \mathcal{ELH} \) is \( \text{PTime} \)-complete w.r.t. data complexity [18].

Although the aforementioned techniques are closely related, they have all been developed for specific DLs. Our goal, however, is to obtain a unified algorithm inspired by the resolution-based techniques presented in [13, 15], that generalizes and extends the results of [7] and [18]. We present a conjunctive query rewriting algorithm for \( \mathcal{ELHI} \) [1] and use it to obtain the novel result that conjunctive query answering for \( \mathcal{ELHI} \) is \( \text{PTime} \)-complete w.r.t. data complexity. \( \mathcal{ELHI} \) is expressive enough to capture qualified universal quantification, as well as transitivity and functionality assertions on roles, via known encodings; thus, it is one of the most expressive Horn DLs for which query answering remains tractable w.r.t. data complexity. In addition we show that, given a conjunctive query \( Q \) and a TBox \( \mathcal{T} \) expressed in a sublanguage \( L \) of \( \mathcal{ELHI} \), our algorithm produces a rewriting that is optimal for \( L \). If \( L \) is a language of the DL-Lite family, then our rewriting is a union of conjunctive queries, as in [7]; if \( L \) is \( \text{DL-Lite}^+ \), then our rewriting consists of a union of conjunctive queries and a linear datalog program, as in [17]; finally, if \( L \) is a language of the \( \mathcal{EL} \) family, then our rewriting is a datalog program, as in [18]. Therefore, our technique not only deals with the full spectrum of DLs from \( \mathcal{ELHI} \) down to \( \text{DL-Lite}_\text{core} \) [7], but is optimal w.r.t. data complexity for all such languages. The initial version of this algorithm was presented in [17], in which we considered \( \text{DL-Lite}^+ \).

2 Preliminaries

2.1 Description Logic \( \mathcal{ELHI} \)

For \( P \) an atomic role, an \( \mathcal{ELHI} \) basic role has the form \( P \) or \( P^- \). For \( A \) an atomic concept, and \( R \) a basic role, an \( \mathcal{ELHI} \) basic concept has the form \( A, \exists R, \exists R.A, \) or \( B_1 \sqcap B_2, A \sqsubseteq B_1 \sqcap B_2, B_1 \sqsubseteq R_1 \sqsubseteq B_2, \) where \( B_1 \) and \( B_2 \) are basic concepts, and \( R_1 \) and \( R_2 \) are basic roles. Without loss of generality we can assume that every axiom in the TBox is in one of the following forms: \( A_1 \sqsubseteq A_2, A_1 \sqcap A_2 \sqsubseteq A_3, A_1 \sqsubseteq \exists R_1, A_1 \sqsubseteq \exists R_1.A_2, \)
Semantics of concepts and roles:

\[ A^2 \subseteq \triangle^2 \]
\[ P^2 \subseteq \triangle^2 \times \triangle^2 \]
\[ (B_1 \cap B_2)^2 = B_1^2 \cap B_2^2 \]
\[ (P^{-})^2 = \{ (x, y) \mid (y, x) \in P^2 \} \]
\[ (\exists R.A)^2 = \{ x \mid \exists y. (x, y) \in R^2 \} \]
\[ (\exists R.A)^2 = \{ x \mid \exists y. (x, y) \in R^2 \land y \in A^2 \} \]

Semantics of assertions:

\[ I \models A(a) \quad \text{iff} \quad a^2 \in A^2 \]
\[ I \models P(a, b) \quad \text{iff} \quad (a^2, b^2) \in P^2 \]
\[ I \models B_1 \sqsubseteq B_2 \quad \text{iff} \quad B_1^2 \subseteq B_2^2 \]
\[ I \models P_1 \sqsubseteq P_2 \quad \text{iff} \quad P_1^2 \subseteq P_2^2 \]

An ABox is a set of membership assertions of the form \( A(a) \) or \( P(a, b) \), where \( A \) is an atomic concept, \( P \) is an atomic role, and \( a \) and \( b \) are constants. An ELHI knowledge base (KB) \( \mathcal{K} \) is a tuple \( (T, A) \), where \( T \) is a TBox and \( A \) is an ABox.

An interpretation \( I = (\triangle^2, \cdot^2) \) consists of a nonempty interpretation domain \( \triangle^2 \) and a function \( \cdot^2 \) that maps each concept \( C \) to a subset \( C^2 \) of \( \triangle^2 \), each role \( R \) to a subset \( R^2 \) of \( \triangle^2 \times \triangle^2 \), and each constant \( a \) to an element \( a^2 \) of \( \triangle^2 \). The function \( \cdot^2 \) is shown in the left part of Table 1. An interpretation \( I \) is a model of an inclusion or membership assertion \( \alpha \), written \( I \models \alpha \), if and only if the conditions shown in the right part of Table 1. An interpretation \( I \) is a model of a KB \( \mathcal{K} = (T, A) \), written \( I \models \mathcal{K} \), if \( I \) satisfies each of the assertions in \( T \) and \( A \). A KB \( \mathcal{K} \) is satisfiable if it has at least one model; furthermore, \( \mathcal{K} \) logically implies an assertion \( \alpha \), written \( \mathcal{K} \models \alpha \), if all models of \( \mathcal{K} \) are also models of \( \alpha \).

### 2.2 Conjunctive and Datalog Queries

We use the well-known notions of a first-order signature, terms, variables, and atoms. A Horn clause is an expression of the form \( H \leftarrow B_1 \land \cdots \land B_m \), where \( H \) is a possibly empty atom and \( \{B_i\} \) is a set of atoms. The atom \( H \) is called the head and the set \( \{B_i\} \) is called the body. With \( \square \) we denote the empty clause. A Horn clause \( C \) is safe if all variables occurring in the head also occur in the body. With \( \text{var}(C) \) we denote the number of variables in a clause \( C \). The depth of a term is defined as:

- \( \text{depth}(t) = 0 \) for \( t \) a constant or a variable,
- \( \text{depth}(f(s_1, \ldots, s_m)) = 1 + \max(\text{depth}(s_i)) \) for a functional term,
- \( \text{depth}(R(t_1, \ldots, t_n)) = \max(\text{depth}(t_1), \ldots, \text{depth}(t_n)) \) for atoms, and
- \( \text{depth}(C) = \max(\text{depth}(H), \text{depth}(B_1), \ldots, \text{depth}(B_m)) \) for Horn clauses.

A datalog program \( P \) is a set of function-free, safe Horn clauses. The extensional database (EDB) predicates of \( P \) are those that do not occur in the head atom of any Horn clause in \( P \); all other predicates are called intensional database (IDB) predicates. The program \( P \) is linear if each Horn clause in \( P \) contains at most one IDB predicate in the body. A datalog query \( Q \) is a tuple \( (Q_P, P) \),
where \( Q_P \) is a \textit{query predicate} and \( P \) is a datalog program. \( Q \) is a \textit{linear datalog query} if \( P \) is a linear datalog program; \( Q \) is called a \textit{union of conjunctive queries} if \( Q_P \) is the only IDB predicate in \( P \) and the body of each clause in \( P \) does not contain \( Q_P \); finally, \( Q \) is a \textit{conjunctive query} if it is a union of conjunctive queries and \( P \) contains exactly one Horn clause. A tuple of constants \( \vec{a} \) is an \textit{answer} of a datalog query \( Q = \langle Q_P, P \rangle \) on an \( \cal{ELHI} \) KB \( \cal{K} = \langle \cal{T}, \cal{A} \rangle \) if and only if \( \cal{K} \cup \cal{P} \models Q_P(\vec{a}) \), where \( P \) is considered to be a set of universally quantified implications with the usual first-order semantics; the set of all answers of \( Q \) on \( \cal{K} \) is denoted by \( \text{ans}(Q, \cal{K}) \).

### 2.3 Resolution with Free Selection

Resolution with free selection is a well-known calculus that can be used to decide satisfiability of a set of Horn clauses [4]. The calculus is parameterized by a \textit{selection function} \( S \) that assigns to each Horn clause \( C \) a nonempty set of atoms such that either \( S(C) = \{H\} \) or \( S(C) \subseteq \{B_i\} \). The atoms in \( S(C) \) are said to be \textit{selected} in \( C \). The resolution calculus with free selection \( \cal{R} \) consists of the following \textit{resolution inference rule}.

\[
\begin{align*}
A &\leftarrow B_1 \land \cdots \land B_i \land \cdots \land B_n \quad C &\leftarrow D_1 \land \cdots \land D_m \\
A\sigma &\leftarrow B_1\sigma \land \cdots \land B_i\sigma \land \cdots \land B_n\sigma \land D_1\sigma \land \cdots \land D_m\sigma
\end{align*}
\]

The two clauses above the inference line are called the \textit{premises} and the clause below is called the \textit{resolvent}. 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 a selection function and \( \sigma = \text{MGU}(B_i, C) \) as defined in [3]. A set of Horn clauses \( N \) is \textit{saturated} by \( \cal{R} \) if, for any two premises \( P_1, P_2 \in N \), the set \( N \) contains a clause that is equivalent to the resolvent of \( P_1 \) and \( P_2 \) up to variable renaming. A \textit{derivation} by \( \cal{R} \) from a set of Horn clauses \( N \) is a sequence of sets of Horn clauses \( N = N_0, N_1, \ldots \) such that, for each \( i \geq 0 \), we have that \( N_{i+1} = N_i \cup \{C\} \), where \( C \) is the conclusion of an inference by \( \cal{R} \) from premises in \( N_i \). The limit \( N_\infty \) of a fair derivation from a set of Horn clauses \( N \) is defined as \( N_\infty = \bigcup_i N_i \). A set of Horn clauses \( N \) is satisfiable if and only if \( \emptyset \notin N_\infty \). A clause \( C \) is said to be \textit{derivable} from \( N \) if \( C \in N_\infty \).

### 3 Answering Conjunctive Queries in \( \cal{ELHI} \)

Given an \( \cal{ELHI} \) TBox \( \cal{T} \) and a conjunctive query \( Q = \langle Q_P, \{Q_C\} \rangle \), our goal is to compute a query \( \text{rew}(Q, \cal{T}) \) such that, for each ABox \( \cal{A} \), evaluating \( \text{rew}(Q, \cal{T}) \) over \( \cal{A} \) and evaluating the query \( Q \) directly over \( \cal{K} = \langle \cal{T}, \cal{A} \rangle \) produces exactly the same answers. We derive this algorithm in two steps. In this section, we first show how to compute the set of answers \( \text{ans}(Q, \cal{K}) \) directly; then, in Section 4 we use this result to derive the rewriting algorithm.

It is well known that \( \vec{a} \in \text{ans}(Q, \cal{K}) \) if and only if \( \Xi(\cal{K}) \cup \{Q_C, \bot \leftarrow Q_P(\vec{a})\} \) is unsatisfiable, where \( \Xi(\cal{K}) \) is the transformation of \( \cal{K} \) into a set of clauses. Therefore, to answer \( Q \) over \( \cal{K} \), we need a decision procedure for checking satisfiability of sets of clauses. We derive this procedure using the principles outlined
with the following selection function

**Definition 1.** Let $S$ be an ELHI knowledge base and $Q = \{Q_P, \{Q_C\}\}$ a conjunctive query. The set of clauses $\Xi(K)$ is obtained by transforming $K$ according to Table 2, then $\Xi(K) \cup \{Q_C\} \subseteq N$. Finally, we saturate $\Xi(K) \cup \{Q_C, P\} \models \{Q_P, \{Q_C\}\}$ using $R^{DL}$—a slightly parameterized resolution with free selection calculus. Since $R^{DL}$ is sound and complete, in order to obtain a decision procedure we only need to show that each saturation terminates. This is done in the key Lemma 1 by showing that $N$ is closed under $R^{DL}$. We now formally define $\Xi, N$, and $R^{DL}$.

**Definition 1.** Let $K$ be an ELHI knowledge base and $Q = \{Q_P, \{Q_C\}\}$ a conjunctive query. The set of clauses $\Xi(K)$ is obtained by transforming $K$ as shown in Table 2. The set of ELHI clauses $N$ is the set of all clauses of types shown in Table 3 constructed using the symbols in $Q_C$ and $K$. With $R^{DL}$ we denote the resolution calculus with free selection parameterized with the following selection function $S$: given a clause $C$, (a) if $C$ is of type $A_2,A_3,T_3$–$T_8$, then $S$ selects the atoms that are underlined in Table 3; (b) if $C$ is of type $A_1, A_1$, or $T_2$, then, if the body is empty or the depth of the head is greater than the maximal depth of the body, $S$ selects the head; otherwise, $S$ selects all deepest body atoms; and (c) if $C$ is of type $Q_1$, then $S$ selects the head if $C$ contains functional terms in the head or if the body is empty; otherwise, $S$ selects all deepest body atoms. $N_{\infty}$ denotes the limit of a fair derivation from a set of clauses $N$ by $R^{DL}$.

**Lemma 1.** For each two clauses $C_1, C_2 \in N$ and $C_r$ the resolvent of $C_1$ and $C_2$ by $R^{DL}$, we have that $C_r \in N$.

**Proof.** It can be seen in Table 4 that if $C_1$ and $C_2$ are of type $A_1$–$T_8$, then $C_r$ is also of type $A_1$–$T_8$. Assume that $C_1$ is of type $Q_1$, satisfying properties (i)–(iii) of Table 3. If the head atom $Q_P(t)$ of $C_1$ is selected, then resolution is not possible, since no clause in $N$ contains $Q_P$ in the body. If a unary body atom $A(t)$ of $C_1$ is selected, then $C_2$ can be of type $A_1$ or $T_2$; we now show that $C_r$ satisfies properties (i)–(iii) of Table 3.

- If $C_2$ is of type $A_1$, unification is possible only if the term $t$ is either a constant $a$ or a variable $y$. In the former case, the unifier $\sigma$ is empty; in the latter case,
Note 2. \(A, B, C, D\) are atomic concepts; \(P, S\) are atomic roles; \(L\) is an atomic concept or role; and \(\vec{u}, \vec{t}\) are tuples of terms. With \(C(t)\) we denote \(\bigwedge_{i=1}^{n} C_i(t)\), and with \(C(\vec{u})\), we denote \(\bigwedge_{i=1}^{n} C_i(\vec{u})\), for some \(n\). With \([A(t)]\) we denote a possible occurrence of the atom \(A(t)\) in a clause. For each clause \(C\) of type other than \(Q_1\), if \(C\) contains a function symbol of the form \(f_A(x)\), then \(C\) contains \(A(x)\) in the body. Each clause \(C\) of type \(Q_1\) has the following properties: (i) \(\text{var}(C) \leq \text{var}(Q_2)\), (ii) \(\text{depth}(C) \leq \max(1, \text{var}(Q_2) - \text{var}(C))\), and (iii) if a variable \(x\) occurs in a functional term in \(C\), then \(x\) occurs in all functional terms in \(C\).

\[\sigma = \{y \mapsto a\}\]. Clearly, \(\text{var}(C_r) \leq \text{var}(C_1)\) and \(\text{depth}(C_r) = \text{depth}(C_1)\), so \(C_r\) satisfies (i) and (ii). Furthermore, since \(A(t)\) is the deepest atom in \(C_1\), the clause \(C_1\) does not contain functional terms, so \(C_r\) does not contain them either; hence, \(C_r\) satisfies (iii) vacuously.

- If \(C_2\) is of type \(T_2\), unification is possible only if the term \(t\) is a variable or a functional term.

\[\sigma = \{y \mapsto f_A(x)\}\]. Clearly, \(\text{var}(C_r) = \text{var}(C_1)\), so \(C_r\) satisfies (i). Furthermore, \(\text{depth}(C_1) = 0\) and \(\text{depth}(C_r) \leq 1\), so \(C_r\) satisfies (ii). Finally, every occurrence of \(y\) is replaced with \(f_A(x)\), and \(C_1\) does not contain functional terms, so (iii) holds as well.

- If \(t\) is a functional term \(f_A(s)\), the unifier is of the form \(\sigma = \{x \mapsto s\}\). Clearly, \(\text{var}(C_r) = \text{var}(C_1)\), so \(C_r\) satisfies (i). Furthermore, since no term in \(C_1\) is deeper than \(f_A(s)\), we have \(\text{depth}(C_r) \leq \text{depth}(C_1)\), so \(C_r\) satisfies (ii). Finally, all the functional terms introduced by the inference share the same variable in \(C_2\), namely \(x\), so \(C_r\) satisfies (iii).

If a binary atom \(P(s, t)\) is selected in \(C_1\), then \(C_2\) can be of type \(T_7\) or \(T_8\). We now show that \(C_r\) satisfies properties (i)–(iii) of Table 3.

- If \(C_2\) is of type \(T_3\), the unification is possible only if the terms \(s\) and \(t\) are not functional terms. If they are both constants, the substitution \(\sigma\) is empty; otherwise, \(\sigma\) maps \(s, t\), or both to the corresponding constants in \(C_2\). Clearly, \(\text{var}(C_r) \leq \text{var}(C_1)\), so \(C_r\) satisfies (i). Furthermore, \(\text{depth}(C_r) = \text{depth}(C_1)\), so \(C_r\) satisfies (ii). Finally, since \(P(s, t)\) is the deepest atom in \(C_1\), the clause \(C_1\) does not contain functional terms, so \(C_r\) satisfies (iii) vacuously.

- If \(C_2\) is of type \(T_7\) (analogous for \(T_8\)), unification is possible only if the term \(t\) is a variable or a functional term.

\[\sigma = \{x_t \mapsto f_A(s), x \mapsto s\}\]. Due to the occurrence check in unification, \(x_t\) cannot occur in \(s\). The inference thus decreases the number of variables of \(C_1\) in \(C_r\) by one: \(\text{var}(C_r) = \text{var}(C_1) - 1\), so

<table>
<thead>
<tr>
<th>Type</th>
<th>(\mathcal{ELHI}) clause</th>
<th>Type</th>
<th>(\mathcal{ELHI}) clause</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>(B(a) \leftarrow A(a))</td>
<td>T4</td>
<td>(B(x) \leftarrow P(y, x) \land [A(y)])</td>
</tr>
<tr>
<td>A2</td>
<td>(B(a) \leftarrow A(b))</td>
<td>T5</td>
<td>(S(x, y) \leftarrow P(x, y))</td>
</tr>
<tr>
<td>A3</td>
<td>(P(a, b))</td>
<td>T6</td>
<td>(S(x, y) \leftarrow P(y, x))</td>
</tr>
<tr>
<td>T1</td>
<td>(D(x) \leftarrow A(x) \land B(f_2(x)))</td>
<td>T7</td>
<td>(P(x, f_b(x)) \leftarrow A(x))</td>
</tr>
<tr>
<td>T2</td>
<td>(f_2(x) \leftarrow A(x) \land B(f_2(x)))</td>
<td>T8</td>
<td>(P(f_b(x), x) \leftarrow A(x))</td>
</tr>
<tr>
<td>T3</td>
<td>(B(x) \leftarrow P(x, y) \land [A(y)])</td>
<td>Q1</td>
<td>(Q_f(\vec{u}) \leftarrow L(\vec{t}))</td>
</tr>
</tbody>
</table>
In this section, we present an algorithm for query rewriting. Given a conjunctive query \(Q\) and a TBox \(\mathcal{T}\), we compute a datalog query \(\text{rew}(Q, \mathcal{T})\) such that, for any ABox \(\mathcal{A}\), the sets of answers of \(Q\) over \(\langle \mathcal{T}, \mathcal{A}\rangle\) and of \(\text{rew}(Q, \mathcal{T})\) over \(\mathcal{A}\) are the same. Our goal is to produce an optimal rewriting for all sublanguages of \(\mathcal{ELHI}\): if \(\mathcal{T}\) is in a language between DL-Lite\(_\text{core}\) and DL-Lite\(_R\), then \(\text{rew}(Q, \mathcal{T})\) should be a union of conjunctive queries; if \(\mathcal{T}\) is in DL-Lite\(_R^+\), then \(\text{rew}(Q, \mathcal{T})\) should consist of a union of conjunctive queries and a linear datalog program; finally, if \(\mathcal{T}\) is in a language between \(\mathcal{EL}\) and \(\mathcal{ELHI}\), then \(\text{rew}(Q, \mathcal{T})\) should be a datalog query.

We derive the rewriting algorithm in two phases: we first show how to convert \(\Xi(\mathcal{T})\) into a nonoptimal datalog program by eliminating function symbols; then, we present an additional step to obtain rewritings of optimal form.

### 4 Rewriting Conjunctive Queries in \(\mathcal{ELHI}\)

In this section, we present an algorithm for query rewriting: given \(Q\) and an \(\mathcal{ELHI}\) TBox \(\mathcal{T}\), we compute a datalog query \(\text{rew}(Q, \mathcal{T})\) such that, for any ABox \(\mathcal{A}\), the sets of answers of \(Q\) over \(\langle \mathcal{T}, \mathcal{A}\rangle\) and of \(\text{rew}(Q, \mathcal{T})\) over \(\mathcal{A}\) are the same. Our goal is to produce an optimal rewriting for all sublanguages of \(\mathcal{ELHI}\): if \(\mathcal{T}\) is in a language between DL-Lite\(_\text{core}\) and DL-Lite\(_R\), then \(\text{rew}(Q, \mathcal{T})\) should be a union of conjunctive queries; if \(\mathcal{T}\) is in DL-Lite\(_R^+\), then \(\text{rew}(Q, \mathcal{T})\) should consist of a union of conjunctive queries and a linear datalog program; finally, if \(\mathcal{T}\) is in a language between \(\mathcal{EL}\) and \(\mathcal{ELHI}\), then \(\text{rew}(Q, \mathcal{T})\) should be a datalog query.

We derive the rewriting algorithm in two phases: we first show how to convert \(\Xi(\mathcal{T})\) into a nonoptimal datalog program by eliminating function symbols; then, we present an additional step to obtain rewritings of optimal form.

#### 4.1 Elimination of Function Symbols

The following definition summarizes the first step of our rewriting algorithm.

**Definition 2.** For \(Q = \langle Q_P, \{Q_C\} \rangle\) a conjunctive query and \(\mathcal{T}\) an \(\mathcal{ELHI}\) TBox, \(\mathcal{ff}(Q, \mathcal{T})\) is the set that contains exactly all function-free clauses contained in \(\langle \Xi(\mathcal{T}) \cup \{Q_C\} \rangle_\mathcal{\infty}\).

We now show that, for each ABox \(\mathcal{A}\), we have \(\mathcal{T} \cup \{Q_C\} \cup \mathcal{A} \models Q_P(\bar{a})\) if and only if \(\mathcal{ff}(Q, \mathcal{T}) \cup \mathcal{A} \models Q_P(\bar{a})\), which makes \(\mathcal{ff}(Q, \mathcal{T})\) a rewriting of \(Q\) w.r.t. \(\mathcal{T}\), albeit not necessarily an optimal one. We prove the claim proof-theoretically: we show that \(Q_P(\bar{a})\) is derivable from \(\Xi(\mathcal{T}) \cup \{Q_C\} \cup \mathcal{A}\) if and only if it is derivable from \(\mathcal{ff}(Q, \mathcal{T}) \cup \mathcal{A}\). To this end, we first need to show that we can
### Table 4: Inferences of $\mathcal{R}^{DL}$ on $\mathcal{N}$

<table>
<thead>
<tr>
<th>Inference</th>
<th>Rule 1</th>
<th>Rule 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_1 + A_1 = A_1$:</td>
<td>$A(a) \quad C(a) \leftarrow A(a) \land B(a)$</td>
<td>$A(a) \quad B(b) \leftarrow A(a)$</td>
</tr>
<tr>
<td>$A_1 + T_1 = A_1$:</td>
<td>$A(a) \quad C(x) \leftarrow A(x) \land B(x)$</td>
<td>$A(a) \quad P(a, b) \leftarrow B(x) \land [A(y)]$</td>
</tr>
<tr>
<td>$A_3 + T_5 = A_3$:</td>
<td>$P(a, b) \quad S(x, y) \leftarrow P(x, y)$</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inference</th>
<th>Rule 1</th>
<th>Rule 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_1 + T_2 = T_2$:</td>
<td>$C(x) \leftarrow A(x) \land B(x)$</td>
<td>$A(f_1(x)) \leftarrow F(x)$</td>
</tr>
<tr>
<td>$T_1 + T_2 = T_1$:</td>
<td>$E(x) \leftarrow B(f_1(x)) \land C(f_1(x)) \land D(x)$</td>
<td>$B(f_1(x)) \leftarrow G(x)$</td>
</tr>
<tr>
<td>$T_2 + T_2 = T_2$:</td>
<td>$E(f_1(x)) \leftarrow B(f_1(x)) \land C(f_1(x)) \land D(x)$</td>
<td>$B(f_1(x)) \leftarrow F(x)$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inference</th>
<th>Rule 1</th>
<th>Rule 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_3 + T_7 = T_1$:</td>
<td>$B(x) \leftarrow P(x, y) \land [C(y)]$</td>
<td>$P(x, f_1(x)) \leftarrow A(x)$</td>
</tr>
<tr>
<td>$T_3 + T_8 = T_2$:</td>
<td>$B(x) \leftarrow P(x, y) \land [C(y)]$</td>
<td>$P(f_1(x), x) \leftarrow A(x)$</td>
</tr>
<tr>
<td>$T_5 + T_7 = T_7$:</td>
<td>$S(x, y) \leftarrow P(x, y)$</td>
<td>$P(x, f_1(x)) \leftarrow A(x)$</td>
</tr>
<tr>
<td>$T_5 + T_8 = T_8$:</td>
<td>$S(x, y) \leftarrow P(x, y)$</td>
<td>$P(f_1(x), x) \leftarrow A(x)$</td>
</tr>
</tbody>
</table>

### Note 3

The notation $A + B = C$ denotes that “resolving a clause of type $A$ with a clause of type $B$ produces a clause of type $C$.” For simplicity we omit analogous inferences with inverses.
Lemma 2. Let \( Q = \langle Q_P, \{ Q_C \} \rangle \) be a conjunctive query, \( \mathcal{T} \) an \( \mathcal{ELHI} \) TBox, and \( \mathcal{A} \) an ABox. For each clause \( \mathcal{C} \) of type Q1 that is derivable from \( \Xi(\mathcal{T}) \cup \{ Q_C \} \cup \mathcal{A} \), a clause \( \mathcal{C}' \) of type Q1 is derivable from \( \Xi(\mathcal{T}) \cup \{ Q_C \} \) such that, for \( G \) the subset of all clauses of type A1 and A3 in \( (\mathcal{W}(Q, \mathcal{T}) \cup \mathcal{A})_{\inf} \), we have \( \{ \mathcal{C}' \} \cup G \vdash \mathcal{C} \).

Proof. We prove the claim by induction on the height of a derivation tree by which \( \mathcal{C} \) is derived from \( \Xi(\mathcal{T}) \cup \{ Q_C \} \cup \mathcal{A} \). If the derivation tree has height zero, then \( \mathcal{C} \) must be the clause \( Q_C \), so the claim follows trivially for \( \mathcal{C}' = Q_C \).

Assume that the claim holds for each clause derived from \( \Xi(\mathcal{T}) \cup \{ Q_C \} \cup \mathcal{A} \) by a derivation tree of height \( n \), and consider a clause \( \mathcal{C} \) derived by a derivation tree of height \( n + 1 \). The clause \( \mathcal{C} \) is obtained by resolving some clauses \( \mathcal{C}_1 \) and \( \mathcal{C}_2 \). According to Table 4, one of the premises has to be of type Q1, so we denote it by \( \mathcal{C}_1 \); the other premise \( \mathcal{C}_2 \) can be of type A1, A3, T2, T7 or T8. By the induction hypothesis, some clause \( \mathcal{C}'_1 \) of type Q1 is derivable from \( \Xi(\mathcal{T}) \cup \{ Q_C \} \) such that \( \{ \mathcal{C}'_1 \} \cup G \vdash \mathcal{C}_1 \). We now consider the different forms that \( \mathcal{C}_2 \) can have.

Assume that \( \mathcal{C}_2 \) is of type A1 or A3. From Table 4 we can see that each derivation of a clause of type A1 or A3 involves only function-free clauses, so \( \mathcal{C}_2 \in \mathcal{G} \). The inductive claim now trivially holds for \( \mathcal{C}' = \mathcal{C}'_1 \).

Assume that \( \mathcal{C}_2 \) is of type T2, T7 or T8—that is, of the form \( \mathcal{L}(\overline{v}) \leftarrow \mathcal{H}(\overline{h}) \). By examining the inferences between \( \mathcal{ELHI} \) clauses shown in Table 4, we can see that \( \mathcal{C}_2 \) is derivable from \( \Xi(\mathcal{T}) \). Note that \( \mathcal{G} \) contains only ground clauses of types A1 and A3; thus, since \( \{ \mathcal{C}'_1 \} \cup G \vdash \mathcal{C}_1 \), a subset \( \{ \mathcal{G}_i(\overline{a}_i) \} \subseteq G \) exists, such that resolving \( \mathcal{C}'_1 \) on a conjunction \( \mathcal{G}(\overline{g}) \) of its body atoms with the elements of \( \{ \mathcal{G}_i(\overline{a}_i) \} \) produces \( \mathcal{C}_1 \). Furthermore, all such resolution inferences just remove body atoms; therefore, if \( \mathcal{C}_1 \) is to contain the atom \( \mathcal{L}(\overline{q}) \) in the body, the clause \( \mathcal{C}'_1 \) must contain an atom \( \mathcal{L}(\overline{s}) \) in its body. Hence, \( \mathcal{C}'_1 \) is of the form (1), and \( \mathcal{C}_1 \) is of the form (2), where \( \delta \) maps some variables to constants occurring in \( \{ \mathcal{G}_i(\overline{a}_i) \} \) such that \( \mathcal{L}(\overline{s}) \delta = \mathcal{L}(\overline{q}) \). Finally, resolving \( \mathcal{C}_1 \) and \( \mathcal{C}_2 \) produces the clause \( \mathcal{C} \), which is of the form (3) for \( \sigma = \text{MGU}(\mathcal{L}(\overline{s}) \delta, \mathcal{L}(\overline{t})) \).

\[
\begin{align*}
\mathcal{C}'_1 &= \mathcal{Q}_P(\overline{v}) \leftarrow \mathcal{L}(\overline{s}) \land \mathcal{G}(\overline{g}) \land \mathcal{M}(\overline{m}) & (1) \\
\mathcal{C}_1 &= \mathcal{Q}_P(\overline{v}) \delta \leftarrow \mathcal{L}(\overline{s}) \delta \land \mathcal{M}(\overline{m}) \delta & (2) \\
\mathcal{C} &= \mathcal{Q}_P(\overline{v}) \delta \sigma \leftarrow \mathcal{H}(\overline{h}) \sigma \land \mathcal{M}(\overline{m}) \delta \sigma & (3)
\end{align*}
\]

Note that no inference used to derive \( \mathcal{C}_1 \) changes the number of function symbols of \( \mathcal{C}'_1 \); therefore, \( \mathcal{L}(\overline{s}) \) is deepest in \( \mathcal{C}'_1 \). Furthermore, each variable of \( \mathcal{C}'_1 \) that is replaced by a constant clearly does not occur in \( \mathcal{L}(\overline{s}) \delta \); hence, the substitutions \( \delta \) and \( \sigma \) have disjoint domains, and \( \sigma = \delta \sigma \). We now transform this derivation into a derivation in which all inferences with ABox clauses are performed after all inferences with only TBox clauses. Let \( \mathcal{C}' \) be the clause obtained by resolving \( \mathcal{C}'_1 \) and \( \mathcal{C}_2 \); we can assume that \( \mathcal{C}' \) has the form (4),
where \( \sigma' = \text{MGU}(L(\bar{s}), L(\bar{t})) \).

\[
C' = Q_P(\bar{a})\sigma' \leftarrow H(\bar{h})\sigma' \land G(\bar{g})\sigma' \land M(\bar{m})\sigma'
\]  

(4)

Since \( L(\bar{s}) \) is the deepest literal of \( C'_1 \), the inference between \( C'_1 \) and \( C_2 \) satisfies the selection function of the calculus \( R^{DL} \). Since both \( C'_1 \) and \( C_2 \) are derivable from \( \Xi(T) \cup \{Q_C\} \), the clause \( C' \) is derivable from \( \Xi(T) \cup \{Q_C\} \) as well. Let \( x \) be a variable that occurs in \( L(\bar{s}) \) and is replaced by \( \delta \) with a constant. Clearly, \( \sigma \) does not contain such \( x \); hence, without loss of generality, we can assume (*) that \( \sigma' \) does not contain such variables either—that is, instead of mapping \( x \) to a term in \( L(\bar{t}) \), we can assume that the corresponding term is mapped to \( x \).

Let \( D \) now be the clause obtained by resolving the conjunction of literals \( G(\bar{g})\sigma' \) in \( C' \) with the ground clauses \( \{G_i(a_i)\} \). The clause \( D \) has the following form, where \( \delta' \) maps some variables of \( C' \) to constants.

\[
D = Q_P(\bar{a})\sigma' \leftarrow H(\bar{h})\sigma' \delta' \land M(\bar{m})\sigma' \delta'
\]  

(5)

Due to (*), \( \sigma \) and \( \sigma' \) have the same domain which is disjoint with the domain of \( \delta \), so \( \sigma = \sigma'\delta' \). None of the variables occurring in \( G(\bar{g}) \) is in the domain of \( \sigma' \), so \( \delta = \delta' \). Since \( \sigma = \sigma'\delta' \) and \( \delta = \delta' \), we have \( \sigma = \sigma'\delta' \). Moreover, since \( \sigma = \delta\sigma \), we have \( \sigma = \delta\sigma = \sigma'\delta' \), so \( C = D \), which proves our claim. \( \square \)

This lemma now allows us to prove the desired relationship between the answers of \( Q \) over \( T \) and \( \mathcal{A} \), and the answers of \( ff(Q,T) \) over \( \mathcal{A} \). Clearly, if we assume that \( Q_P(\bar{a}) \) is derivable from \( \Xi(T) \cup \{Q_C\} \cup \mathcal{A} \), then, since \( Q_P(\bar{a}) \) is of type \( Q_1 \), by Lemma 2, a clause \( C' \) of type \( Q_1 \) is derivable from \( \Xi(T) \cup \{Q_C\} \) such that \( \{C'\} \cup \mathcal{A} \models C \). Since \( Q_P(\bar{a}) \) does not contain function symbols, \( C' \) cannot contain function symbols either, so \( C' \in ff(Q,T) \). Thus, \( Q_P(\bar{a}) \) is implied by \( ff(Q,T) \cup \mathcal{A} \) so, by the definition of \( G \), we have \( ff(Q,T) \cup \mathcal{A} \models Q_P(\bar{a}) \). Therefore, Lemma 3 easily follows.

**Lemma 3.** Let \( Q = \{Q_P, \{Q_C\}\} \) be a conjunctive query, \( T \) an \( ELHI \) TBox, and \( \mathcal{A} \) an ABox. Then, \( \bar{a} \in \text{ans}(Q, \langle T, \mathcal{A} \rangle) \) if and only if \( ff(Q,T) \cup \mathcal{A} \models Q_P(\bar{a}) \).

### 4.2 Optimizing the Program through Unfolding

According to Lemma 3, the datalog program \( ff(Q,T) \) is a rewriting of \( Q \) w.r.t. \( T \). We note, however, that it is not necessarily optimal for TBoxes of DLs for which query answering is in NLogSpace w.r.t. data complexity. We illustrate our point with a simple example. Consider the following DL-Lite TBox \( T \) and its translation \( \Xi(T) \) to clauses:

<table>
<thead>
<tr>
<th>( \Xi(T) )</th>
<th>( T )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \exists \text{hasParent}.\text{Human} \sqsubseteq \text{Human} )</td>
<td>( \exists \text{hasParent}.\text{Human} \sqsubseteq \text{Human} )</td>
</tr>
<tr>
<td>( \text{hasMother} \sqsubseteq \text{hasParent} )</td>
<td>( \text{hasMother} \sqsubseteq \text{hasParent} )</td>
</tr>
</tbody>
</table>

\( \text{Human}(x) \leftarrow \text{hasParent}(x,y) \land \text{Human}(y) \)  

(1)

\( \text{hasParent}(x,y) \leftarrow \text{hasMother}(x,y) \)  

(2)

Given the query \( Q = \{Q_P, \{Q_P(x) \leftarrow \text{Human}(x)\}\} \), one can easily verify that \( ff(Q,T) = \Xi(T) \cup \{Q_P(x) \leftarrow \text{Human}(x)\} \). This is so because such a set does not contain functional symbols and it is already saturated by \( R^{DL} \). It follows
from [17] that in the case of DL-Lite\(^+\), a worst-case optimal rewriting consists of a linear datalog program and a union of conjunctive queries. In this case, however, predicates Human and hasParent are IDB predicates in \(\Xi(T)\); therefore, (1) is not linear, which means that \(\text{ff}(Q, T)\) is not an optimal rewriting of \(Q\) w.r.t. \(T\).

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

**Definition 3.** The unfolding of \(L(\vec{x}) \leftarrow M(\vec{m})\) in \(N(\vec{n}) \leftarrow L(\vec{x}) \land P(\vec{p})\) is the clause \(N(\vec{n})\sigma \leftarrow M(\vec{m})\sigma \land P(\vec{p})\sigma\), where \(\sigma = \text{MGU}(L(\vec{x}), L(\vec{x}))\). Given two sets of safe Horn clauses \(R\) and \(U\), let \(R_U\) be the smallest set such that \(R \subseteq R_U\) and, for each unfolding \(C_r\) of a clause \(C_1 \in R \cap U\) in a clause \(C_2 \in R\), we have that \(C_r \in R_U\). The unfolding of \(R\) w.r.t. \(U\) is defined as \(\text{unfold}(R, U) = R_U \setminus U\).

Given an \(\mathcal{ELHI}\) clause type \(T\), with \(T^*\) we denote all clauses of type \(T\) with at most one body atom.

The rewriting \(\text{rew}(Q, T)\) of a conjunctive query \(Q = \langle Q_P, \{Q_C\} \rangle\) w.r.t. an \(\mathcal{ELHI}\) TBox \(T\) is the query \(\langle Q_P, \text{unfold}(R, U) \rangle\), where \(R = \text{ff}(Q, T)\) and \(U\) is the subset of \(\mathcal{N}\) of all clauses of types \(T1^*, T3^*, T4^*, T5, T6\).

It was shown in [17] that given two sets of clauses \(R\) and \(U\), for any set of facts \(A\) and any predicate \(F\) that does not occur in \(U\), we have \(R \cup A \models F(\vec{m})\) if and only if \(\text{unfold}(R, U) \cup A \models F(\vec{m})\). Theorem 1 follows from this fact, given Lemma 3, and that we can assume that \(Q_P\) does not occur in \(\Xi(T)\).

**Theorem 1.** For a conjunctive query \(Q\), an \(\mathcal{ELHI}\) TBox \(T\), and an ABox \(A\), we have \(\text{ans}(Q, \langle T, A \rangle) = \text{ans}(\text{rew}(Q, T), A)\).

We now prove important properties about the structure of the rewriting. First, we note that since \(\text{ff}(Q, T)\) is function-free and the unfolding phase does not introduce functional terms, Lemma 4 trivially follows.

**Lemma 4.** For \(Q = \langle Q_P, \{Q_C\} \rangle\) a conjunctive query and \(T\) an \(\mathcal{ELHI}\) TBox, \(\text{rew}(Q, T)\) is a datalog query.

**Lemma 5.** For \(Q = \langle Q_P, \{Q_C\} \rangle\) a conjunctive query and \(T\) a DL-Lite\(^+\) TBox, and \(\text{rew}(Q, T) = \langle Q_P, P \rangle\), the program \(P\) can be split into disjoint subsets \(U_Q\) and \(U_C\) such that \(\langle Q_P, U_Q\rangle\) is a union of conjunctive queries and \(\langle Q_P, U_C\rangle\) is a linear datalog query.

**Proof.** Let \(U_Q \subseteq P\) be the set of all clauses of type \(Q1\) in \(P\). Clearly, \(\langle Q_P, U_Q\rangle\) is a union of conjunctive queries. It follows from Table 2 that \(\Xi(T)\) only contains clauses of types \(T1^*, T2^*, T3\), \(T5\), and \(T7\). By analyzing the inferences shown in Table 4, one can see that saturating a set of clauses of these types by \(R^{DL}\) produces only clauses of types \(T1, T2^*, T3, T5\), and \(T7\). Therefore, \(\text{ff}(Q, T)\) only contains clauses of types \(T1^*, T3, T5\), and \(Q1\). Let \(U_C = P \setminus U_Q\). The datalog program \(U_C\) is obtained by unfolding clauses of types \(T1^*, T3^*,\) and \(T5\) in \(\text{ff}(Q, T)\), and then by removing all clauses of such types and of type \(Q1\). Thus, \(U_C\) contains only clauses of type \(T3\) and clauses of the form \(B(x) \leftarrow P(x, y) \land S(y, z)\) that are obtained by unfolding a clause of type \(T3^*\) in a clause of type \(T3\). Since
no clause in $U_C$ contains a role predicate in the head, all role predicates are EDB predicates. Moreover, since clauses of type T3 can contain a unary predicate in the head, unary predicates can be IDB predicates. Nevertheless, unary predicates can only occur in the body of clauses of type T3, so all such clauses are linear. Thus, $U_C$ is a linear datalog program.

**Lemma 6.** For $Q = \langle Q_P, \{Q_C\} \rangle$ a conjunctive query and $T$ a DL-Lite$\mathcal{R}$ TBox, $\mathit{rew}(Q, T)$ is a union of conjunctive queries.

**Proof.** It follows from Table 2 that $\Xi(T)$ only contains clauses of types $T1^*$, $T3^*$, $T4^*$, $T5$, $T6$, $T7$ and $T8$. By analyzing the inferences shown in Table 4, one can see that saturating a set of clauses of these types by $\mathcal{R}^D$ produces only clauses of types $T1^*$, $T2^*$, $T3^*$, $T4^*$, $T5$, $T6$, $T7$ and $T8$. Therefore, $\mathit{ff}(Q, T)$ only contains clauses of types $T1^*$, $T3^*$, $T4^*$, $T5$, $T6$, and $Q1$. These are precisely the types of clauses that are to be unfolded; therefore, all clauses in $\mathit{ff}(Q, T)$ that are not of type $Q1$ are unfolded in clauses of type $Q1$, which immediately means that $\mathit{rew}(Q, T)$ is a union of conjunctive queries. $\square$

### 5 Complexity Analysis

In this section, we determine the data complexity of answering conjunctive queries over $\mathcal{ELHI}$ KBs, and show that our algorithm produces worst-case optimal rewritings for all the subsets of $\mathcal{ELHI}$ with query answering ranging from $\text{LogSpace}$ to PTime-complete. According to [6], checking entailment of a ground concept assertion is PTime-hard if we allow for assertions of the form $\exists P.A \sqsubseteq B$ and $A \sqcap B \sqsubseteq C$. Moreover, it is well known that deciding if $P \cup A \models Q_P(\bar{a})$ for a datalog program $P$, a set of facts $A$, and a tuple of constants $\bar{a}$, can be performed in PTime in the size of $A$ [8]. Therefore, Theorem 2 follows given Theorem 1, Lemma 4, and that the size of $\mathit{rew}(Q, T)$ does not depend on the size of $A$.

**Theorem 2.** For a conjunctive query $Q = \langle Q_P, \{Q_C\} \rangle$ and an $\mathcal{ELHI}$ knowledge base $K = \langle T, A \rangle$, deciding whether $\bar{a} \in \mathit{ans}(Q, K)$ is PTime-complete w.r.t. data complexity.

We now discuss the optimality of our rewriting. First, by Lemma 4, if $T$ is an $\mathcal{EL}$ TBox, then $\mathit{rew}(Q, T)$ is a datalog query; therefore, we can evaluate $\mathit{rew}(Q, T)$ in PTime [8], just as in [18]. Similarly, by Lemma 5, if $T$ is a DL-Lite$^+$ TBox, then $\mathit{rew}(Q, T)$ consists of a union of conjunctive queries and a linear datalog program; therefore, we can evaluate $\mathit{rew}(Q, T)$ in NLogSpace [17], just as in [17]. Finally, by Lemma 6, if $T$ is a DL-Lite$\mathcal{R}$ TBox, then $\mathit{rew}(Q, T)$ is a union of conjunctive queries; therefore, we can evaluate $\mathit{rew}(Q, T)$ in LogSpace [12], just as in [7]. Summing up, our algorithm deals with the full spectrum of languages from DL-Lite$\mathcal{R}$ to $\mathcal{ELHI}$, which includes DL-Lite$^+$ and $\mathcal{EL}$. Furthermore, it is optimal w.r.t. data complexity for all such languages, which makes it a generalization of the rewriting algorithms of Rosati [18], Pérez-Urbina et al. [17], and Calvanese et al. [7].
6 Future Work

We plan to extend the technique to deal with more expressive DLs, and in particular with an extended version of $\mathcal{ELHI}$ including nominals. Finally, we plan to implement our query answering technique in a prototype Information Integration system—we have established a promising relationship with researchers at the University of Newcastle who are using Information Integration in their ComparaGRID project,\(^1\) and we plan to use ComparaGRID as an evaluation framework for our prototype system.

References


\(^1\) http://www.comparagrid.org