Rewriting Conjunctive Queries 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. 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 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 LogSpace to 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 [20]. 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 [21].

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 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 PTime-complete w.r.t. data complexity [19].

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 [19]. 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 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 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 [19]. Therefore, our technique not only deals with the full spectrum of DLs from $\mathcal{ELHI}$ down to DL-Lite$_{core}$ [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 DL-Lite$^+$.

2 Preliminaries

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 *TBox* is a set of *inclusion assertions* or axioms of the form $B_1 \sqsubseteq B_2$ or $R_1 \sqsubseteq R_2$, 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: $A_1 \sqsubseteq A_2$, $A_1 \sqcap A_2 \sqsubseteq A_3$, $A_1 \sqsubseteq \exists R_1$, $A_1 \sqsubseteq \exists R_1.A_2$, $\exists R_1 \sqsubseteq A_1$, $\exists R_1.A_1 \sqsubseteq A_2$, and $R_1 \sqsubseteq R_2$, where each A_i is an atomic concept, and R_i is a basic role [1]. DL-Lite$^+$

Table 1: Semantics of $\mathcal{ELHI}$

Semantics of concepts and roles:	Semantics of assertions:	
$A^{\mathcal{I}} \subseteq \triangle^{\mathcal{I}}$		
$P^{\mathcal{I}} \subseteq \triangle^{\mathcal{I}} \times \triangle^{\mathcal{I}}$	$\mathcal{I} \models A(a)$	iff $a^{\mathcal{I}} \in A^{\mathcal{I}}$
$(B_1 \sqcap B_2)^{\mathcal{I}} = B_1^{\mathcal{I}} \cap B_2^{\mathcal{I}}$	$\mathcal{I} \models P(a,b)$	iff $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in P^{\mathcal{I}}$
$(P^-)^{\mathcal{I}} = \{\langle x,y\rangle \mid \langle y,x\rangle \in P^{\mathcal{I}}\}$	$\mathcal{I} \models B_1 \sqsubseteq B_2$	iff $B_1^{\mathcal{I}} \subseteq B_2^{\mathcal{I}}$
$(\exists R)^{\mathcal{I}} = \{x \mid \exists y.\langle x,y\rangle \in R^{\mathcal{I}}\}$	$\mathcal{I} \models P_1 \sqsubseteq P_2$	iff $P_1^{\mathcal{I}} \subseteq P_2^{\mathcal{I}}$
$(\exists R.A)^{\mathcal{I}} = \{x \mid \exists y.\langle x,y\rangle \in R^{\mathcal{I}} \wedge y \in A^{\mathcal{I}}\}$		

is obtained from $\mathcal{ELHI}$ by disallowing inverse roles and axioms of the form $A_1 \sqcap A_2 \sqsubseteq A_3$. DL-Lite$_R$ is obtained from $\mathcal{ELHI}$ by disallowing concepts of the form $\exists R_1.A_1$ and axioms of the form $A_1 \sqcap A_2 \sqsubseteq A_3$.

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 $\mathcal{ELHI}$ *knowledge base* (KB) $\mathcal{K}$ is a tuple $\langle \mathcal{T}, \mathcal{A}\rangle$, where $\mathcal{T}$ is a 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 concept C to a subset $C^{\mathcal{I}}$ of $\triangle^{\mathcal{I}}$, each role R to a subset $R^{\mathcal{I}}$ of $\triangle^{\mathcal{I}} \times \triangle^{\mathcal{I}}$, and each constant a to an element $a^{\mathcal{I}}$ of $\triangle^{\mathcal{I}}$. The function $\cdot^{\mathcal{I}}$ is shown in the left part 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 part 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 each of the assertions in $\mathcal{T}$ and $\mathcal{A}$. A KB $\mathcal{K}$ is *satisfiable* if it has at least one model; furthermore, $\mathcal{K}$ *logically implies* an assertion α, written $\mathcal{K} \models \alpha$, if all models of $\mathcal{K}$ are also models of α.

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 $H \leftarrow B_1 \wedge \cdots \wedge B_m$, 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 $\mathsf{var}(C)$ we denote the number of variables in a clause C. The *depth* of a term is defined as $\mathsf{depth}(t) = 0$ for t a constant or a variable, and $\mathsf{depth}(f(s_1,\ldots,s_m)) = 1 + \max(depth(s_i))$; for atoms we have $\mathsf{depth}(R(t_1,\ldots,t_n)) = \max(\mathsf{depth}(t_1),\ldots,\mathsf{depth}(t_n))$; and for Horn clauses, $\mathsf{depth}(C) = \max(\mathsf{depth}(H), \mathsf{depth}(B_1),\ldots,\mathsf{depth}(B_m))$.

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 $\langle Q_P, P\rangle$, where Q_P is a *query predicate* and P is a datalog program. Q is a *linear datalog query* if P is a linear datalog program; Q is called a *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 *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 *answer* of a datalog query $Q = \langle Q_P, P\rangle$ on an $\mathcal{ELHI}$ KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A}\rangle$ if and only

if $\mathcal{K} \cup P \models Q_P(\vec{a})$, 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 $\mathcal{K}$ is denoted by $\mathsf{ans}(Q, \mathcal{K})$.

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 *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 *selected* in C. The resolution calculus with free selection $\mathcal{R}$ consists of the following *resolution inference rule*.

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

The two clauses above the inference line are called the *premises* and the clause below is called the *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 $\sigma = \mathsf{MGU}(B_i, C)$ as defined in [3]. A set of Horn clauses N is *saturated* by $\mathcal{R}$ 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 *derivation* by $\mathcal{R}$ 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 $\mathcal{R}$ from premises in N_i. The limit N_∞ 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 $\square \notin N_\infty$. A clause C is said to be *derivable* from N iff $C \in N_\infty$.

3 Answering Conjunctive Queries in $\mathcal{ELHI}$

Given an $\mathcal{ELHI}$ TBox $\mathcal{T}$ and a conjunctive query $Q = \langle Q_P, \{Q_C\}\rangle$, our goal is to compute a query $\mathsf{rew}(Q, \mathcal{T})$ such that, for each ABox $\mathcal{A}$, evaluating $\mathsf{rew}(Q, \mathcal{T})$ over $\mathcal{A}$ and evaluating the query Q directly over $\mathcal{K} = \langle \mathcal{T}, \mathcal{A}\rangle$ 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 $\mathsf{ans}(Q, \mathcal{K})$ directly; then, in Section 4 we use this result to derive the rewriting algorithm.

It is well known that $\vec{a} \in \mathsf{ans}(Q, \mathcal{K})$ if and only if $\Xi(\mathcal{K}) \cup \{Q_C,\ \bot \leftarrow Q_P(\vec{a})\}$ is unsatisfiable, where $\Xi(\mathcal{K})$ is the transformation of $\mathcal{K}$ into a set of clauses. Therefore, to answer Q over $\mathcal{K}$, we need a decision procedure for checking satisfiability of sets of clauses. We derive this procedure using the principles outlined by Joyner [11]. Given $\mathcal{K}$ and Q, with $\mathcal{N}$ we denote the set of clauses of the forms shown in Table 3. Clearly, $\mathcal{N}$ is finite assuming that $\mathcal{K}$ and Q are finite. Moreover, if we translate $\mathcal{K}$ into $\Xi(\mathcal{K})$ according to Table 2, then $\Xi(\mathcal{K}) \cup \{Q_C\} \subseteq \mathcal{N}$. Finally, we saturate $\Xi(\mathcal{K}) \cup \{Q_C,\ \bot \leftarrow Q_P(\vec{a})\}$ using $\mathcal{R}^{DL}$—a suitably parameterized resolution with free selection calculus. Since $\mathcal{R}^{DL}$ 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 $\mathcal{N}$ is closed under $\mathcal{R}^{DL}$. We now formally define Ξ, $\mathcal{N}$, and $\mathcal{R}^{DL}$.

Table 2: Translating $\mathcal{ELHI}$ axioms into clauses

$\mathcal{ELHI}$ clause	$\mathcal{ELHI}$ axiom	$\mathcal{ELHI}$ clause	$\mathcal{ELHI}$ axiom
$A(a)$	$A(a)$	$P(f^i_A(x), x) \leftarrow A(x)$	$A \sqsubseteq \exists P^-.B$
$P(a, b)$	$P(a, b)$	$B(f^i_A(x)) \leftarrow A(x)$	
$B(x) \leftarrow A(x)$	$A \sqsubseteq B$	$B(x) \leftarrow P(x, y) \wedge A(y)$	$\exists P.A \sqsubseteq B$
$C(x) \leftarrow A(x) \wedge B(x)$	$A \sqcap B \sqsubseteq C$	$B(x) \leftarrow P(y, x) \wedge A(y)$	$\exists P^-.A \sqsubseteq B$
$P(x, f^i_A(x)) \leftarrow A(x)$	$A \sqsubseteq \exists P.B$	$S(x, y) \leftarrow P(x, y)$	$P \sqsubseteq S, P^- \sqsubseteq S^-$
$B(f^i_A(x)) \leftarrow A(x)$		$S(x, y) \leftarrow P(y, x)$	$P^- \sqsubseteq S, P \sqsubseteq S^-$

Note 1. Each axiom of the form $A \sqsubseteq \exists P^{(-)}.B$ is uniquely associated with a function symbol f^i_A.

Definition 1. *Let $\mathcal{K}$ be an $\mathcal{ELHI}$ knowledge base and $Q = \langle Q_P, \{Q_C\}\rangle$ a conjunctive query. The set of clauses $\Xi(\mathcal{K})$ is obtained by transforming $\mathcal{K}$ as shown in Table 2. The set of $\mathcal{ELHI}$ clauses $\mathcal{N}$ is the set of all clauses of types shown in Table 3 constructed using the symbols in Q_C and $\mathcal{K}$.*

With $\mathcal{R}^{DL}$ 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 A2,A3,T3–T8, then S selects the atoms that are underlined in Table 3; (b) if C is of type A1, T1, or T2, 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 Q1, 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_∞ denotes the limit of a fair derivation from a set of clauses N by $\mathcal{R}^{DL}$.

Table 3: Clause Set $\mathcal{N}$ for $Q = \langle Q_P, Q_C\rangle$ and $\mathcal{K}$

Type	$\mathcal{ELHI}$ clause	Type	$\mathcal{ELHI}$ clause
A1	$B(a) \leftarrow \mathbf{A}(a)$	T4	$B(x) \leftarrow \underline{P(y, x)} \wedge [A(y)]$
A2	$B(a) \leftarrow \underline{A(b)}$	T5	$S(x, y) \leftarrow \underline{P(x, y)}$
A3	$\underline{P(a, b)}$	T6	$S(x, y) \leftarrow \underline{P(y, x)}$
T1	$\underline{D(x)} \leftarrow \mathbf{A}(x) \wedge \mathbf{B}(f^i_C(x))$	T7	$\underline{P(x, f^i_A(x))} \leftarrow A(x)$
T2	$D(f^i_C(x)) \leftarrow \mathbf{A}(x) \wedge \mathbf{B}(f^i_C(x))$	T8	$\underline{P(f^i_A(x), x)} \leftarrow A(x)$
T3	$B(x) \leftarrow \underline{P(x, y)} \wedge [A(y)]$	Q1	$\underline{Q_P(\vec{u})} \leftarrow \mathbf{L}(\vec{t_i})$

Note 2. A, B, C, D are atomic concepts; P, S are atomic roles; L is an atomic concept or role; and $\vec{u}, \vec{t_i}$ are tuples of terms. With $\mathbf{C}(t)$ we denote $\bigwedge_{i=1}^n C_i(t)$, and with $\mathbf{C}(\vec{t_i})$, we denote $\bigwedge_{i=1}^n C_i(\vec{t_i})$, 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 Q1, if C contains a function symbol of the form $f^i_A(x)$, then C contains $A(x)$ in the body. For each clause C of type Q1, (i) $\mathsf{var}(C) \leq \mathsf{var}(Q_C)$, (ii) $\mathsf{depth}(C) \leq \max(1, \mathsf{var}(Q_C) - \mathsf{var}(C))$, and (iii) if a variable x occurs in a functional term in C, then x occurs in all functional terms in C.

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

Proof. It can be seen in Table 4 that if C_1 and C_2 are of type A1–T8, then C_r is also of type A1–T8. Assume that C_1 is of type Q1, satisfying properties (i)–(iii) of Table 3. If the head atom $Q_P(\vec{t})$ of C_1 is selected, then resolution is not possible, since no clause in $\mathcal{N}$ 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 A1 or T2; we now show that C_r satisfies properties (i)–(iii) of Table 3.

- If C_2 is of type A1, unification is possible only if the term t is either a constant a or a variable y. In the former case, the unifier σ is empty; in the latter case, $\sigma = \{y \mapsto a\}$. Clearly, $\mathsf{var}(C_r) \leq \mathsf{var}(C_1)$ and $\mathsf{depth}(C_r) = \mathsf{depth}(C_1)$, 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 T2, unification is possible only if the term t is a variable or a functional term.
  - If t is a variable y, then $\sigma = \{y \mapsto f^i_A(x)\}$. Clearly, $\mathsf{var}(C_r) = \mathsf{var}(C_1)$, so C_r satisfies (i). Furthermore, $\mathsf{depth}(C_1) = 0$ and $\mathsf{depth}(C_r) \leq 1$, so C_r satisfies (ii). Finally, every occurrence of y is replaced with $f^i_A(x)$, and C_1 does not contain functional terms, so (iii) holds as well.
  - If t is a functional term $f^i_A(s)$, the unifier is of the form $\sigma = \{x \mapsto s\}$. Clearly, $\mathsf{var}(C_r) = \mathsf{var}(C_1)$, so C_r satisfies (i). Furthermore, since no term in C_1 is deeper than $f^i_A(s)$, we have $\mathsf{depth}(C_r) \leq \mathsf{depth}(C_1)$, 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 A3, T7 or T8. We now show that C_r satisfies properties (i)–(iii) of Table 3.

- If C_2 is of type A3, the unification is possible only if the terms s and t are not functional terms. If they are both constants, the substitution σ is empty; otherwise, σ maps s, t, or both to the corresponding constants in C_2. Clearly, $\mathsf{var}(C_r) \leq \mathsf{var}(C_1)$, so C_r satisfies (i). Furthermore, $\mathsf{depth}(C_r) = \mathsf{depth}(C_1)$, so C_r satisfies (ii). Finally, since $P(s,t)$ is the deepest atom in in C_1, the clause C_1 does not contain functional terms, so C_r satisfies (iii) vacuously.
- If C_2 is of type T7 (analogous for T8), unification is possible only if the term t is a variable or a functional term.
  - If t is a variable x_t, then $\sigma = \{x_t \mapsto f^i_A(s), x \mapsto s\}$. Due to the occurs-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: $\mathsf{var}(C_r) = \mathsf{var}(C_1) - 1$, so C_r satisfies (i). Furthermore, C_1 satisfies (iii), so x_t does not occur in a functional term in C_1 (because it does not occur in s). Hence, even though x_t is mapped to a functional term, $\mathsf{depth}(C_r) \leq \mathsf{depth}(C_1) + 1$, so C_r satisfies (ii). Finally, since every occurrence of x_t is replaced with $f^i_A(s)$, C_r satisfies (iii) as well.
  - Assume that t is a functional term $f^i_A(t')$. If s does not occur in t', then s is a variable x_s and $\sigma = \{x \mapsto t', x_s \mapsto t'\}$. If s occurs in t', the only way for the inference to be possible is if $t' = s$, so $\sigma = \{x \mapsto t'\}$. In both cases, $\mathsf{var}(C_r) \leq \mathsf{var}(C_1)$ and $\mathsf{depth}(C_r) \leq \mathsf{depth}(C_1)$, so C_r satisfies

properties (i) and (ii). Furthermore, the inference does not introduce new functional terms, so C_r satisfies (iii) as well.

This covers all possible forms of C_1 and C_2, so the lemma holds. □

Clearly, Lemma 1 implies that the saturation of $\Xi(\mathcal{K}) \cup \{Q_C\}$ by $\mathcal{R}^{DL}$ terminates. The principles outlined before Definition 1, however, allow us only to check whether some tuple $\vec{a}$ is an answer to Q over $\mathcal{K}$. In order to compute the entire set $\mathsf{ans}(Q, \mathcal{K})$, we use the *answer literal* technique—that is, $\vec{a} \in \mathsf{ans}(Q, \mathcal{K})$ if and only if $Q_P(\vec{a}) \in (\Xi(\mathcal{K}) \cup \{Q_C\})_\infty$, for any tuple of constants $\vec{a}$ (c.f. [10]).

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 $\mathsf{rew}(Q, \mathcal{T})$ such that, for any ABox $\mathcal{A}$, the sets of answers of Q over $\langle \mathcal{T}, \mathcal{A} \rangle$ and of $\mathsf{rew}(Q, \mathcal{T})$ 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$_{core}$ and DL-Lite$_R$, then $\mathsf{rew}(Q, \mathcal{T})$ should be a union of conjunctive queries; 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 program; finally, if $\mathcal{T}$ is in a language between $\mathcal{EL}$ and $\mathcal{ELHI}$, then $\mathsf{rew}(Q, \mathcal{T})$ 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.

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, $\mathsf{ff}(Q, \mathcal{T})$ is the set that contains exactly all function-free clauses contained in $(\Xi(\mathcal{T}) \cup \{Q_C\})_\infty$.*

We now show that, for each ABox $\mathcal{A}$, we have $\mathcal{T} \cup \{Q_C\} \cup \mathcal{A} \models Q_P(\vec{a})$ if and only if $\mathsf{ff}(Q, \mathcal{T}) \cup \mathcal{A} \models Q_P(\vec{a})$, which makes $\mathsf{ff}(Q, \mathcal{T})$ a rewriting of Q w.r.t. $\mathcal{T}$, albeit not necessarily an optimal one. We prove the claim proof-theoretically: we show that $Q_P(\vec{a})$ is derivable from $\Xi(\mathcal{T}) \cup \{Q_C\} \cup \mathcal{A}$ if and only if it is derivable from $\mathsf{ff}(Q, \mathcal{T}) \cup \mathcal{A}$. To this end, we first need to show that we can always "postpone" the inferences with the ABox clauses in the saturation of $\Xi(\mathcal{T}) \cup \{Q_C\} \cup \mathcal{A}$—that is, we can first perform all inferences with nonground clauses only, and then perform the inferences involving a ground clause. We omit the proof for lack of space, it can be found in [18].

Lemma 2 ([18]). *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 C of type Q1 that is derivable from $\Xi(\mathcal{T}) \cup \{Q_C\} \cup \mathcal{A}$, a clause 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 $(\mathsf{ff}(Q, \mathcal{T}) \cup \mathcal{A})_\infty$, we have $\{C'\} \cup G \models C$.*

Table 4: Inferences of $\mathcal{R}^{DL}$ on $\mathcal{N}$

A1 + A1 = A1:
$$\frac{\underline{A(a)} \quad C(a) \leftarrow \underline{A(a)} \wedge \mathbf{B}(a)}{C(a) \leftarrow \mathbf{B}(a)}$$

A1 + A2 = A1:
$$\frac{\underline{A(a)} \quad B(b) \leftarrow \underline{A(a)}}{B(b)}$$

A1 + T1 = A1:
$$\frac{\underline{A(a)} \quad C(x) \leftarrow \underline{A(x)} \wedge \mathbf{B}(x)}{C(a) \leftarrow \mathbf{B}(\mathbf{a})}$$

A3 + T3 = A1/A2:
$$\frac{\underline{P(a,b)} \quad B(x) \leftarrow \underline{P(x,y)} \wedge [A(y)]}{B(a) \leftarrow [A(b)]}$$

A3 + T5 = A3:
$$\frac{\underline{P(a,b)} \quad S(x,y) \leftarrow \underline{P(x,y)}}{S(a,b)}$$

T1 + T2 = T2:
$$\frac{C(x) \leftarrow \underline{A(x)} \wedge \mathbf{B}(x) \quad \underline{A(f_D^i(x))} \leftarrow \mathbf{F}(x)}{C(f_D^i(x)) \leftarrow \mathbf{B}(f_D^i(x)) \wedge \mathbf{F}(x)}$$

T1 + T2 = T1:
$$\frac{E(x) \leftarrow \underline{B(f_A^i(x))} \wedge \mathbf{C}(f_A^i(x)) \wedge \mathbf{D}(x) \quad \underline{B(f_A^i(x))} \leftarrow \mathbf{G}(x)}{E(x) \leftarrow \mathbf{C}(f_A^i(x)) \wedge \mathbf{I}(x)}$$

T2 + T2 = T2:
$$\frac{E(f_A^i(x)) \leftarrow \underline{B(f_A^i(x))} \wedge \mathbf{C}(f_A^i(x)) \wedge \mathbf{D}(x) \quad \underline{B(f_A^i(x))} \leftarrow \mathbf{F}(x)}{E(f_A^i(x)) \leftarrow \mathbf{C}(f_A^i(x)) \wedge \mathbf{G}(x)}$$

T3 + T7 = T1:
$$\frac{B(x) \leftarrow \underline{P(x,y)} \wedge [C(y)] \quad \underline{P(x, f_A^i(x))} \leftarrow A(x)}{B(x) \leftarrow A(x) \wedge [C(f_A^i(x))]}$$

T3 + T8 = T2:
$$\frac{B(x) \leftarrow \underline{P(x,y)} \wedge [C(y)] \quad \underline{P(f_A^i(x), x)} \leftarrow A(x)}{B(f_A^i(x)) \leftarrow A(x) \wedge [C(x)]}$$

T5 + T7 = T7:
$$\frac{S(x,y) \leftarrow \underline{P(x,y)} \quad \underline{P(x, f_A^i(x))} \leftarrow A(x)}{S(x, f_A^i(x)) \leftarrow A(x)}$$

T5 + T8 = T8:
$$\frac{S(x,y) \leftarrow \underline{P(x,y)} \quad \underline{P(f_A^i(x), x)} \leftarrow A(x)}{S(f_A^i(x), x) \leftarrow A(x)}$$

Q1 + A1 = Q1:
$$\frac{Q_P(\vec{u}) \leftarrow \underline{A(t)} \wedge \mathbf{L}(\vec{t_i}) \quad \underline{A(a)}}{Q_P(\vec{u})\sigma \leftarrow \mathbf{L}(\vec{t_i})\sigma}$$

Q1 + A3 = Q1:
$$\frac{Q_P(\vec{u}) \leftarrow \underline{P(s,t)} \wedge \mathbf{L}(\vec{t_i}) \quad \underline{P(a,b)}}{Q_P(\vec{u})\sigma \leftarrow \mathbf{L}(\vec{t_i})\sigma}$$

Q1 + T2 = Q1:
$$\frac{Q_P(\vec{u}) \leftarrow \underline{C(t)} \wedge \mathbf{L}(\vec{t_i}) \quad \underline{C(f_A^i(x))} \leftarrow \mathbf{B}(x)}{Q_P(\vec{u})\sigma \leftarrow \mathbf{B}(x)\sigma \wedge \mathbf{L}(\vec{t_i})\sigma}$$

Q1 + T7 = Q1:
$$\frac{Q_P(\vec{u}) \leftarrow \underline{P(s,t)} \wedge \mathbf{L}(\vec{t_i}) \quad \underline{P(x, f_A^i(x))} \leftarrow A(x)}{Q_P(\vec{u})\sigma \leftarrow A(x)\sigma \wedge \mathbf{L}(\vec{t_i})\sigma}$$

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.

This lemma now allows us to prove the desired relationship between the answers of Q over $\mathcal{T}$ and $\mathcal{A}$, and the answers of $\mathsf{ff}(Q,\mathcal{T})$ over $\mathcal{A}$. Clearly, if we assume that $Q_P(\vec{a})$ is derivable from $\Xi(\mathcal{T}) \cup \{Q_C\} \cup \mathcal{A}$, then, since $Q_P(\vec{a})$ is of type Q1, by Lemma 2, a clause C' of type Q1 is derivable from $\Xi(\mathcal{T}) \cup \{Q_C\}$ such that $\{C'\} \cup G \models C$. Since $Q_P(\vec{a})$ does not contain function symbols, C' cannot contain function symbols either, so $C' \in \mathsf{ff}(Q,\mathcal{T})$. Thus, $Q_P(\vec{a})$ is implied by $\mathsf{ff}(Q,\mathcal{T}) \cup G$ so, by the definition of G, we have $\mathsf{ff}(Q,\mathcal{T}) \cup \mathcal{A} \models Q_P(\vec{a})$. Therefore, Lemma 3 easily follows.

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

Optimizing the Program through Unfolding. According to Lemma 3, the datalog program $\mathsf{ff}(Q,\mathcal{T})$ is a rewriting of Q w.r.t. $\mathcal{T}$. 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 $\mathcal{T}$ and its translation $\Xi(\mathcal{T})$ to clauses:

$\mathcal{T}$	$\Xi(\mathcal{T})$	
$\exists$ hasParent.Human $\sqsubseteq$ Human	$\text{Human}(x) \leftarrow \underline{\text{hasParent}(x,y)} \wedge \underline{\text{Human}(y)}$	(1)
hasMother $\sqsubseteq$ hasParent	$\text{hasParent}(x,y) \leftarrow \underline{\text{hasMother}(x,y)}$	(2)

Given the query $Q = \langle Q_P, \{Q_P(x) \leftarrow \text{Human}(x)\}\rangle$, one can easily verify that $\mathsf{ff}(Q,\mathcal{T}) = \Xi(\mathcal{T}) \cup \{Q_P(x) \leftarrow \text{Human}(x)\}$. This is so because such a set does not contain functional symbols and it is already saturated by $\mathcal{R}^{DL}$. 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 $\Xi(\mathcal{T})$; therefore, (1) is not linear, which means that $\mathsf{ff}(Q,\mathcal{T})$ is not an optimal rewriting of Q w.r.t. $\mathcal{T}$.

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

Definition 3. *The* unfolding *of $L(\vec{x}) \leftarrow \mathbf{M}(\vec{m})$ in $N(\vec{n}) \leftarrow L(\vec{x'}) \wedge \mathbf{P}(\vec{p})$ is the clause $N(\vec{n})\sigma \leftarrow \mathbf{M}(\vec{m})\sigma \wedge \mathbf{P}(\vec{p})\sigma$, where $\sigma = \mathsf{MGU}(L(\vec{x}), L(\vec{x'}))$. 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 $\mathsf{unfold}(R,U) = R_U \setminus U$.*

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

The rewriting $\mathsf{rew}(Q,\mathcal{T})$ *of a conjunctive query $Q = \langle Q_P, \{Q_C\}\rangle$ w.r.t. an $\mathcal{ELHI}$ TBox $\mathcal{T}$ is the query $\langle Q_P, \mathsf{unfold}(R,U)\rangle$, where $R = \mathsf{ff}(Q,\mathcal{T})$ and U is the subset of $\mathcal{N}$ of all clauses of types T1*, T3*, T4*, T5, and 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 $R \cup A \models F(\vec{a})$ if and only if $\mathsf{unfold}(R,U) \cup A \models F(\vec{a})$. Theorem 1 follows from this fact, given Lemma 3, and that we can assume that Q_P does not occur in $\Xi(\mathcal{T})$.

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

We now prove important properties about the structure of the rewriting. First, we note that since $\mathsf{ff}(Q, \mathcal{T})$ 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 $\mathcal{T}$ an $\mathcal{ELHI}$ TBox,* $\mathsf{rew}(Q, \mathcal{T})$ *is a datalog query.*

Lemma 5. *For $Q = \langle Q_P, \{Q_C\} \rangle$ a conjunctive query and $\mathcal{T}$ a DL-Lite$^+$ TBox, and* $\mathsf{rew}(Q, \mathcal{T}) = \langle Q_P, P \rangle$*, 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 $\Xi(\mathcal{T})$ 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 $\mathcal{R}^{DL}$ produces only clauses of types T1, T2*, T3, T5, and T7. Therefore, $\mathsf{ff}(Q, \mathcal{T})$ 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 $\mathsf{ff}(Q, \mathcal{T})$, 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 $B(x) \leftarrow P(x, y) \wedge S(y, z)$ 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 $\mathcal{T}$ a DL-Lite$_R$ TBox,* $\mathsf{rew}(Q, \mathcal{T})$ *is a union of conjunctive queries.*

Proof. It follows from Table 2 that $\Xi(\mathcal{T})$ 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 $\mathcal{R}^{DL}$ produces only clauses of types T1*, T2*, T3*, T4*, T5, T6, T7 and T8. Therefore, $\mathsf{ff}(Q, \mathcal{T})$ 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 $\mathsf{ff}(Q, \mathcal{T})$ that are not of type Q1 are unfolded in clauses of type Q1, which immediately means that $\mathsf{rew}(Q, \mathcal{T})$ is a union of conjunctive queries. □

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 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 $P \cup A \models Q_P(\vec{a})$ for a datalog program P, a set of facts A, and a tuple of constants $\vec{a}$, 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 $\mathsf{rew}(Q, \mathcal{T})$ does not depend on the size of $\mathcal{A}$.

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

We now discuss the optimality of our rewriting. First, by Lemma 4, if $\mathcal{T}$ is an $\mathcal{ELH}$ TBox, then $\mathsf{rew}(Q, \mathcal{T})$ is a datalog query; therefore, we can evaluate $\mathsf{rew}(Q, \mathcal{T})$ in PTime [8], just as in [19]. Similarly, by Lemma 5, 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 program; therefore, we can evaluate $\mathsf{rew}(Q, \mathcal{T})$ in NLogSpace [17], just as in [17]. Finally, by Lemma 6, if $\mathcal{T}$ is a DL-Lite$_R$ TBox, then $\mathsf{rew}(Q, \mathcal{T})$ is a union of conjunctive queries; therefore, we can evaluate $\mathsf{rew}(Q, \mathcal{T})$ in LogSpace [12], just as in [7]. Summing up, our algorithm deals with the full spectrum of languages from DL-Lite$_R$ 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 [19], 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. F. Baader, S. Brandt, and C. Lutz. Pushing the EL Envelope. In *International Joint Conferences on Artificial Intelligence (IJCAI-05)*, 2005.
2. F. Baader and W. Nutt. *Basic Description Logics*, chapter 2, pages 47–100. Cambridge University Press, 2003.
3. F. Baader and W. Snyder. Unification Theory. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume I, chapter 8, pages 445–532. Elsevier Science, 2001.

[1] `http://www.comparagrid.org`

4. L. Bachmair and H. Ganzinger. Resolution Theorem Proving. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume 1, chapter 2, pages 19–100. North Holland, 2001.
5. A. Calì. Query Answering by Rewriting in GLAV Data Integration Systems Under Constraints. In C. Bussler, V. Tannen, and I. Fundulaki, editors, *SWDB*, volume 3372, pages 167–184, 2004.
6. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Data complexity of query answering in description logics. In *Proc. of the 10th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2006)*, pages 260–270, 2006.
7. 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. *J. of Automated Reasoning*, 2007.
8. 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.
9. R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data Exchange: Semantics and Query Answering. In *ICDT*, pages 207–224, 2003.
10. 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.
11. W. H. Joyner. Resolution strategies as decision procedures. *J. ACM*, 23(3):398–417, 1976.
12. P. C. Kanellakis, G. M. Kuper, and P. Z. Revesz. Constraint query languages. In *Symposium on Principles of Database Systems*, pages 299–313, 1990.
13. Y. Kazakov. *Saturation-Based Decision Procedures for Extensions of the Guarded Fragment*. PhD thesis, Universität des Saarlandes, March 2006.
14. M. Lenzerini. Data Integration: a theoretical perspective. In *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.
15. B. Motik. *Reasoning in Description Logics using Resolution and Deductive Databases*. PhD thesis, Univesität Karlsruhe (TH), January 2006.
16. P. F. Patel-Schneider, P. Hayes, and I. Horrocks. OWL Web Ontology Language Semantics and Abstract Syntax. W3C Recommendation, Frebruary 2004.
17. 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. To appear.
18. H. Pérez-Urbina, B. Motik, and I. Horrocks. Rewriting Conjunctive Queries under Description Logic Constraints. Technical report, University of Oxford, 2008.
19. R. Rosati. On conjunctive query answering in EL. In *Proceedings of the 2007 International Workshop on Description Logics (DL2007)*, CEUR-WS, 2007.
20. 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.
21. J. Widom. Research Problems in Data Warehousing. In *4th International Conference on Information and Knowledge Management*, pages 25–30, Baltimore, Maryland, 1995.