Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015)

Efficient Query Rewriting in the Description Logic $\mathcal{EL}$ and Beyond

Peter Hansen and **Carsten Lutz** and **İnanç Seylan**
University of Bremen, Germany
{hansen, clu, seylan}@informatik.uni-bremen.de

Frank Wolter
University of Liverpool, UK
frank@csc.liv.ac.uk

Abstract

We propose a new type of algorithm for computing first-order (FO) rewritings of concept queries under $\mathcal{ELH}^{dr}$-TBoxes. The algorithm is tailored towards efficient implementation, yet complete. It outputs a succinct non-recursive datalog rewriting if the input is FO-rewritable and otherwise reports non-FO-rewritability. We carry out experiments with real-world ontologies which demonstrate excellent performance in practice and show that TBoxes originating from applications admit FO-rewritings of reasonable size in almost all cases, even when in theory such rewritings are not guaranteed to exist.

1 Introduction

Ontology-based data access (OBDA) with Description Logics (DLs) is a very active topic of research, which has resulted in various approaches to implementing OBDA in practice [Poggi *et al.*, 2008; Lutz *et al.*, 2009; Pérez-Urbina *et al.*, 2010; Eiter *et al.*, 2012]. A particularly promising such approach is query rewriting, which enables implementations based on conventional relational database systems (RDBMSs), thus taking advantage of those systems' efficiency and maturity [Poggi *et al.*, 2008; Kontchakov *et al.*, 2013]. The general idea is to transform the original query q and the relevant TBox $\mathcal{T}$ into a first-order (FO) query $q_{\mathcal{T}}$ that is then handed over to the RDBMS for execution. One limitation of this approach is that, for the majority of description logics that are used as ontology languages, the query $q_{\mathcal{T}}$ is not guaranteed to exist. In fact, this is the case already for the members of the popular $\mathcal{EL}$ family of lightweight DLs [Baader *et al.*, 2005; Lutz *et al.*, 2009], which underly the EL profile of the OWL2 ontology language and are frequently used in health care and biology ontologies. This observation, however, does not rule out the possibility that FO-rewritings still exist in many practically relevant cases. In fact, TBoxes that emerge from practical applications tend to have a rather simple structure and one might thus speculate that, indeed, FO-rewritings under $\mathcal{EL}$-TBoxes tend to exist in practice.

In this paper, we consider the construction of FO-rewritings of concept queries under TBoxes that are formulated in the description logic $\mathcal{ELH}^{dr}$. The latter is a logical core of OWL2 EL that extends basic $\mathcal{EL}$ with role inclusions and domain/range restrictions on roles [Lutz *et al.*, 2009] while concept queries take the form $C(x)$ with C an $\mathcal{EL}$-concept. Constructing the desired rewritings is computationally hard: it follows from results in [Bienvenu *et al.*, 2013] that deciding whether a given concept query $C(x)$ is FO-rewritable under a given TBox $\mathcal{T}$ is PSPACE-complete both in $\mathcal{EL}$ and in $\mathcal{ELH}^{dr}$, and it even becomes EXPTIME-complete when the vocabulary of the admitted database instances (i. e., ABoxes) is an additional input—a feature that we allow in the current paper.

Existing approaches to query rewriting under DL TBoxes can be summarized as follows: (i) approaches that target rewritings into the more expressive query language datalog and which are incomplete in the sense that existing datalog-rewritings are not guaranteed to be found and, moreover, the generated datalog-rewritings are not necessarily non-recursive even if there is an FO-rewriting [Rosati, 2007; Pérez-Urbina *et al.*, 2010; Kaminski and Grau, 2013; Mora and Corcho, 2013]; (ii) backwards chaining approaches for existential rules (a strict generalization of $\mathcal{ELH}^{dr}$) which are complete in the sense that they find an FO-rewriting if there is one, but need not terminate otherwise [König *et al.*, 2012]; (iii) complete and terminating approaches which aim to prove upper complexity bounds, but which are not practically feasible [Bienvenu *et al.*, 2013; 2014].

The aim of this paper is *to design, for the first time, algorithms for computing FO-rewritings of concept queries under $\mathcal{EL}$- and $\mathcal{ELH}^{dr}$-TBoxes that are complete, terminating, and feasible in practice.* We start with a marriage of approaches (ii) and (iii) above to get the best of both worlds; in particular, (ii) appears to be practically much more feasible than (iii) while (iii) provides a way to achieve termination. The resulting backwards chaining algorithm is conceptually simple and constitutes a significant step towards our goal. However, it produces FO-rewritings that are unions of conjunctive queries (UCQs), which has two significant drawbacks: first, recent experiments [Lutz *et al.*, 2013] have shown that executing UCQ-rewritings on RDBMSs is prohibitively expensive while executing equivalent rewritings that take the form of non-recursive datalog programs is much more feasible; and second, UCQ-rewritings can be of excessive size even in practically relevant cases [Rosati and Almatelli, 2010].

To address these shortcomings, the main contribution of

this paper is to refine our initial backwards chaining algorithm to what we call a *decomposed algorithm*. While the initial algorithm uses tree-shaped conjunctive queries (CQs) as an internal data structure, the new algorithm only represents single nodes of tree-shaped CQs together with information of how to reassemble these nodes into full queries. This can be seen as a way to implement structure sharing and allows us to directly produce rewritings that are non-recursive datalog programs, avoiding UCQ-rewritings altogether. The algorithm runs in exponential time, is capable of deciding the existence of FO-rewritings in ExpTime (which is optimal), and is capable of producing monadic non-recursive datalog rewritings that are of at most exponential size (but much smaller in practice). We also show that an exponential blowup is unavoidable when rewriting into monadic non-recursive datalog. Technically, the decomposed algorithm is much more subtle than the initial one.

We then evaluate the decomposed algorithm by carrying out experiments with seven ontologies from practical applications. We ask for an FO-rewriting of every concept query $A(x)$ with A a concept name from the ontology under consideration. Out of 10989 inputs in total and with a timeout of 30 seconds, the decomposed algorithm terminates on all but 127 inputs and on average needs less than 0.5 seconds per input. We also analyse the size of the generated non-recursive datalog rewritings, with extremely encouraging results. Our experiments show that the decomposed algorithm performs very well on ontologies from practical applications and also confirm our initial belief that such ontologies very often admit FO-rewritings. In particular, only 74 of 10989 inputs turn out to be not FO-rewritable.

Proof details are deferred to the appendix, available at http://www.informatik.uni-bremen.de/tdki/p.html.

2 Preliminaries

We use $\mathsf{N_C}$ and $\mathsf{N_R}$ to denote mutually disjoint countably infinite sets of concept and role names, respectively. An *$\mathcal{EL}$-concept* is formed according to the syntax rule $C ::= A \mid \top \mid C \sqcap C \mid \exists r.C$, where $A \in \mathsf{N_C}$ and $r \in \mathsf{N_R}$. Let C, D be $\mathcal{EL}$-concepts and r, s be role names. Then $C \sqsubseteq D$ is a *concept inclusion (CI)*, $r \sqsubseteq s$ is a *role inclusion (RI)*, $\mathsf{dom}(r) \sqsubseteq C$ is a *domain restriction*, and $\mathsf{ran}(r) \sqsubseteq C$ is a *range restriction*. An *$\mathcal{ELH}$-TBox* is a finite set of CIs and RIs and an $\mathcal{ELH}^{dr}$-TBox additionally admits domain and range restrictions. The semantics of concepts and TBoxes is defined as usual [Baader *et al.*, 2007; 2005]. For a CI or RI α, we write $\mathcal{T} \models \alpha$ if every model of $\mathcal{T}$ satisfies α; when $\mathcal{T}$ is empty, we write $\models \alpha$.

An ABox $\mathcal{A}$ is a set of *assertions* of the form $A(a)$ and $r(a, b)$ with A a concept name, r a role name, and a, b individual names from a countably infinite set $\mathsf{N_I}$. We use $\mathsf{ind}(\mathcal{A})$ to denote the set of individual names that occur in $\mathcal{A}$. A *concept query* is an expression $C(x)$ with C an $\mathcal{EL}$-concept. We write $\mathcal{A}, \mathcal{T} \models C(a)$ and say that *a is a certain answer to C given the ABox $\mathcal{A}$ and TBox $\mathcal{T}$* if $a \in \mathsf{ind}(\mathcal{A})$ and $a^{\mathcal{I}} \in C^{\mathcal{I}}$ for all models $\mathcal{I}$ of $\mathcal{T}$ and $\mathcal{A}$. A *signature* is a finite set $\Sigma \subseteq \mathsf{N_C} \cup \mathsf{N_R}$. An ABox is a *$\Sigma$-ABox* if it only uses concept and role names from Σ.

For an FO-formula $q(x)$ with one free variable x, we write $\mathcal{A} \models q(a)$ if $\mathcal{A}$ (viewed as an interpretation) satisfies q under the assignment that maps x to a. A concept query $C(x)$ is *FO-rewritable under a TBox $\mathcal{T}$ and signature Σ* if there is an FO-formula $\varphi(x)$ such that for all Σ-ABoxes $\mathcal{A}$ and individuals a, we have $\mathcal{A}, \mathcal{T} \models C(a)$ iff $\mathcal{A} \models \varphi(a)$. In this case, we call $\varphi(x)$ an *FO-rewriting* of $C(x)$ under $\mathcal{T}$ and Σ. When studying FO-rewritability of concept queries $C(x)$, we can assume w. l. o. g. that C is a concept name since $C(x)$ is FO-rewritable under a TBox $\mathcal{T}$ and Σ iff $A(x)$ is FO-rewritable under $\mathcal{T} \cup \{C \sqsubseteq A\}$ and Σ, where A is a fresh concept name [Bienvenu *et al.*, 2013].

We will also consider other forms of rewritings, in particular into *unions of conjunctive queries (UCQs)* and *non-recursive datalog programs*. Although, strictly speaking, non-recursive datalog rewritings are not FO-rewritings, the existence of either kind of rewriting coincides with the existence of an FO-rewriting [Bienvenu *et al.*, 2014]. In particular, non-recursive datalog rewritings can be viewed as a compact representation of a UCQ-rewriting. We will also consider monadic datalog as a special case, where all intensional (IDB) predicates are unary.

For any of these rewritings, if Σ is the set of all concept and role names in $\mathcal{T}$ and C, then $C(x)$ is rewritable under $\mathcal{T}$ and Σ iff it is rewritable under $\mathcal{T}$ and any signature. If this is the case, we say that $C(x)$ is rewritable under $\mathcal{T}$ and the *full signature*.

For our technical constructions, it will be convenient to view $\mathcal{EL}$-concepts as conjunctive queries (CQs) that take the form of a directed tree. We will represent such queries as sets of atoms of the form $A(x)$ and $r(x, y)$ with A a concept name, r a role name and x, y variables. Tree-shapedness of a conjunctive query q then means that the directed graph $(V, \{(x, y) \mid r(x, y) \in q\})$ is a tree, where V is the set of variables in q, and that $r(x, y), s(x, y) \in q$ implies $r = s$. We will not distinguish explicitly between an $\mathcal{EL}$-concept C and its query representation. We thus use $\mathsf{var}(C)$ to denote the set of variables that occur in C and x_ε to denote the root variable in C. For an $x \in \mathsf{var}(C)$, we use $C|_x$ to denote the $\mathcal{EL}$-concept represented by (the subtree rooted at) x. When we speak of a *top-level conjunct (tlc)* of an $\mathcal{EL}$-concept C, we mean a concept name A such that $A(x_\varepsilon) \in C$ or a concept $\exists r.D$ such that $r(x_\varepsilon, y) \in C$ and $D = C|_y$. We use $\mathsf{tlc}(C)$ to denote the set of all top-level conjuncts of C. For any syntactic object (such as a concept or a TBox), we define its *size* to be the number of symbols used to write it.

We show that we can remove domain and range restrictions from $\mathcal{ELH}^{dr}$-TBoxes and work with $\mathcal{ELH}$-TBoxes when developing algorithms that decide rewritability or compute rewritings.

Lemma 1. *For every $\mathcal{ELH}^{dr}$-TBox $\mathcal{T}$, signature Σ, and concept query $A_0(x)$, one can construct in polynomial time an $\mathcal{ELH}$-TBox $\mathcal{T}'$ and signature Σ' such that $A_0(x)$ is FO-rewritable under $\mathcal{T}$ and Σ iff it is FO-rewritable under $\mathcal{T}'$ and Σ'.*

Moreover, every UCQ and non-recursive datalog rewriting of $A_0(x)$ under $\mathcal{T}'$ and Σ' can be converted in linear time into a UCQ and, respectively, non-recursive datalog rewriting of A_0 under $\mathcal{T}$ and Σ.

3 A Backwards Chaining Algorithm

The algorithm presented in this section constructs a set of UCQ-rewritings of $A_0(x)$ under an $\mathcal{ELH}$-TBox $\mathcal{T}$ by starting from $\{A_0\}$ and then exhaustively applying the axioms in $\mathcal{T}$ as rules in a backwards chaining manner. It terminates by either constructing a UCQ-rewriting or returning 'not FO-rewritable'. For simplicity and since the purpose of the algorithm presented here is mainly to prepare for the subsequent, more refined one, we consider rewritability for the full signature only.

Let A_0 and $\mathcal{T}$ be an input to the algorithm. We start with introducing the central backwards chaining steps. Let C and D be $\mathcal{EL}$-concepts and let α be a (concept or role) inclusion from $\mathcal{T}$. The notion of D *being obtained from* C *by applying* α is defined as follows.

(CI) Let $\alpha = E \sqsubseteq F$ be a CI, $x \in \mathsf{var}(C)$, and let there be at least one tlc G of $C|_x$ with $\models F \sqsubseteq G$. Then D is *obtained from* C *by applying* α *at* x if D can be obtained from C by

- removing $A(x)$ for all concept names A with $\models F \sqsubseteq A$;
- removing the subtree rooted at y whenever $r(x, y) \in C$ and $\models F \sqsubseteq \exists r.(C|_y)$ (including the edge $r(x, y)$);
- adding $A(x)$ for all concept names A that are a tlc of E;
- adding the subtree $\exists r.H$ to x for each $\exists r.H$ that is a tlc of E.

(RI) Let $\alpha = r \sqsubseteq s$ be an RI and let $s(x, y) \in C$. Then D is *obtained from* C *by applying* α *at* $s(x, y)$ if D can be obtained from C by replacing $s(x, y)$ by $r(x, y)$.

The following is immediate.

Lemma 2. *If $\mathcal{T} \models C \sqsubseteq A_0$ and D can be obtained from C by applying some inclusion in $\mathcal{T}$, then $\mathcal{T} \models D \sqsubseteq A_0$.*

Apart from applying backwards chaining, our algorithm also minimises the generated concepts to attain completeness and termination. To make this precise, we introduce some notation. Let C and D be $\mathcal{EL}$-concepts and $x \in \mathsf{var}(C)$. Then $C \setminus C|_x$ denotes the concept obtained by removing from C the subtree rooted at x, and we write $C \prec D$ if there exists $x \in \mathsf{var}(D)$ such that $C = D \setminus D|_x$. We use $\prec^*$ to denote the reflexive and transitive closure of $\prec$ and say that C *is* $\prec$*-minimal with* $\mathcal{T} \models C \sqsubseteq A_0$ if $\mathcal{T} \models C \sqsubseteq A_0$ and there is no $C' \prec C$ with $\mathcal{T} \models C' \sqsubseteq A_0$. Note that if $\mathcal{T} \models C \sqsubseteq A_0$, then it is possible to find in polynomial time a $C' \prec^* C$ that is $\prec$-minimal with $\mathcal{T} \models C' \sqsubseteq A_0$.

The constructions in [Bienvenu *et al.*, 2013] suggest that, to achieve termination, we can use a certain form of blocking, similar to the blocking used in DL tableau algorithms. Let $\mathsf{sub}(\mathcal{T})$ denote the set of subconcepts of (concepts that occur in) $\mathcal{T}$. For each $\mathcal{EL}$-concept C and $x \in \mathsf{var}(C)$, we set $\mathsf{con}^C_{\mathcal{T}}(x) := \{D \in \mathsf{sub}(\mathcal{T}) \mid \mathcal{T} \models C|_x \sqsubseteq D\}$. We say that C is *blocked* if there are $x_1, x_2, x_3 \in \mathsf{var}(C)$ such that

1. x_1 is a proper ancestor of x_2 is an ancestor of x_3 and
2. $\mathsf{con}^D_{\mathcal{T}}(x_1) = \mathsf{con}^D_{\mathcal{T}}(x_2)$ for $D \in \{C, C \setminus C|_{x_3}\}$.

The algorithm is formulated in Figure 1. Note that, by Lemma 2, the concept D considered in the condition of the while loop satisfies $\mathcal{T} \models D \sqsubseteq A_0$ and thus we are guaranteed to find the desired D'. Also note that each of the potentially many candidates for D' will work.

procedure find-rewriting($A_0(x), \mathcal{T}$)
  $M := \{A_0\}$
  while there is a $C \in M$ and a concept D such that
    1. D can be obtained from C by applying some axiom in $\mathcal{T}$ and
    2. there is no $D' \prec D$ with $D' \in M$ then
    find a $D' \prec^* D$ that is $\prec$-minimal with $\mathcal{T} \models D' \sqsubseteq A_0$
    if D' is blocked then return 'not FO-rewritable'
    else add D' to M
  return the UCQ $\bigvee M$.

Figure 1: The backwards chaining algorithm

Example 3. *Let $\mathcal{T} = \{\ \exists r.(B_1 \sqcap B_2) \sqsubseteq A_0, \exists s.B_2 \sqsubseteq B_2\ \}$. Starting with $M = \{A_0\}$ and applying the first CI to A_0, we get $M = \{A_0, \exists r.(B_1 \sqcap B_2)\}$. The second CI can be applied repeatedly, starting with $\exists r.(B_1 \sqcap B_2)$, and yields concepts*

$$D_1 = \exists r.(B_1 \sqcap \exists s.B_2),$$
$$D_2 = \exists r.(B_1 \sqcap \exists s.\exists s.B_2),$$
$$D_3 = \exists r.(B_1 \sqcap \exists s.\exists s.\exists s.B_2),\ \ldots,$$

all of which are $\prec$-minimal with $\mathcal{T} \models D_i \sqsubseteq A_0$. D_3 is blocked and the algorithm will classify A_0 as 'not FO-rewritable' under $\mathcal{T}$.

Now consider the TBox $\mathcal{T}' = \mathcal{T} \cup \{B_1 \sqsubseteq B_2\}$. Already the concept D_1 is not $\prec$-minimal with $\mathcal{T}' \models D_1 \sqsubseteq A_0$ and instead of D_1, $\exists r.B_1$ is added to M. At this point, rule application stops and the UCQ $\bigvee M$ is returned as an FO-rewriting of A_0 under $\mathcal{T}'$.

$\mathcal{T}'$ illustrates that the algorithm is incomplete without the minimization step since this step prevents generation of the blocked concept D_3.

We now establish correctness and termination.

Theorem 4. *The algorithm in Figure 1 returns a UCQ-rewriting $\bigvee M$ if A_0 is FO-rewritable under $\mathcal{T}$ and the full signature and 'not FO-rewritable' otherwise.*

The generated UCQ-rewritings are not necessarily of minimal size. It is possible to attain minimal-size rewritings by using a stronger form of minimality when constructing the concept D', namely by redefining the relation "$\prec$" so that $C \preceq D$ if there is a root-preserving homomorphism from C to D (c.f. *most-general rewritings* in [König *et al.*, 2012]). As a consequence, the $\prec$-minimal concept D' with $\mathcal{T} \models D' \sqsubseteq A$ can then be of size exponential in the size of D. However, D' can still be constructed in output-polynomial time.

We prove in the appendix that all concepts in M have outdegree at most n and depth at most 2^{2n}, n the size of $\mathcal{T}$. As remarked in [Bienvenu *et al.*, 2013], the size of UCQ-rewritings can be triple exponential in the size of $\mathcal{T}$, and thus the same is true for the runtime of the presented algorithm. While this worst case is probably not encountered in practice, the size of M can become prohibitively large for realistic inputs. For this reason, we propose an improved algorithm in the subsequent section, which produces non-recursive datalog rewritings instead of UCQ-rewritings and whose runtime is at most single exponential.

4 A Decomposed Algorithm

The algorithm presented in this section consists of three phases. In Phase 1, a set Γ is computed that can be viewed as a decomposed representation of the set M from Section 3 in the sense that we store only single nodes of the tree-shaped concepts in M, rather than entire concepts. In many cases, we can already construct a non-recursive datalog rewriting after Phase 1. If this is not possible, we compute in Phase 2 a set Ω that enriches the node representation provided by Γ with sets of logical consequences that are relevant for Point 2 of the definition of *blocked* concepts. In Phase 3, we first execute a certain cycle check on Ω, which corresponds to checking the existence of a blocked concept in M. If no cycle is found, we then construct a non-recursive datalog rewriting.

Phase 1. Assume that $\mathcal{T}$ is a TBox, Σ an ABox-signature, and A_0 a concept name for which we want to compute an FO-rewriting under $\mathcal{T}$ and Σ. To present the algorithm, it is convenient to decompose conjunctions on the right-hand side of CIs, that is, to assume that $\mathcal{T}$ consists, apart from RIs, only of CIs of the form $C \sqsubseteq A$ with A a concept name and $C \sqsubseteq \exists r.D$. We start with describing the construction of a set Γ, whose elements we call node pairs. A *node pair* has the form (C, S), where $C \in \mathsf{sub}(\mathcal{T})$, and $S \subseteq \mathsf{sub}(\mathcal{T})$ is a set of concept names and concepts of the form $\exists r.C$. Intuitively, a node pair (C, S) describes a set of concepts D such that $\mathcal{T} \models D \sqsubseteq C$ and the following conditions are satisfied:

(i) the concept names that are tlcs of D are $S \cap \mathsf{N_C}$;

(ii) the existential restrictions that are the tlcs of D are obtained from the existential restrictions in S by replacing each $\exists r.E \in S$ with some $\exists s.E'$ such that $\mathcal{T} \models \exists s.E' \sqsubseteq \exists r.E$.

The computation of Γ starts with $\{(A_0, \{A_0\})\}$ and proceeds by exhaustively applying the following two rules:

(r1) if $(C, S) \in \Gamma$, $D \sqsubseteq A \in \mathcal{T}$ and $A \in S$, then extend Γ with $(C, (S \setminus \{A\}) \cup \mathsf{tlc}(D))$.

(r2) if $(C, S) \in \Gamma$, $D \sqsubseteq \exists r.F \in \mathcal{T}$, and there is an $\exists s.G \in S$ with $\mathcal{T} \models F \sqsubseteq G$ and $\mathcal{T} \models r \sqsubseteq s$, then extend Γ with $(C, (S \setminus \{\exists s.G \mid \mathcal{T} \models F \sqsubseteq G \text{ and } \mathcal{T} \models r \sqsubseteq s\}) \cup \mathsf{tlc}(D))$.

After applying either rule, we also have to add the pair $(G, \mathsf{tlc}(G))$ for every subconcept $\exists r.G$ of D to trigger further derivation.

Example 5. *Let $\mathcal{T} = \{\exists r.(B_1 \sqcap B_2) \sqsubseteq A_0,\ \exists s.B_2 \sqsubseteq B_2\}$, and Σ the full signature. Starting with the pair $(A_0, \{A_0\})$ and applying* (r1)*, we extend Γ by $(A_0, \{\exists r.(B_1 \sqcap B_2)\})$ and $((B_1 \sqcap B_2), \{B_1, B_2\})$.* (r1) *is now applicable to the latter, yielding $((B_1 \sqcap B_2), \{B_1, \exists s.B_2\})$ and $(B_2, \{B_2\})$. Another application of* (r1) *results in e. g. $(B_2, \{\exists s.B_2\})$ being added.*

From Γ we can extract a (potentially infinitary) UCQ-rewriting of A_0 under $\mathcal{T}$ and Σ, as follows. Start with defining Γ_Σ to be the set of all $(C, S) \in \Gamma$ such that $S \cap \mathsf{N_C} \subseteq \Sigma$. Then $\widehat{\Gamma}_\Sigma$ is obtained as the limit of the sequence of sets $\widehat{\Gamma}^0_\Sigma, \widehat{\Gamma}^1_\Sigma, \ldots$ defined as follows:

- $\widehat{\Gamma}^0_\Sigma := \{(C, \bigsqcap S) \mid (C, S) \in \Gamma_\Sigma \text{ and } S \subseteq \mathsf{N_C}\}$.
- $\widehat{\Gamma}^{i+1}_\Sigma$ is $\widehat{\Gamma}^i_\Sigma$ extended with all pairs (C, D) such that there is $(C, S) \in \Gamma_\Sigma$ with the following property: for each $\exists r.G \in S$ there are $(G, C_{r,G}) \in \widehat{\Gamma}^i_\Sigma$ and $s_{r,G} \in \Sigma$ such that $\mathcal{T} \models s_{r,G} \sqsubseteq r$ and

$$D = \bigsqcap_{A \in S \cap \mathsf{N_C}} A \ \sqcap \bigsqcap_{\exists r.G \in S} \exists s_{r,G}.C_{r,G}.$$

Note that if $(C, D) \in \widehat{\Gamma}_\Sigma$, then D uses only symbols from Σ. The set $\widehat{\Gamma}_\Sigma$ represents a UCQ-rewriting as follows.

Proposition 6 (Soundness and Completeness of $\widehat{\Gamma}_\Sigma$). *For all Σ-ABoxes $\mathcal{A}$ and $a \in \mathsf{ind}(\mathcal{A})$, we have $\mathcal{A}, \mathcal{T} \models A_0(a)$ iff there is an $(A_0, D) \in \widehat{\Gamma}_\Sigma$ with $\mathcal{A} \models D(a)$.*

Γ_Σ provides us with a sufficient condition for FO-rewritability of A_0 under $\mathcal{T}$ and Σ and suggests a way to (sometimes) produce a non-recursive datalog rewriting. In fact, if Γ_Σ is *acyclic* in the sense that the directed graph

$$G = (\Gamma_\Sigma, \{((C, S), (C', S')) \mid S \text{ contains } \exists r.C' \text{ with } \mathcal{T} \models s \sqsubseteq r \text{ for some } s \in \Sigma\})$$

contains no cycle, then $\widehat{\Gamma}_\Sigma$ is finite and we obtain a non-recursive datalog program Π_{Γ_Σ} that is a rewriting of A_0 under $\mathcal{T}$ and Σ by taking the rule

$$P_C(x) \leftarrow \bigwedge_{A \in S} A(x) \ \wedge \bigwedge_{\exists r.D \in S} \ \bigvee_{\mathcal{T} \models s \sqsubseteq r, s \in \Sigma} \big(s(x, y_{r,D}) \wedge P_D(y_{r,D}) \big)$$

for each $(C, S) \in \Gamma_\Sigma$ and using P_{A_0} as the goal predicate. Note that the disjunctions can be removed by introducing auxiliary (monadic) IDB predicates, without causing a significant blowup. If Γ_Σ is acyclic, we output the above rewriting. Note that it is potentially much smaller than a UCQ-rewriting since it implements structure sharing. However, if Γ_Σ is not acyclic, then A_0 could still be FO-rewritable under $\mathcal{T}$ and Σ, but the above program will be recursive.

Example 5 (continued). Γ_Σ *contains the node pairs $(A_0, \{\exists r.(B_1 \sqcap B_2\})$, $(B_1 \sqcap B_2, \{B_1, B_2\})$, $(B_2, \{\exists s.B_2\})$ and is thus cyclic. This is the expected outcome since, as argued in Example 3, A_0 is not FO-rewritable under $\mathcal{T}$ and Σ*

Now, let $\mathcal{T}' := \mathcal{T} \cup \{B_1 \sqsubseteq B_2\}$ as in the second part of Example 3. The resulting Γ_Σ still contains the above node pairs and is thus cyclic. To find out that A_0 is FO-rewritable under $\mathcal{T}'$ and Σ, the algorithm enters Phase 2.

Phase 2. In the second phase of the algorithm, we construct a set of node tuples Ω_Σ by further annotating (and duplicating) the pairs in Γ_Σ. A *node tuple* takes the form $t = (C_t, S_t, \mathsf{con}_t, E_t, \mathsf{xcon}_t)$ where C_t and S_t have the same form as the components of node pairs in Γ_Σ, $C_t \in \mathsf{con}_t \subseteq \mathsf{sub}(\mathcal{T})$, E_t is the special symbol "$-$" or of the form $\exists s.C$ such that $\exists r.C \in S_t$ for some r with $\mathcal{T} \models s \sqsubseteq r$, and xcon_t is a subset of con_t or "$-$". Intuitively, a node tuple $t \in \Omega_\Sigma$ describes a set of concepts D such that $(C_t, D) \in \widehat{\Gamma}_\Sigma$ and apart from (i) and (ii) above the following additional conditions are satisfied:

(iii) $\mathcal{T} \models D \sqsubseteq E$ iff $E \in \mathsf{con}_t$, for each $E \in \mathsf{sub}(\mathcal{T})$;

(iv) if $E_t = \exists s.C$, then there is a tlc $\exists s.E$ in D and a leaf node in E such that $(C, E) \in \widehat{\Gamma}_\Sigma$ and for the concept D' obtained from D by dropping this node, we have $\mathcal{T} \models D' \sqsubseteq E$ iff $E \in \mathsf{xcon}_t$, for each $E \in \mathsf{sub}(\mathcal{T})$.

When S_t contains no existential restrictions, we use "$-$" for E_t and xcon_t. To understand E_t, it is useful to think of D as a tree and of E_t as a selected successor of the root of that tree. We start the construction of Ω_Σ with setting

$$\Omega_\Sigma = \{(C, S \cap \mathsf{N_C}, \mathsf{con}_\mathcal{T}(S \cap \mathsf{N_C}), -, -) \mid (C, S) \in \Gamma_\Sigma\},$$

where for a set of concepts M, $\mathsf{con}_\mathcal{T}(M)$ denotes the set of concepts $D \in \mathsf{sub}(\mathcal{T})$ such that $\mathcal{T} \models \bigsqcap M \sqsubseteq D$. We call the tuples in the set above *leaf tuples*. The final set Ω_Σ is constructed by exhaustively applying the following rule:

(r_Ω) If $t = (C_t, S_t, \mathsf{con}_t, E_t, \mathsf{xcon}_t)$ is a node tuple with $\exists r_0.D_0, \dots, \exists r_n.D_n$ the existential restrictions in S_t ($n \geq 0$) and there are role names $s_0, \dots, s_n \in \Sigma$ and node tuples $t_0, \dots, t_n \in \Omega_\Sigma$ and an $\ell \in \{0, \dots, n\}$ such that the following conditions are satisfied, then add t to Ω_Σ:

1. $\mathcal{T} \models s_i \sqsubseteq r_i$ and $C_{t_i} = D_i$ for $0 \leq i \leq n$;
2. $E_t = \exists s_\ell.D_\ell$;
3. there is a node pair $(C_t, S) \in \Gamma_\Sigma$ with $S_t \subseteq S$ and $S \cap \mathsf{N_C} = S_t \cap \mathsf{N_C}$;
4. $\mathsf{con}_t = \mathsf{con}_\mathcal{T}(M)$, where
$$M = (S_t \cap \mathsf{N_C}) \cup \{\exists s_i. \bigsqcap \mathsf{con}_{t_i} \mid i \leq n\};$$
5. $\mathsf{xcon}_t = \mathsf{con}_\mathcal{T}(M')$, where
$$M' = (S_t \cap \mathsf{N_C}) \cup \{\exists s_\ell. \bigsqcap \mathsf{xcon}_{t_\ell}\} \cup \{\exists s_i. \bigsqcap \mathsf{con}_{t_i} \mid \ell \neq i \leq n\}.$$

In Point 5, $\exists r.-$ is identified with $\top$. For $t, t' \in \Omega_\Sigma$, we write $t \rightsquigarrow_{\Omega_\Sigma} t'$ if there are $t_0, \dots, t_n \in \Omega_\Sigma$ that satisfy the conditions listed in (r_Ω) and such that $t' = t_\ell$, that is, t' is the tuple that was chosen for the selected successor.

Example 5 (continued). *For the TBox $\mathcal{T}$, Ω_Σ is initialized to*

$$\{\ (A_0, \{A_0\}, \{A_0\}, -, -), \quad (t_a)$$
$$(A_0, \emptyset, \emptyset, -, -), \quad (t_b)$$
$$(B_1 \sqcap B_2, \{B_1, B_2\}, \{B_1, B_2\}, -, -),\ \dots\ \}. \quad (t_c)$$

(r_Ω) *can be applied using $(A_0, \{\exists r.(B_1 \sqcap B_2)\}) \in \Gamma_\Sigma$ for (C_t, S) in Point 3, with $n = \ell = 0$, $s_0 = r$, and $t_0 = t_c$, adding to Ω_Σ the following node tuple t:*

$$(A_0, \{\exists r.(B_1 \sqcap B_2)\}, \{\exists r.(B_1 \sqcap B_2), A_0\}, \exists r.(B_1 \sqcap B_2), \emptyset).$$

We now have $t \rightsquigarrow_{\Omega_\Sigma} t_c$. Note that $M = \{\exists r.B_1 \sqcap B_2\}$ in Point 4 and that $M' = \emptyset$ in Point 5, resulting in $\mathsf{xcon}_t = \emptyset$.

Phase 3. The third and last phase of the algorithm first checks whether an FO-rewriting exists at all and, if so, produces a rewriting that takes the form of a non-recursive monadic datalog program.

We start with introducing the relevant notion of a cycle. A tuple $t \in \Omega_\Sigma$ is a *root tuple* if $C_t = A_0$, $A_0 \in \mathsf{con}_t$ and $A_0 \notin \mathsf{xcon}_t$. A *path through* Ω_Σ is a finite sequence of node tuples $t_1, \dots, t_k$ from Ω_Σ such that $t_i \rightsquigarrow_{\Omega_\Sigma} t_{i+1}$ for $1 \leq i < k$. A tuple $t \in \Omega_\Sigma$ is *looping* if there is a path $t_1, \dots, t_k$ through Ω_Σ such that $k > 1$, $t = t_1$, $\mathsf{con}_t = \mathsf{con}_{t_k}$, and $\mathsf{xcon}_t = \mathsf{xcon}_{t_k}$. We say that Ω_Σ *contains a root cycle* if there are tuples $t, t' \in \Omega_\Sigma$ such that t is a root tuple, t' is a looping tuple, and t' is reachable from t along $\rightsquigarrow_{\Omega_\Sigma}$.

Theorem 7. *A_0 is not FO-rewritable under $\mathcal{T}$ and Σ if and only if Ω_Σ contains a root cycle.*

Example 5 (continued). *After completing Phase 2 for $\mathcal{T}$, the algorithm checks Ω_Σ for cyclicity and finds a root cycle:*

$$(A_0, \{\exists r.(B_1 \sqcap B_2)\}, \{\exists r.(B_1 \sqcap B_2), A_0\}, \exists r.(B_1 \sqcap B_2), \emptyset),$$
$$(B_1 \sqcap B_2, \{B_1, \exists s.B_2\}, \{B_1 \sqcap B_2, B_1, B_2, \exists s.B_2\}, \exists s.B_2, \{B_1\}),$$
$$(B_2, \{\exists s.B_2\}, \{\exists s.B_2, B_2\}, \exists s.B_2, \emptyset).$$

The last node tuple possesses a reflexive $\rightsquigarrow_{\Omega_\Sigma}$ edge and leads to the root cycle.

For the TBox $\mathcal{T}'$, the second tuple in this path will have $\{B_1 \sqcap B_2, B_1, B_2\}$ as xcon_t (we do not need the selected successor to infer B_2 here), and the first tuple above will contain A_0 in xcon_t and will therefore not be a root tuple. Indeed, for $\mathcal{T}'$ the resulting set Ω_Σ does not contain a root cycle.

As suggested by Theorem 7, our algorithm first checks whether Ω_Σ contains a root cycle and, if so, returns 'not FO-rewritable'. Otherwise, it constructs a datalog program $\Pi_{A_0,\mathcal{T}}$ that is a rewriting of A_0 under $\mathcal{T}$ and non-recursive iff A_0 is FO-rewritable under $\mathcal{T}$ and Σ (which is guaranteed at this point by Theorem 7).

The IDB relations of $\Pi_{A_0,\mathcal{T}}$ take the form $P_{C,\mathsf{con},\mathsf{XCON}}$ where $C \in \mathsf{sub}(\mathcal{T})$, con is a subset of $\mathsf{sub}(\mathcal{T})$ and XCON is a set of such subsets or the special set $\{-\}$. We start with rules

$$P_{C_t,\mathsf{con}_t,\{-\}}(x) \leftarrow \bigwedge_{A \in S_t} A(x)$$

for all $t \in \Omega$ with $S_t \subseteq \mathsf{N_C}$ (which are of the form $(C_t, S_t, \mathsf{con}_t, -, -)$) and then exhaustively add a rule

$$P_{C_t,\mathsf{con}_t,\mathsf{XCON}}(x) \leftarrow \bigwedge_{A \in S_t \cap \mathsf{N_C}} A(x) \wedge \bigwedge_{i \leq n} \big(s_i(x, y_i) \wedge P_{C_{t_i},\mathsf{con}_{t_i},\mathsf{XCON}_i}(y_i) \big)$$

for all $t \in \Omega$ with existential restrictions $\exists r_0.D_0, \dots, \exists r_n.D_n$ the existential restrictions in S_t, tuples $t_0, \dots, t_n \in \Omega$, role names $s_0, \dots, s_n \in \Sigma$, $\ell \in \{0, \dots, n\}$, and sets $\mathsf{XCON}_0, \dots, \mathsf{XCON}_n$ such that

1. Conditions 1 to 5 from the rule (r_Ω) hold;
2. $P_{C_{t_i},\mathsf{con}_{t_i},\mathsf{XCON}_i}$ already occurs in $\Pi_{A_0,\mathcal{T}}$, for all $i \leq n$;
3. XCON consists of all sets $\mathsf{con}_\mathcal{T}(M')$ such that there is an $\ell' \leq n$ and an $\mathsf{xcon} \in \mathsf{XCON}_{\ell'}$ with
$$M' = (S_t \cap \mathsf{N_C}) \cup \{\exists s_{\ell'}. \bigsqcap \mathsf{xcon}\} \cup \{\exists s_i. \bigsqcap \mathsf{con}_{t_i} \mid \ell' \neq i \leq n\}.$$

In Point 3 above, we again identify $\exists r.-$ with $\top$. The goal predicates of $\Pi_{A_0,\mathcal{T}}$ are the predicates of the form $P_{A_0,\mathsf{con},\mathsf{XCON}}$ with $A_0 \in \mathsf{con}$ and $A_0 \notin \mathsf{xcon}$ for all $\mathsf{xcon} \in \mathsf{XCON}$. To eliminate 'accidental' recursiveness from $\Pi_{A_0,\mathcal{T}}$, remove all rules that contain a predicate which is not reachable from a goal predicate (defined in the obvious way).

Theorem 8.

1. *The program $\Pi_{A_0,\mathcal{T}}$ is a rewriting of A_0 under $\mathcal{T}$.*
2. *If A_0 is FO-rewritable under $\mathcal{T}$ and Σ, then $\Pi_{A_0,\mathcal{T}}$ is non-recursive.*

Note that $\Pi_{A_0,\mathcal{T}}$ is of double exponential size in the worst case. It is possible to find a (monadic) program that is only single exponential in the worst case, but unlike the program $\Pi_{A_0,\mathcal{T}}$ it is also best-case exponential. This cannot be significantly improved without giving up monadicity.

Theorem 9. *If A_0 is FO-rewritable under $\mathcal{T}$ and Σ, then it has a monadic non-recursive datalog rewriting of size at most $2^{p(n)}$, n the size of $\mathcal{T}$ and $p()$ a polynomial.*

There is a family of TBoxes $\mathcal{T}_1, \mathcal{T}_2, \ldots$ such that for all $n \geq 1$, $\mathcal{T}_n$ is of size $\mathcal{O}(n^2)$, the concept name A_0 is FO-rewritable under $\mathcal{T}_n$, and the smallest non-recursive monadic datalog rewriting has size at least 2^n.

Let us briefly analyze the complexity of the decomposed algorithm. It is easy to verify that the number of Γ-pairs and Ω-tuples is singly exponential in the size of $\mathcal{T}$ and that all required operations for building Γ and Ω and for determining the existence of a root cycle require only polynomial time. By Theorem 7, we have thus found an ExpTime algorithm for deciding FO-rewritability of concept queries under $\mathcal{EL}$-TBoxes and ABox signatures, which is optimal [Bienvenu *et al.*, 2013].

5 Experiments

We have implemented the decomposed algorithm in the *Grind* system and conducted a number of experiments. The system can be downloaded from http://www.informatik.uni-bremen.de/∼hansen/grind, and is released under GPL. In the following, we point out some selected aspects of the implementation. We use numbers to represent subconcepts in $\mathcal{T}$, store the S-component of each pair $(C, S) \in \Gamma$, which is a set of subconcepts of $\mathcal{T}$, as an ordered set, and use *tries* as a data structure to store Γ. We remove pairs $(C, S) \in \Gamma$ where S is not minimal, that is, for which there is a $(C, S') \in \Gamma$ with $S' \subsetneq S$. It can be verified that this optimization does not compromise correctness. We eagerly check for cycles in Γ without waiting for Phase 1 to complete and immediate start Phase 2 once a cycle is found to check whether it gives raise to non-FO-rewritability. If this is not the case, we return to Phase 1.

The experiments were carried out on a Linux (3.2.0) machine with a 3.5 GHz quad-core processor and 8 GB of RAM. Although a large number of $\mathcal{ELH}^{dr}$-TBoxes is available on the web and from various repositories, most of them are *acyclic TBoxes* in the traditional DL sense, that is, the left-hand sides of all CIs are concept names, there are no two CIs with the same left-hand side, and there are no syntactic cycles.

TBox	CI	CN	RN	no	stop	time	RQ stop	RQ time
ENVO	1942	1558	7	7	100%	2s	92.6%	2m52s
FBbi	567	517	1	0	100%	3s	86.1%	19m25s
MOHSE	3665	2203	71	1	99.6%	6m35s	58.7%	7h17m
NBO	1468	962	8	6	100%	3s	61.5%	3h05m
not-galen	4636	2748	159	44	95.9%	1h15m	48.9%	11h43m
SO	3160	2095	12	15	99.8%	4m28s	77.9%	3h53m
XP	1046	906	27	1	100%	27s	0.0%	7h33m

Table 1: TBoxes used in the experiments.

Since concept queries are always FO-rewritable under such TBoxes [Bienvenu *et al.*, 2012a], they are not useful for our experiments. We have identified seven TBoxes that do not fall into this class, listed in Table 1 together with the number of concept inclusions (CI), concept names (CN), and role names (RN) that they contain. Role inclusions occur in MOHSE, not-galen, and SO. All TBoxes together with information about their origin are available at http://www.informatik.uni-bremen.de/∼hansen/grind. All experiments conducted were using the full ABox signature (which in practice is the most difficult case since smaller signatures result in fewer node pairs and node tuples).

For each of these TBoxes, we have applied the decomposed algorithm to every concept name in the TBox. In some rare cases, either the set Γ has reached excessive size or calculation of Ω took too long, resulting in non-termination. We have thus established a 30 second timeout which results in termination in almost all cases; the "stop" column of Table 1 shows the fraction of inputs on which the algorithm successfully terminated, and the "time" column lists the overall runtime needed to process all concept names from the ontology (including timeouts). The generated non-recursive datalog-rewritings are typically of very reasonable size. The number of rules in the rewriting is displayed in the upper part of Figure 2; for example, for NBO, about 55% of all rewritings consist of a single rule, about 18% have two or three rules, about 10% have 4–7 rules, and so on. Note that the x-axis has exponential scale. The size of the rule bodies is typically very small, between one and two atoms in the vast majority of cases, and we have never encountered a rule with more than ten body atoms. It is also interesting to consider the number of IDB predicates in a rewriting, as intuitively these correspond to views that have to be generated by a database system that executes the query. As shown in the lower part of Figure 2, this number is rather small, and considerably lower than the number of rules in the produced programs (we again use exponential scale on the x-axis).

The experiments also confirm our initial belief that ontologies which are used in practical applications have a simple structure. As shown in the "no" column of Table 1, the number of concept names that are not FO-rewritable is extremely small. Moreover, if a concept name was FO-rewritable, then we were always able to find a rewriting already in Phase 1 of our algorithm. Note, though, that for those cases that turned out to be not FO-rewritable, we had to go through Phases 2 and 3.

We have also compared the performance of Grind with that of REQUIEM, which implements an incomplete resolution-

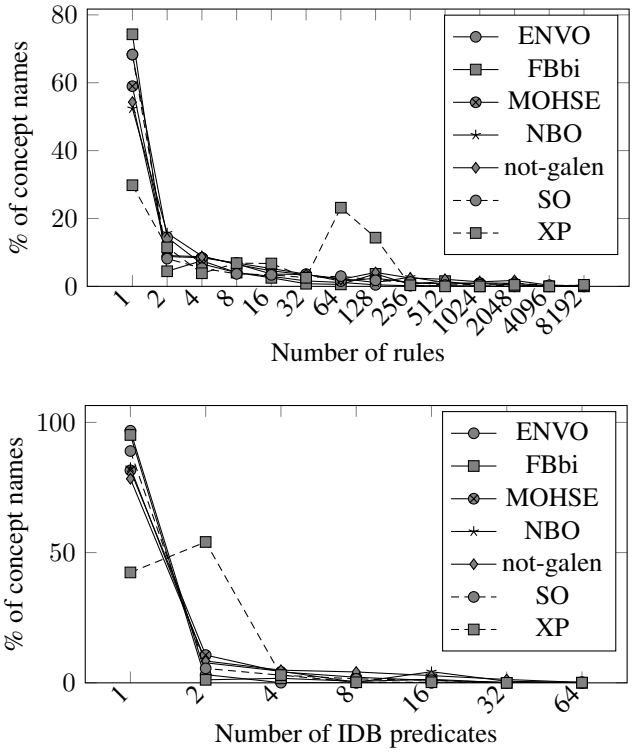

Figure 2: Number of rules and IDB predicates in the rewriting

based approach to computing FO-rewritings for ontology languages up to $\mathcal{ELHI}$ [Pérez-Urbina *et al.*, 2010]. We use the same 30s timeout, which resulted in the termination rate and overall runtime (again including timeouts) displayed in the columns of Table 1 marked with "RQ". The termination cases correspond to positive answers, as REQUIEM cannot determine that an input is not FO-rewritable. Since REQUIEM tends to terminate only on relatively simple inputs, the constructed rewritings of the two tools are often identical. When they are not, either tool may produce a shorter rewriting.

6 Outlook

As future work, we plan to extend the algorithm and implementation from concept queries to conjunctive queries and to more expressive Horn-DLs such as $\mathcal{ELHI}$ and Horn-$\mathcal{SHIQ}$. This is non-trivial for several reasons; for example, with inverse roles the computation of the con and xcon sets in Phase 2 is no longer as straightforward as for $\mathcal{ELH}^{dr}$.

References

[Baader *et al.*, 2005] F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope. In *IJCAI*, pages 364–369, 2005.

[Baader *et al.*, 2007] F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, and P.F. Patel-Schneider. *The Description Logic Handbook*. Cambridge University Press, 2007.

[Baget *et al.*, 2011] J.-F. Baget, M. Leclère, M.-L. Mugnier, and E. Salvat. On rules with existential variables: Walking the decidability line. In *AI*, 175(9-10):1620–1654, 2011.

[Bienvenu *et al.*, 2012a] M. Bienvenu, C. Lutz, and F. Wolter. Deciding FO-rewritability in $\mathcal{EL}$. In *DL*, pages 70–80, 2012.

[Bienvenu *et al.*, 2012b] M. Bienvenu, C. Lutz, and F. Wolter. Query containment in description logics reconsidered. In *KR*, pages 221–231, 2012.

[Bienvenu *et al.*, 2013] M. Bienvenu, C. Lutz, and F. Wolter. First order-rewritability of atomic queries in Horn description logics. In *IJCAI*, pages 754–760, 2013.

[Bienvenu *et al.*, 2014] M. Bienvenu, B. ten Cate, C. Lutz, and F. Wolter. Ontology-based data access: a study through Disjunctive Datalog, CSP, and MMSNP. In *TODS*, 39, 2014.

[Eiter *et al.*, 2012] T. Eiter, M. Ortiz, M. Simkus, T.-K. Tran, G. Xiao. Query rewriting for Horn-$\mathcal{SHIQ}$ plus rules. In *AAAI*, 2012.

[Kaminski and Grau, 2013] M. Kaminski and B. Cuenca Grau. Sufficient conditions for first-order and datalog rewritability in $\mathcal{ELU}$. In *DL*, pages 271–293, 2013.

[König *et al.*, 2012] M. König, M. Leclère, M.-L. Mugnier, and M. Thomazo. A sound and complete backward chaining algorithm for existential rules. In *RR*, pages 122–138, 2012.

[Kontchakov *et al.*, 2013] R. Kontchakov, M. Rodriguez-Muro, and M. Zakharyaschev. Ontology-based data access with databases: A short course. In *Reasoning Web*, pages 194–229, 2013.

[Lutz *et al.*, 2009] C. Lutz, D. Toman, and F. Wolter. Conjunctive query answering in the description logic $\mathcal{EL}$ using a relational database system. In *IJCAI*, pages 2070–2075, 2009.

[Lutz *et al.*, 2013] C. Lutz, İ. Seylan, D. Toman, and F. Wolter. The combined approach to OBDA: Taming role hierarchies using filters. In *ISWC*, pages 314–330, 2013.

[Mora and Corcho, 2013] J. Mora and Ó. Corcho. Engineering optimisations in query rewriting for OBDA. In *I-SEMANTICS*, pages 41–48, 2013.

[Pérez-Urbina *et al.*, 2010] H. Pérez-Urbina, B. Motik, and I. Horrocks. Tractable query answering and rewriting under description logic constraints. In *J. of Applied Logic*, 8(2):186–209, 2010.

[Poggi *et al.*, 2008] A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. Linking data to ontologies. In *J. on Data Semantics*, 10:133–173, 2008.

[Rosati and Almatelli, 2010] R. Rosati and A. Almatelli. Improving query answering over DL-Lite ontologies. In *KR*, pages 290–300, 2010.

[Rosati, 2007] R. Rosati. On conjunctive query answering in $\mathcal{EL}$. In *DL*, pages 451–458, 2007.