Ontology-Mediated Querying with the Description Logic $\mathcal{EL}$: Trichotomy and Linear Datalog Rewritability

Carsten Lutz and **Leif Sabellek**
University of Bremen, Germany
{clu,sabellek}@informatik.uni-bremen.de

Abstract

We consider ontology-mediated queries (OMQs) based on an $\mathcal{EL}$ ontology and an atomic query (AQ), provide an ultimately fine-grained analysis of data complexity and study rewritability into linear Datalog– aiming to capture linear recursion in SQL. Our main results are that every such OMQ is in AC_0, NL-complete or PTIME-complete, and that containment in NL coincides with rewritability into linear Datalog (whereas containment in AC_0 coincides with rewritability into first-order logic). We establish natural characterizations of the three cases, show that deciding linear Datalog rewritability (as well as the mentioned complexities) is EXPTIME-complete, give a way to construct linear Datalog rewritings when they exist, and prove that there is no constant bound on the arity of IDB relations in linear Datalog rewritings.

1 Introduction

An important application of ontologies is to enrich data with a semantics and with domain knowledge while also providing additional vocabulary for query formulation [Calvanese *et al.*, 2009; Kontchakov *et al.*, 2013; Bienvenu *et al.*, 2014; Bienvenu and Ortiz, 2015]. The combination of a traditional database query and an ontology can be viewed as a compound query, commonly referred to as an *ontology-mediated query (OMQ)*. Substantial research efforts have been invested into studying OMQs based on description logic (DL) ontologies, with two dominating topics being the *data complexity* of OMQs [Hustadt *et al.*, 2005; Krisnadhi and Lutz, 2007; Rosati, 2007; Calvanese *et al.*, 2013] and their *rewritability* into more standard database query languages such as SQL (which in this context is often equated with first-order logic) and into Datalog [Pérez-Urbina *et al.*, 2010; Eiter *et al.*, 2012; Bienvenu *et al.*, 2013; 2014; Kaminski *et al.*, 2014; Trivela *et al.*, 2015; Feier *et al.*, 2017]. While the former topic aims to understand the feasibility of OMQs from a theoretical angle, the latter is inspired by rather practical concerns: since most database systems are unaware of ontologies, rewriting OMQs into standard query languages provides an important avenue for implementing OMQ execution in practical applications; however, a major challenge emerges from the fact that the desired rewritings are typically not guaranteed to always exist, though they often do exist in practically relevant cases. Both topics are thoroughly intertwined since rewritability into first-order logic (FO) is closely related to AC_0 data complexity while rewritability into Datalog is closely related to PTIME data complexity.

Modern DLs can roughly be divided into two families: 'expressive DLs' such as $\mathcal{ALC}$ and $\mathcal{SHIQ}$ which typically have CONP data complexity and where rewritability is guaranteed neither into FO nor into Datalog [Bienvenu *et al.*, 2014; Trivela *et al.*, 2015; Feier *et al.*, 2017], and 'Horn DLs' such as $\mathcal{EL}$ and Horn-$\mathcal{SHIQ}$ which typically have PTIME data complexity and where rewritability into Datalog is guaranteed, but FO-rewritability is not [Bienvenu *et al.*, 2013; Hansen *et al.*, 2015; Bienvenu *et al.*, 2016] (with the notable exception of DL-Lite [Calvanese *et al.*, 2009]). In this paper, we consider the OMQ language $(\mathcal{EL}, \text{AQ})$ where the ontology is formulated in the Horn description logic $\mathcal{EL}$ and where the actual queries are *atomic queries (AQs)* of the form $A(x)$, studying data complexity, rewritability, and their relations. Our actual contribution is two-fold.

First, we carry out an ultimately fine-grained analysis of data complexity. In fact, we establish a trichotomy, showing that every OMQ from $(\mathcal{EL}, \text{AQ})$ is in AC_0, NL-complete, or PTIME-complete, a remarkable sparseness of complexities. We also establish elegant characterizations that separate the three classes of OMQs. In particular, we show that an OMQ Q is in NL if there is a bound k such that any minimal tree-shaped ABox $\mathcal{A}$ whose root is an answer to the OMQ Q does not contain a full binary tree of depth k as a minor, and PTIME-hard otherwise. We additionally use a second, more operational characterization to determine the precise complexity of deciding whether a given OMQ is in AC_0, NL-complete, or PTIME-complete, which turns out to be EXPTIME-complete.

And second, we put rewritability into *linear* Datalog onto the agenda of OMQ research. In fact, the equation "SQL = FO" often adopted in this area ignores the fact that SQL contains linear recursion from its version 3 published in 1999 on, which exceeds the expressive power of FO. We believe that, in the context of OMQs, linear Datalog is a natural abstraction of SQL that includes linear recursion, despite the fact that it does not contain full FO. Indeed, all OMQs from $(\mathcal{EL}, \text{AQ})$ that are FO-rewritable are also rewritable into a union of conjunctive queries (UCQ) and thus into linear Datalog (and the same is true for much more expressive OMQ languages) [Bienvenu *et al.*, 2014]. This shows that the expressive power of FO that lies outside of linear Datalog is not useful when using SQL as a target language for OMQ rewriting. We prove that rewritability into linear Datalog coincides with containment in NL. By what was said above, it is thus EXPTIME-complete to decide whether a given OMQ is rewritable. Moreover, we show how to construct linear Datalog rewritings when they exist and prove that there is no constant bound on the arity of IDB relations in linear Datalog rewritings.

Proof details are in the appendix, which is provided at http://www.cs.uni-bremen.de/tdki/research/papers.html.

2 Preliminaries

Let $\mathsf{N_C}$, $\mathsf{N_R}$, and $\mathsf{N_I}$ be countably infinite sets of *concept names*, *role names*, and *individual names*. An $\mathcal{EL}$-*concept* is built according to the syntax rule $C, D ::= \top \mid A \mid C \sqcap D \mid \exists r.C$ where A ranges over concept names and r over role names. An $\mathcal{EL}$-*TBox* is a finite set of *concept inclusions (CIs)* of the form $C \sqsubseteq D$, C and D $\mathcal{EL}$-concepts. The size of $\mathcal{T}$, denoted $|\mathcal{T}|$, is the number of symbols needed to write all CIs of $\mathcal{T}$, with each concept and role name counting as one symbol.

An *ABox* is a finite set of *concept assertions* $A(a)$ and *role assertions* $r(a, b)$ where A is a concept name, r a role name, and a, b individual names. We use $\mathsf{Ind}(\mathcal{A})$ to denote the set of individuals of the ABox $\mathcal{A}$. A *signature* is a set of concept and role names. We often assume that the ABox is formulated in a prescribed signature, which we call the *ABox signature*. An ABox that only uses concept and role names from a signature Σ is called a Σ-*ABox*.

The semantics of DLs is given in terms of an *interpretation* $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$, where $\Delta^\mathcal{I}$ is a non-empty set (the *domain*) and $\cdot^\mathcal{I}$ is the *interpretation function*, assigning to each $A \in \mathsf{N_C}$ a set $A^\mathcal{I} \subseteq \Delta^\mathcal{I}$ and to each $r \in \mathsf{N_R}$ a relation $r^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$. The interpretation function is extended to compound concepts by setting $\top^\mathcal{I} = \Delta^\mathcal{I}$, $(C \sqcap D)^\mathcal{I} = C^\mathcal{I} \cap D^\mathcal{I}$, and $(\exists r.C)^\mathcal{I} = \{d \in \Delta \mid \exists e \in \Delta : (d, e) \in r^\mathcal{I}\}$. An interpretation $\mathcal{I}$ *satisfies* a CI $C \sqsubseteq D$ if $C^\mathcal{I} \subseteq D^\mathcal{I}$, a concept assertion $A(a)$ if $a \in A^\mathcal{I}$, and a role assertion $r(a, b)$ if $(a, b) \in r^\mathcal{I}$. We say that $\mathcal{I}$ is a *model of* a TBox or an ABox if it satisfies all inclusions or assertions in it.

An *atomic query (AQ)* takes the form $A(x)$, A a concept name. We write $\mathcal{A}, \mathcal{T} \models A(a)$ if $a \in \mathsf{Ind}(\mathcal{A})$ and in every model $\mathcal{I}$ of $\mathcal{A}$ and $\mathcal{T}$, we have $a \in A^\mathcal{I}$. An *ontology-mediated query (OMQ)* is a triple $Q = (\mathcal{T}, \Sigma, A(x))$ with $\mathcal{T}$ a TBox, Σ an ABox signature, and $A(x)$ an AQ. We assume w.l.o.g. that A occurs in $\mathcal{T}$. Let $\mathcal{A}$ be a Σ-ABox. We write $\mathcal{A} \models Q(a)$ and say that a is *an answer to* Q *on* $\mathcal{A}$ if $\mathcal{A}, \mathcal{T} \models A(a)$. The *evaluation problem* for Q is to decide, given a Σ-ABox $\mathcal{A}$ and an $a \in \mathcal{A}$, whether $\mathcal{A} \models Q(a)$. When we speak about the complexity of an OMQ Q, we generally mean its evaluation problem. It is thus understood what we mean when saying that Q is in PTIME or NL-hard. We use $(\mathcal{EL}, \text{AQ})$ to denote set of all OMQs $(\mathcal{T}, \Sigma, A(x))$ where $\mathcal{T}$ is an $\mathcal{EL}$-TBox. It is known that all OMQs in $(\mathcal{EL}, \text{AQ})$ are in PTIME [Rosati, 2007; Krisnadhi and Lutz, 2007].

A *Datalog rule* ρ has the form $S(\mathbf{x}) \leftarrow R_1(\mathbf{y}_1) \wedge \cdots \wedge R_n(\mathbf{y}_n)$ where $n > 0$ and $S, R_1, \ldots, R_n$ are relations of any arity and $\mathbf{x}, \mathbf{y}_i$ denote tuples of variables. We refer to $S(\mathbf{x})$ as the *head* of ρ, and to $R_1(\mathbf{y}_1) \wedge \cdots \wedge R_n(\mathbf{y}_n)$ as the *body*. Every variable that occurs in the head of a rule is required to also occur in its body. A *Datalog program* Π is a finite set of Datalog rules with a selected unary *goal relation* goal that does not occur in rule bodies. Relation symbols that occur in the head of at least one rule of Π are *intensional (IDB) relations*, and all remaining relation symbols in Π are *extensional (EDB) relations*. In our context, EDB relations must be unary or binary and are identified with concept names and role names. Note that, by definition, goal is an IDB relation. A Datalog program is *linear* if each rule body contains at most one IDB relation. The *width* of a Datalog program is the maximum arity of non-goal IDB relations used in it and its *diameter* is the maximum number of variables that occur in a rule in Π. For an ABox $\mathcal{A}$ that uses only EDB relations from Π and $a \in \mathsf{Ind}(\mathcal{A})$, we write $\mathcal{A} \models \Pi(a)$ if a is an answer to Π on $\mathcal{A}$, defined in the usual way [Abiteboul *et al.*, 1995].

A Datalog program Π over EDB signature Σ is a *rewriting* of an OMQ $Q = (\mathcal{T}, \Sigma, A(x))$ if for all Σ-ABoxes $\mathcal{A}$ and all $a \in \mathsf{Ind}(\mathcal{A})$, we have $\mathcal{A} \models Q(a)$ iff $\mathcal{A} \models \Pi(a)$. We say that Q is *(linear) Datalog-rewritable* if there is a (linear) Datalog program that is a rewriting of Q. It is well-known that, in $\mathcal{EL}$, all OMQs are Datalog-rewritable. It follows from the results in this paper that there are simple OMQs $Q = (\mathcal{T}, \Sigma, A(x))$ that are not linear Datalog-rewritable, choose e.g. $\mathcal{T} = \{\exists r.A \sqcap \exists s.A \sqsubseteq A\}$ and $\Sigma = \{r, s, A\}$.

Throughout the paper, we generally and without further notice assume TBoxes to be in *normal form*, that is, to contain only concept inclusions of the form $\exists r.A_1 \sqsubseteq A_2$, $\top \sqsubseteq A_1$, $A_1 \sqcap A_2 \sqsubseteq A_3$, $A_1 \sqsubseteq \exists r.A_2$ where all A_i are concept names. Every TBox $\mathcal{T}$ can be converted into a TBox $\mathcal{T}'$ in normal form in linear time [Baader *et al.*, 2005], introducing fresh concept names; the resulting TBox $\mathcal{T}'$ is a conservative extension of $\mathcal{T}$, that is, every model of $\mathcal{T}'$ is a model of $\mathcal{T}$ and, conversely, every model of $\mathcal{T}$ can be extended to a model of $\mathcal{T}'$. Consequently, when $\mathcal{T}$ is replaced in an OMQ $Q = (\mathcal{T}, \Sigma, A_0(x))$ with $\mathcal{T}'$ resulting in an OMQ Q', then Q and Q' are equivalent in the sense that they give the same answers on all Σ-ABoxes. Thus, conversion of the TBox in an OMQ into normal form does not impact evaluation complexity nor rewritability into linear Datalog (or any other language).

We shall often deal with ABoxes that are tree-shaped. By a *tree*, we generally mean a directed (unlabelled) tree $T = (V, E)$, defined in the usual way. Every ABox gives rise to a directed graph $G_\mathcal{A} = (\mathsf{Ind}(\mathcal{A}), \{(a, b) \mid r(a, b) \in \mathcal{A} \text{ for some } r\})$. We say that $\mathcal{A}$ is *tree-shaped* if $G_\mathcal{A}$ is a tree and $r(a, b), s(a, b) \in \mathcal{A}$ implies $r = s$. The importance of tree-shaped ABoxes is due to the fact that OMQs from $(\mathcal{EL}, \text{AQ})$ cannot distinguish between a Σ-ABox and its unraveling into a tree, see [Lutz and Wolter, 2012] or the appendix of this paper.

We introduce some further standard graph theoretic notions for ABoxes. A *homomorphism* from an ABox $\mathcal{A}_1$ to an ABox $\mathcal{A}_2$ is a total function $h : \mathsf{Ind}(\mathcal{A}_1) \to \mathsf{Ind}(\mathcal{A}_2)$ such that $A(a) \in \mathcal{A}_1$ implies $A(h(a)) \in \mathcal{A}_2$ and $r(a, b) \in \mathcal{A}_1$ implies $r(h(a), h(b)) \in \mathcal{A}_2$. We write $\mathcal{A}_1 \to \mathcal{A}_2$ if there is a homomorphism from $\mathcal{A}_1$ to $\mathcal{A}_2$. A directed graph $G = (V, E)$ is a *minor* of an ABox $\mathcal{A}$ if G is a minor of $G_\mathcal{A}$, that is, if

G can be obtained from $G_{\mathcal{A}}$ by deleting edges and vertices and by contracting edges. A *path decomposition* of a directed graph (V, E) is a sequence $S_1, \ldots, S_n$ of subsets of V such that for every $(a, b) \in E$ there is a set S_i with $a, b \in S_i$ and $S_i \cap S_k \subseteq S_j$, for all $i \leq j \leq k$. A path decomposition is an (ℓ, k)-*path decomposition* if $k = \max_{i=1}^{n} |S_i|$ and $\ell = \max_{i=1}^{n-1} |S_i \cap S_{i+1}|$. The *pathwidth* of a directed graph (V, E) is the smallest k such that (V, E) has an $(\ell, k+1)$-path decomposition for some $\ell \geq 0$. We identify the *pathwidth of an ABox* $\mathcal{A}$ with the pathwidth of $G_{\mathcal{A}}$.

3 NL, PTime, Linear Datalog Rewritability

We establish a dichotomy between PTIME and NL for evaluating queries from $(\mathcal{EL}, \text{AQ})$, also showing that containment in NL coincides with rewritability into linear Datalog (unless NL = PTIME). The dichotomy is based on a characterization of containment in NL via a 'bounded amount of branching' in ABoxes whose root is an answer to the query. The linear Datalog programs constructed in the proofs are of unbounded width. We establish a hierarchy theorem which shows that this is unavoidable.

Let $Q = (\mathcal{T}, \Sigma, A_0(x))$ be an OMQ. We say that Q is *unboundedly branching* if for every $k \geq 0$, there is a tree-shaped Σ-ABox $\mathcal{A}$ such that

1. $\mathcal{A}, \mathcal{T} \models A_0(a)$, a the root of $\mathcal{A}$, and $\mathcal{A}$ is minimal with this property (w.r.t. set inclusion)
2. $\mathcal{A}$ has the full binary tree of depth k as a minor.

Otherwise, Q is *boundedly branching*. In the latter case, the *branching limit* of Q is the maximum integer k such that there is a tree-shaped Σ-ABox $\mathcal{A}$ that satisfies Conditions 1 and 2 above. The branching limit is 0 if there is no tree-shaped Σ-ABox $\mathcal{A}$ that satisfies Condition 1.

Example 1. (1) The OMQ $Q_1 = (\mathcal{T}_1, \{A, r, s\}, A(x))$ with $\mathcal{T}_1 = \{\exists r.A \sqsubseteq B_1, \exists s.A \sqsubseteq B_2, B_1 \sqcap B_2 \sqsubseteq A\}$ is unboundedly branching as witnessed by the ABoxes $\mathcal{A}_1, \mathcal{A}_2, \ldots$ where $\mathcal{A}_i$ is a full binary tree of depth i, each left successor connected via the role name r, each right successor via the role name s, and with the concept name A asserted for each leaf.

(2) The OMQ $Q_2 = (\mathcal{T}_2, \{A, r, s\}, B_{12}(x))$ with $\mathcal{T}_2 = \{\exists r.A \sqsubseteq B_1, \exists s.A \sqsubseteq B_2, \exists s.B_2 \sqsubseteq B_2, B_1 \sqcap B_2 \sqsubseteq B_{12}, \exists r.B_{12} \sqsubseteq B_1\}$ is boundedly branching with branching limit one. In fact, every minimal tree-shaped Σ-ABox whose root is an answer to Q_2 consists of a single r-path with an s-path starting at each non-leaf node and with A asserted for each leaf. Note that the number of individuals at which a branching occurs is unbounded in such ABoxes.

The following theorem sums up the results obtained in this section, except for the width hierarchy (Theorem 15).

Theorem 2. *For every OMQ $Q \in (\mathcal{EL}, AQ)$, one of the following applies:*

1. *Q is* PTIME*-hard and not expressible in linear Datalog;*
2. *Q is rewritable into linear Datalog and thus in* NL*.*

Bounded branching of Q implies linear Datalog rewritability and delineates the two cases.

Note that Theorem 2 implies that any OMQ from $(\mathcal{EL}, \text{AQ})$ is linear Datalog rewritable if and only if it is in NL (unless NL = PTIME). It is interesting to compare Theorem 2 with the result by [Afrati and Cosmadakis, 1989] that there are Datalog-queries that are not expressible as a linear Datalog program, but belong to $\mathcal{NC}^2$ and are thus unlikely to be PTIME-hard.

3.1 Characterizations and PTime-Hardness

Theorem 2 provides a characterization of PTIME-hardness in terms of unbounded branching that is elegant, but does not lend itself to hardness proofs very well. For this reason, we establish a second characterization designed to enable a reduction from the PTIME-complete *path systems accessibility* (PSA) problem and show that both characterizations are equivalent. The new characterization will also be handy later on to decide the rewritability of OMQs into linear Datalog.

An instance of PSA takes the form $G = (V, E, S, t)$ where V is a finite set of nodes, E is a ternary relation on V, $S \subseteq V$ is a set of source nodes, and $t \in V$ is a target node. G is a yes instance if t is accessible, where a node $v \in V$ is *accessible* if $v \in S$ or there are accessible nodes u, w with $(u, w, v) \in E$.

Before we can state the new characterization, we need some preliminaries. Let $\mathcal{T}$ be a TBox. A $\mathcal{T}$-*type* is a set t of concept names from $\mathcal{T}$ that is closed under $\mathcal{T}$-consequence, that is, if $\mathcal{T} \models \bigsqcap t \sqsubseteq A$, then $A \in t$. For any ABox $\mathcal{A}$ and $a \in \mathsf{Ind}(\mathcal{A})$, we use $\mathsf{tp}_{\mathcal{A},\mathcal{T}}(a)$ to denote the set of concept names A from $\mathcal{T}$ such that $\mathcal{A}, \mathcal{T} \models A(a)$, which is a $\mathcal{T}$-type. If M is a set of concept names, then by $M(a)$ we denote the ABox $\{A(a) \mid A \in M\}$. We also write $\mathcal{A}, \mathcal{T} \models M(a)$, meaning that $\mathcal{A}, \mathcal{T} \models A(a)$ for all $A \in M$. For every tree-shaped ABox $\mathcal{A}$ and $a \in \mathsf{Ind}(\mathcal{A})$, we use $\mathcal{A}^a$ to denote the sub-tree ABox of $\mathcal{A}$ that has a as the root. Moreover, we use $\mathcal{A}_a$ to denote $\mathcal{A} \setminus \mathcal{A}^a$, that is, the ABox obtained from $\mathcal{A}$ by removing all assertions that involve descendants of a (making a a leaf) and all assertions of the form $A(a)$. We also combine these notations, writing for example $\mathcal{A}^a_{bc}$ for $((\mathcal{A}^a)_b)_c$.

Definition 3. An OMQ $(\mathcal{T}, \Sigma, A_0(x)) \in (\mathcal{EL}, \text{AQ})$ *has the ability to simulate* PSA if there are $\mathcal{T}$-types t_0, t_1 and a tree-shaped Σ-ABox $\mathcal{A}$ with root a and distinguished non-root individuals b, c, d where c and d are distinct incomparable descendants of b such that

1. $\mathcal{A}, \mathcal{T} \models A_0(a)$;
2. $t_1 = \mathsf{tp}_{\mathcal{A},\mathcal{T}}(b) = \mathsf{tp}_{\mathcal{A},\mathcal{T}}(c) = \mathsf{tp}_{\mathcal{A},\mathcal{T}}(d)$;
3. $\mathcal{A}_b \cup t_0(b), \mathcal{T} \not\models A_0(a)$;
4. $\mathsf{tp}_{\mathcal{A}_c \cup t_0(c),\mathcal{T}}(b) = \mathsf{tp}_{\mathcal{A}_d \cup t_0(d),\mathcal{T}}(b) = t_0$.

We define $\mathcal{A}_{\mathsf{finish}} := \mathcal{A}_b$, $\mathcal{A}_{\wedge} := \mathcal{A}^b_{cd}$ and $\mathcal{A}_{\mathsf{start}} := \mathcal{A}^b$.

Example 4. The OMQ Q_1 from Example 1 has the ability to simulate PSA. Figure 1 shows a witnessing ABox $\mathcal{A}$ according to Definition 3 where $t_1 = \{A, B_1, B_2\}$ and $t_0 = \{B_2\}$.

PSA is PTIME-hard under FO-reductions [Immerman, 1999]. Using a reduction from this probem, we show that having the ability to simulate PSA is sufficient for PTIME-hardness under FO-reductions. In particular, we use the ABox $\mathcal{A}_{\wedge}$ from Definition 3 to implement an "and" gate where t_0

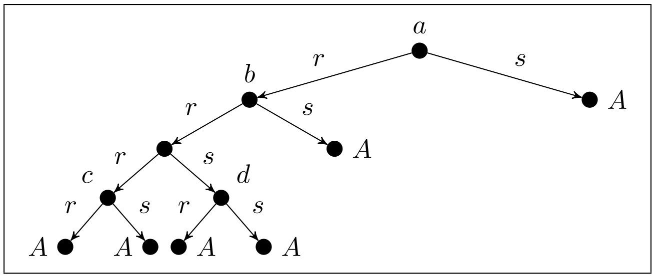

Figure 1: Witness ABox for Example 4

and t_1 represent the truth values zero and one to capture the behaviour of the ternary relation E in PSA.

Lemma 5. *If $Q \in (\mathcal{EL}, AQ)$ has the ability to simulate* PSA*, then Q is* PTIME*-hard under FO-reductions.*

To link Lemma 5 to Theorem 2, we next show that the ability to simulate PSA is equivalent to unbounded branching.

Proposition 6. *Let $Q \in (\mathcal{EL}, AQ)$. Then Q has the ability to simulate* PSA *iff Q is unboundedly branching.*

The "$\Rightarrow$" direction is proved by taking an ABox $\mathcal{A}$ that witnesses the ability to simulate PSA and then glueing together disjoint copies of $\mathcal{A}_\wedge$ to obtain tree-shaped ABoxes whose root is an answer to Q, which are minimal with this property, and that contain deeper and deeper full binary trees as a minor. The "$\Leftarrow$" direction is based on a combinatorial argument: if we take a minimal tree-shaped ABox that makes Q true at the root and contains a deep full binary tree as a minor, then it must contain an ABox that witnesses the ability to simulate PSA.

3.2 NL and Linear Datalog-Rewritability

We show that bounded branching characterizes containment in NL as well as linear Datalog rewritability, which therefore coincide (unless NL $=$ PTIME). We also give a way to construct linear Datalog rewritings when they exist.

Proposition 7. *Let $Q \in (\mathcal{EL}, AQ)$. Then Q is boundedly branching iff Q is rewritable into a linear Datalog program. Moreover, if the branching limit of Q is k, then there is a linear Datalog rewriting of width $k+1$.*

Direction "$\Rightarrow$". Let $Q = (\mathcal{T}, \Sigma, A_0(x))$ be an OMQ from $(\mathcal{EL}, \text{AQ})$. For each $k > 0$, we construct a linear Datalog program $\Pi_{Q,k}$ that is sound as a rewriting of Q and complete on ABoxes that do not have the full binary tree of depth k as a minor. The program $\Pi_{Q,k}$ uses IDB relations of the form $P_{t_1,\dots,t_m}$ where $t_1, \dots, t_m$, are $\mathcal{T}$-types; the arity of this relation is $m \leq k$. For any finite set S of concepts, we use $\mathsf{cl}_\mathcal{T}(S)$ to denote the smallest (w.r.t. set inclusion) $\mathcal{T}$-type t with $\mathcal{T} \models \bigsqcap S \sqsubseteq t$. Let N be the set of all concept names from $\mathcal{T}$. The program $\Pi_{Q,k}$ consists of five types of rules:

Start rules: $P_{\mathsf{cl}_\mathcal{T}(S)}(x) \leftarrow S(x)$ for all $S \subseteq N$ and where $S(x)$ abbreviates $\bigwedge_{A \in S} A(x)$;

Extension rules: $P_{t_1,\dots,t_m,\mathsf{cl}_\mathcal{T}(S)}(x_1, \dots, x_m, y) \leftarrow P_{t_1,\dots,t_m}(x_1, \dots, x_m) \wedge S(y)$ for all $S \subseteq N$ and $\mathcal{T}$-types $t_1, \dots, t_m$;

Step rules: $P_{t_1,\dots,t_{m-1},t}(x_1, \dots, x_{m-1}, y) \leftarrow P_{t_1,\dots,t_m}(x_1, \dots, x_m) \wedge r(y, x_m) \wedge S(y)$ for all $S \subseteq N$ and $\mathcal{T}$-types $t_1, \dots, t_m$ where $t = \mathsf{cl}_\mathcal{T}(S \cup \{\exists r.A \mid A \in t_m\})$;

Consolidation rules: $P_{t_1,\dots,t_{m-2},t}(x_1, \dots, x_{m-1}) \leftarrow P_{t_1,\dots,t_m}(x_1, \dots, x_{m-1}, x_{m-1})$ for all $S \subseteq N$ and $\mathcal{T}$-types $t_1, \dots, t_m, t$ where $t = \mathsf{cl}_\mathcal{T}(t_{m-1} \cup t_m)$;

Goal rules: $\mathsf{goal}(x) \leftarrow P_t(x)$ for all $\mathcal{T}$-types t with $A_0 \in t$.

Example 8. We give a fragment of the program $\Pi_{Q_2,2}$ for the OMQ Q_2 from Example 1 that is equivalent to the full $\Pi_{Q_2,2}$ and showcases the purpose of the different rules. For readability, we use representative concept names in the subscript of IDB relations instead of types:

$$P_A(x) \leftarrow A(x) \qquad P_{B_1}(x) \leftarrow r(x,y) \wedge P_A(y)$$
$$P_{B_1,A}(x,y) \leftarrow P_{B_1}(x) \wedge A(y)$$
$$P_{B_1,B_2}(x,y) \leftarrow s(y,z) \wedge P_{B_1,A}(x,z)$$
$$P_{B_1,B_2}(x,y) \leftarrow s(y,z) \wedge P_{B_1,B_2}(x,z)$$
$$P_{B_{12}}(x) \leftarrow P_{B_1,B_2}(x,x)$$
$$P_{B_1}(x) \leftarrow r(x,y) \wedge P_{B_{12}}(y) \qquad \mathsf{goal}(x) \leftarrow P_{B_{12}}(x)$$

It can be verified that the program $\Pi_{Q,k}$ is sound, that is, $\mathcal{A} \models \Pi_{Q,k}(a)$ implies $\mathcal{A} \models Q(a)$ for any Σ-ABox $\mathcal{A}$. The following lemma states a form of completeness.

Lemma 9. *If $\mathcal{A}$ is a tree-shaped ABox with root a_0 that does not have the full binary tree of depth k as a minor and $\mathcal{A}, \mathcal{T} \models A_0(a_0)$, then $\mathcal{A} \models \Pi_{Q,k}(a_0)$.*

Lemma 9 is proved by exhibiting a suitable strategy for applying the rules in $\Pi_{Q,k}$. Returning to the "$\Rightarrow$" direction of Proposition 7, we next show the following.

Lemma 10. *If $k-1$ is the branching limit of Q, then $\Pi_{Q,k}$ is a rewriting of Q.*

The programs $\Pi_{Q,k}$ allow us to construct a linear Datalog rewriting of an OMQ Q provided that we know an upper bound on its branching limit. The following lemma establishes such an upper bound (in case that Q is rewritable into linear Datalog at all).

Lemma 11. *If $Q = (\mathcal{T}, \Sigma, A_0(x)) \in (\mathcal{EL}, AQ)$ is boundedly branching, then its branching limit is at most $2^{4^{|\mathcal{T}|+1}}$.*

In fact, Lemma 11 is a consequence of the proof of Proposition 6, given in the appendix. Lemma 11 almost yields decidability of linear Datalog rewritability: guess a tree-shaped Σ-ABox $\mathcal{A}$ and verify that it satisfies Conditions 1 and 2 from the definition of k-branching, where k is the bound from Lemma 11. For this to work, we would additionally have to bound the depth and degree of the tree-shaped ABoxes to be guessed. While this is not too difficult, we follow a different route (in Section 5) to obtain tight complexity bounds.

Direction "$\Leftarrow$". For $d, k, n \geq 0$, let $\ell_d^k(n)$ denote the maximum number of leaves in any tree that has degree d, depth n, and does not have as a minor the full binary tree of depth $k+1$. The following lemma says that $\ell_d^k(n)$ as a function of n grows like a polynomial of degree k.

Lemma 12. $(d-1)^k(n-k)^k \leq \ell_d^k(n) \leq (k+1)(d-1)^k n^k$ *for all $d, k \geq 0$ and $n \geq 2k$.*

Let Π be a Datalog program over EDB signature Σ and IDB signature Σ_I, and let $\mathcal{A}$ a Σ-ABox. It is standard to characterize answers to Π in terms of derivations that take the form of a labelled tree, see [Abiteboul *et al.*, 1995] or the appendix. From each derivation D, one can read off an ABox $\mathcal{A}_D$ in a standard way such that the properties summarized by the following lemma are satisfied.

Lemma 13. *Let D be a derivation of $\Pi(a)$ in $\mathcal{A}$, Π of diameter d. Then*

1. $\mathcal{A}_D \models \Pi(a)$;
2. *there is a homomorphism h from $\mathcal{A}_D$ to $\mathcal{A}$ with $h(a) = a$;*
3. *$\mathcal{A}_D$ has pathwidth at most d.*

We are now ready to prove the desired result.

Lemma 14. *If $Q \in (\mathcal{EL}, AQ)$ is unboundedly branching, then it is not rewritable into a linear Datalog program.*

The proof, inspired by [Afrati and Cosmadakis, 1989], is by contradiction. Assume that $Q \in (\mathcal{EL}, \text{AQ})$ is unboundedly branching, but rewritable into a linear Datalog program Π. We choose a minimal tree-shaped Σ-ABox $\mathcal{A}$ that contains a full binary tree of very large depth as a minor and such that $\mathcal{A} \models Q(a_0)$, a_0 the root of $\mathcal{A}$. Consider the derivation D of $\Pi(a_0)$ in $\mathcal{A}$ and the associated ABox $\mathcal{A}_D$. By a sequence of manipulations, we identify a tree-shaped sub-ABox $\mathcal{B} \subseteq \mathcal{A}_D$ such that $\mathcal{B}$ has a very large number of leaves (a consequence of Point 2 of Lemma 13 and the fact that the homomorphism must be surjective due to the minimality of $\mathcal{A}$ and Point 1 of that lemma). By Lemma 12, it follows that $\mathcal{B}$ must contain a full binary tree of large depth as a minor and therefore must have high pathwidth, in contrast to Point 3 of Lemma 13.

3.3 Width Hierarchy

The linear Datalog rewritings constructed in the previous section are of unbounded width. We next show that this is unavoidable, in contrast to the fact that every OMQ from $(\mathcal{EL}, \text{AQ})$ can be rewritten into a *monadic* Datalog program [Baader *et al.*, 2017]. It strengthens a result by [Dalmau and Krokhin, 2008] who establish an analogous statement for constraint satisfaction problems (CSPs). However, while every OMQ from $(\mathcal{EL}, \text{AQ})$ is equivalent to a CSP (up to complementation [Bienvenu *et al.*, 2014]), the converse is false and indeed the CSPs used by Dalmau and Krokhin are not equivalent to an OMQ from $(\mathcal{EL}, \text{AQ})$.

Theorem 15. *For every $\ell > 0$, there is an OMQ from $(\mathcal{EL}, AQ)$ that is rewritable into linear Datalog, but not into a linear Datalog program of width ℓ.*

To prove Theorem 15, we use the following queries: for all $k \geq 1$, let $Q_k = (\mathcal{T}_k, \Sigma, A_k(x))$ where $\Sigma = \{r, s, t, u\}$ and

$$\begin{aligned}
\mathcal{T}_k \quad = \quad & \{\top \sqsubseteq A_0\} \cup \\
& \{\exists x.A_i \sqsubseteq B_{x,i} \mid x \in \{r, s, t, u\}, 0 \leq i \leq k-1\} \cup \\
& \{\exists x.B_{x,i} \sqsubseteq B_{x,i} \mid x \in \{r, s, t, u\}, 0 \leq i \leq k-1\} \cup \\
& \{B_{r,i} \sqcap B_{s,i} \sqsubseteq A_{i+1} \mid 0 \leq i \leq k-1\} \cup \\
& \{B_{t,i} \sqcap B_{u,i+1} \sqsubseteq A_{i+1} \mid 0 \leq i \leq k-1\}.
\end{aligned}$$

In the OMQ Q_k, each concept name A_i, $i \leq k$, represents the existence of a full binary tree of depth i, that is, if A_i is derived at the root of a tree-shaped Σ-ABox $\mathcal{A}$, then $\mathcal{A}$ contains the full binary tree of depth i as a minor. Thus, deriving Q_k at the root implies that $\mathcal{A}$ has the full binary tree of depth k as a minor.

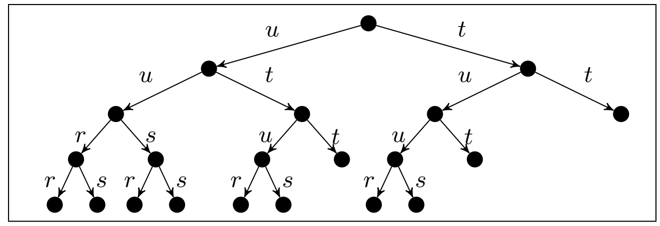

Figure 2: An ABox of depth 4 whose root is an answer to Q_2 and which is minimal with this property. It has 11 leaves, the largest number of leaves that a binary tree of depth 4 can have, unless it contains the full binary tree of depth 3 as a minor.

Furthermore, for every $n \geq k$ there is minimal tree-shaped Σ-ABox $\mathcal{A}$ such that Q_k is derived at the root, $\mathcal{A}$ is of depth n, and $\mathcal{A}$ has the maximum number of leaves that any tree of depth n without the full binary tree of depth $k+1$ as a minor can have. For the case $k = 2$ and $n = 4$, such an ABox is shown in Figure 2. The concept inclusions $\exists x.B_{x,i} \sqsubseteq B_{x,i}$ in $\mathcal{T}_k$ ensure that Q_k is closed under subdivisions of ABoxes, that is, if $\mathcal{A}$ is a Σ-ABox and $\mathcal{A}'$ is obtained from $\mathcal{A}$ by subdividing an edge into a path (using the same role name as the original edge), then $\mathcal{A} \models Q_k(a)$ iff $\mathcal{A}' \models Q_k(a)$ for all $a \in \mathsf{Ind}(\mathcal{A})$.

Lemma 16. *Every Q_k is rewritable into linear Datalog.*

We prove Lemma 16 by showing that each Q_k is boundedly branching with branching limit k and using Proposition 7.

To show that linear Datalog rewritings of the defined family of OMQs require unbounded width, we first show that they require unbounded diameter and then proceed by showing that the width of rewritings cannot be significantly smaller than the required diameter. To make the latter step work, we actually show the former on an infinite family of classes of ABoxes of restricted shape. More precisely, for all $i \geq 0$ we consider the class $\mathfrak{C}_i$ of all forest-shaped Σ-ABoxes in which the distance between any two branching individuals exceeds i (where a forest is a disjoint union of trees and a branching individual is one that has at least two successors). Since the queries Q_k are closed under the subdivision of ABoxes, each class $\mathfrak{C}_i$ contains ABoxes whose root is an answer to the query. The proof of the following is similar to the proof of Lemma 14.

Lemma 17. *For any $i \geq 0$, Q_{2k+3} is not rewritable into a linear Datalog program of diameter k on the class of ABoxes $\mathfrak{C}_i$.*

We are now ready to establish the hierarchy.

Proposition 18. *$Q_{8\ell+13}$ is not rewritable into a linear Datalog program of width ℓ.*

The proof of Proposition 18 is by contradiction. Assume that $Q_{8\ell+13}$ is rewritable into a linear Datalog program Π of width ℓ and let k be the diameter of Π. We show that, on the class of ABoxes $\mathfrak{C}_k$, there must then be a linear Datalog rewriting Π' of $Q_{8\ell+13}$ of diameter $4\ell + 5$, contradicting Lemma 17. In fact, Π' can be obtained from Π by a sequence of manipulations: first rewrite the rules such that the restriction of rule bodies to EDB relations takes the form of a forest in which there is at most one branching node in every tree, then further rewrite to achieve that each such forest contains at most 2ℓ trees, and finally replace each rule with a set of rules of small diameter, slightly increasing the width.

4 AC_0 vs. NL: Completing the Trichotomy

We say that an OMQ $Q = (\mathcal{T}, \Sigma, A_0(x))$ has *unbounded depth* if for every $k \geq 0$, there is a tree-shaped ABox $\mathcal{A}$ with depth at least k and root a such that $\mathcal{A}, \mathcal{T} \models A_0(a)$ and $\mathcal{A}$ is minimal with this property (regarding set inclusion). The following theorem summarizes the results in this section.

Theorem 19. *For every OMQ $Q \in (\mathcal{EL}, AQ)$, one of the following applies:*

1. *Q is FO-rewritable and thus in AC_0.*
2. *Q is not FO-rewritable and* NL*-hard.*

Unbounded depth of Q implies NL*-hardness and delineates the two cases.*

The following characterization of FO-rewritability was established in [Bienvenu *et al.*, 2013].

Theorem 20. *Let $Q \in (\mathcal{EL}, AQ)$. Q is not FO-rewritable iff Q has unbounded depth.*

To prove Theorem 19, it thus remains to show that unbounded depth implies NL-hardness. Similarly to the case of PTIME-hardness, the elegant condition of unbounded depth does not directly lend itself to hardness proofs, and we thus establish a second and equivalent characterization. Here, the second characterization is tailored towards NL-hardness proofs via reduction from reachability in directed graphs (REACH).

Definition 21. An OMQ $(\mathcal{T}, \Sigma, A_0(x)) \in (\mathcal{EL}, \mathrm{AQ})$ has *the ability to simulate* REACH iff there are $\mathcal{T}$-types $t_0 \subsetneq t_1$ and a tree-shaped ABox $\mathcal{A}$ with root a and distinguished non-root individuals b, c where c is a descendant of b such that

1. $\mathcal{A}, \mathcal{T} \models A_0(a)$,
2. $t_1 = \mathsf{tp}_{\mathcal{A},\mathcal{T}}(b) = \mathsf{tp}_{\mathcal{A},\mathcal{T}}(c)$,
3. $\mathsf{tp}_{\mathcal{A}_c \cup t_0(c),\mathcal{T}}(b) = t_0$, and
4. $\mathcal{A}_b \cup t_0(b), \mathcal{T} \not\models A_0(a)$.

We define $\mathcal{A}_{\mathsf{finish}} = \mathcal{A}_b$, $\mathcal{A}_{\mathsf{edge}} = \mathcal{A}^b_c$, and $\mathcal{A}_{\mathsf{start}} = \mathcal{A}^c$.

The three defined sub-ABoxes can be used in a reduction from REACH to Q. We now prove that unbounded depth implies NL-hardness, proceeding via the ability to simulate REACH. The following lemma is essentially implicit already in [Bienvenu *et al.*, 2013].

Lemma 22. *Let $Q \in (\mathcal{EL}, AQ)$. If Q has unbounded depth, then Q has the ability to simulate* REACH*.*

The next lemma is proved similarly to Lemma 5.

Lemma 23. *Let $Q \in (\mathcal{EL}, AQ)$. If Q has the ability to simulate* REACH*, then Q is* NL*-hard under FO-reductions.*

We have completed the proof of Theorem 19, and thus also of the trichotomy.

5 Decidability and Complexity

We first show that an existing reduction in [Bienvenu *et al.*, 2013] yields a variety of relevant hardness results, under various complexity-theoretic assumptions.

Theorem 24. *The following properties of OMQs from $(\mathcal{EL}, AQ)$ are* EXPTIME*-hard: linear Datalog rewritability, containment in* NL *(unless* NL = PTIME*),* NL*-hardness (unless* L = NL*), and* PTIME*-hardness (unless* L = PTIME*).*

For NL-hardness and PTIME-hardness, the complexity-theoretic assumptions can be dropped when hardness is defined under FO-reductions, as a consequence of the fact that Lemma 5 establishes hardness under such reductions.

Regarding upper bounds, we first recall the known result that it is in EXPTIME to decide whether an OMQ from $(\mathcal{EL}, \mathrm{AQ})$ is FO-rewritable [Bienvenu *et al.*, 2013] and observe that, by Theorem 19, we also obtain an EXPTIME upper bound for NL-hardness. For linear Datalog rewritability, containment in NL, and PTIME-hardness, we use an approach based on (one-way) alternating parity automata on finite trees (APTAs). Because of space constraints, we can only give a brief sketch. By Theorem 2 and Proposition 6, it suffices to decide whether a given OMQ has the ability to simulate PSA, that is, whether there are $\mathcal{T}$-types t_0, t_1 and a tree-shaped Σ-ABox $\mathcal{A}$ that satisfy the conditions from Definition 3. We iterate over all choices for t_0, t_1, building for each choice an APTA $\mathfrak{A}_{t_0,t_1}$ that accepts precisely the tree-shaped Σ-ABoxes satisfying the required conditions for the chosen t_0, t_1.

Theorem 25. *It is in* EXPTIME *to decide whether a given OMQ from $(\mathcal{EL}, AQ)$ is rewritable into linear Datalog.*

Interestingly, it is rather unclear how an EXPTIME upper bound would be established based on the characterization in terms of bounded branching. The following corollary sums up the results obtained in this section.

Corollary 26. *For OMQs from $(\mathcal{EL}, AQ)$, all of the following problems are* EXPTIME*-complete (under the same complexity theoretic assumptions for the lower bounds as in Theorem 24): linear Datalog rewritability, containment in* NL*,* NL*-hardness, and* PTIME*-hardness.*

Note that Theorem 25 and the results from Section 3.2 give an algorithm that provides a linear Datalog rewriting of a given OMQ if it exists and reports failure otherwise.

6 Conclusion

We plan to extend our analysis to $(\mathcal{ELI}, \mathrm{AQ})$ where $\mathcal{ELI}$ is the extension of $\mathcal{EL}$ with inverse roles. Then the overall picture changes because there are OMQs from $(\mathcal{ELI}, \mathrm{AQ})$ that express a form of undirected reachability and are L-complete. This also raises the question whether L-completeness coincides with rewritability into symmetric Datalog [Egri *et al.*, 2007]. However, even lifting to $(\mathcal{ELI}, \mathrm{AQ})$ the results established in this paper such as the dichotomy between NL and PTIME is non-trivial. It would also be interesting to replace AQs with conjunctive queries. As illustrated by [Bienvenu *et al.*, 2013] versus [Bienvenu *et al.*, 2016], this makes the technical development more awkward since it requires to replace tree-shaped ABoxes with (somewhat contrived) ABoxes that are almost a tree. It might be more elegant to directly move to frontier-one tuple generating dependencies [Baget *et al.*, 2009]. It would also be interesting to study the size of linear Datalog rewritings, to find ways to construct such rewritings that are efficiently executable, and to analyze empirically whether linear recusion is sufficiently well optimized in SQL database systems to support the rewritten queries.

Acknowledgements

The authors were funded by ERC consolidator grant 647289.

References

[Abiteboul *et al.*, 1995] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[Afrati and Cosmadakis, 1989] Foto N. Afrati and Stavros S. Cosmadakis. Expressiveness of restricted recursive queries (extended abstract). In *Proc. of STOC*, pages 113–126, 1989.

[Baader *et al.*, 2005] Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the $\mathcal{EL}$ envelope. In *Proc. of IJCAI*, pages 364–369, 2005.

[Baader *et al.*, 2017] Franz Baader, Ian Horrocks, Carsten Lutz, and Ulrike Sattler. *An Introduction to Description Logics*. Cambride University Press, 2017.

[Baget *et al.*, 2009] Jean-François Baget, Michel Leclère, Marie-Laure Mugnier, and Eric Salvat. Extending decidable cases for rules with existential variables. In *Proc. of IJCAI*, pages 677–682, 2009.

[Bienvenu and Ortiz, 2015] Meghyn Bienvenu and Magdalena Ortiz. Ontology-mediated query answering with data-tractable description logics. In *Proc. of Reasoning Web*, volume 9203 of *LNCS*, pages 218–307. Springer, 2015.

[Bienvenu *et al.*, 2013] Meghyn Bienvenu, Carsten Lutz, and Frank Wolter. First order-rewritability of atomic queries in horn description logics. In *Proc. of IJCAI*, 2013.

[Bienvenu *et al.*, 2014] Meghyn Bienvenu, Balder ten Cate, Carsten Lutz, and Frank Wolter. Ontology-based data access: A study through disjunctive datalog, CSP, and MMSNP. *ACM Trans. Database Syst.*, 39(4):33:1–33:44, 2014.

[Bienvenu *et al.*, 2016] Meghyn Bienvenu, Peter Hansen, Carsten Lutz, and Frank Wolter. First order-rewritability and containment of conjunctive queries in horn description logics. In *Proc. of IJCAI*, 2016.

[Calvanese *et al.*, 2009] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, Mariano Rodriguez-Muro, and Riccardo Rosati. Ontologies and databases: The DL-Lite approach. In *Proc. of Reasoning Web 2009*, pages 255–356, 2009.

[Calvanese *et al.*, 2013] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Data complexity of query answering in description logics. *Artif. Intell.*, 195:335–360, 2013.

[Dalmau and Krokhin, 2008] Víctor Dalmau and Andrei A. Krokhin. Majority constraints have bounded pathwidth duality. *Eur. J. Comb.*, 29(4):821–837, 2008.

[Egri *et al.*, 2007] László Egri, Benoit Larose, and Pascal Tesson. Symmetric datalog and constraint satisfaction problems in LogSpace. *Electronic Colloquium on Computational Complexity (ECCC)*, 14(024), 2007.

[Eiter *et al.*, 2012] Thomas Eiter, Magdalena Ortiz, Mantas Simkus, Trung-Kien Tran, and Guohui Xiao. Query rewriting for Horn-SHIQ plus rules. In *Proc. of AAAI*. AAAI Press, 2012.

[Feier *et al.*, 2017] Cristina Feier, Antti Kuusisto, and Carsten Lutz. Rewritability in monadic disjunctive datalog, MMSNP, and expressive description logics. In *Proc. of ICDT*, 2017.

[Hansen *et al.*, 2015] Peter Hansen, Carsten Lutz, İnanç Seylan, and Frank Wolter. Efficient query rewriting in the description logic EL and beyond. In *Proc. of IJCAI*, 2015.

[Hustadt *et al.*, 2005] Ullrich Hustadt, Boris Motik, and Ulrike Sattler. Data complexity of reasoning in very expressive description logics. In *Proc. of IJCAI*, pages 466–471. Professional Book Center, 2005.

[Immerman, 1999] Neil Immerman. *Descriptive complexity*. Graduate texts in computer science. Springer, 1999.

[Kaminski *et al.*, 2014] Mark Kaminski, Yavor Nenov, and Bernardo Cuenca Grau. Datalog rewritability of disjunctive datalog programs and its applications to ontology reasoning. In *Proc. of AAAI*, pages 1077–1083. AAAI Press, 2014.

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

[Krisnadhi and Lutz, 2007] Adila Krisnadhi and Carsten Lutz. Data complexity in the EL family of description logics. In Nachum Dershowitz and Andrei Voronkov, editors, *Proc. of LPAR*, volume 4790 of *LNAI*, pages 333–347. Springer, 2007.

[Lutz and Wolter, 2010] Carsten Lutz and Frank Wolter. Deciding inseparability and conservative extensions in the description logic EL. *J. Symb. Comput.*, 45(2):194–228, 2010.

[Lutz and Wolter, 2012] Carsten Lutz and Frank Wolter. Non-uniform data complexity of query answering in description logics. In *Proc. of KR*, 2012.

[Pérez-Urbina *et al.*, 2010] Héctor Pérez-Urbina, Boris Motik, and Ian Horrocks. Tractable query answering and rewriting under description logic constraints. *Journal of Applied Logic*, 8(2):186–209, 2010.

[Rosati, 2007] Riccardo Rosati. The limits of querying ontologies. In *Proc. of ICDT*, volume 4353 of *LNCS*, pages 164–178. Springer, 2007.

[Scheffler, 1989] Petra Scheffler. *Die Baumweite von Graphen als ein Mass für die Kompliziertheit algorithmischer Probleme*. Report (Karl-Weierstrass-Institut für Mathematik). Akademie der Wissenschaften der DDR, Karl-Weierstrass-Institut für Mathematik, 1989.

[Trivela *et al.*, 2015] Despoina Trivela, Giorgos Stoilos, Alexandros Chortaras, and Giorgos B. Stamou. Optimising resolution-based rewriting algorithms for OWL ontologies. *J. Web Sem.*, 33:30–49, 2015.