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 $\ell \in \mathbb{N}$. 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

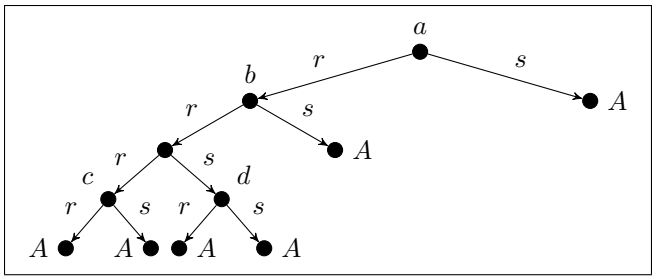

Figure 1: Witness ABox for Example 4

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 $2^{4^{|\mathcal{T}|+1}}$.*

It can be verified that Lemma 11 is a consequence of the proof of Proposition 6. 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 $Q \in (\mathcal{EL}, \text{AQ})$ 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 $(\mathcal{EL}, \text{AQ})$ 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

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

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.

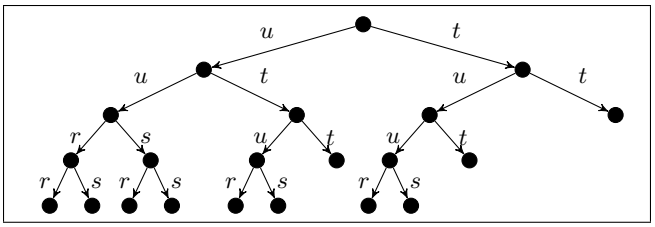

Figure 2: An ABox of depth 4 whose root is an answer to Q_2 and which is minimal with this property. It has 11 leaves, the largest number of leaves that a binary tree of depth 4 can have, unless it contains the full binary tree of depth 3 as a minor.

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 $(\mathcal{T}, \Sigma, A_0(x)) \in (\mathcal{EL}, \text{AQ})$ 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 $(\mathcal{EL}, \text{AQ})$ 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

As future work, we plan to extend our analysis to $(\mathcal{ELI}, \text{AQ})$ where $\mathcal{ELI}$ is the extension of $\mathcal{EL}$ with inverse roles. Then the overall picture changes because there are OMQs from $(\mathcal{ELI}, \text{AQ})$ 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]. Note, though, that even lifting to $(\mathcal{ELI}, \text{AQ})$ 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 'CODA'.

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.

A Two Basic Lemmas

We start with two basic observations. Let $\mathcal{A}$ be a Σ-ABox and $a \in \mathsf{Ind}(\mathcal{A})$. A *path in $\mathcal{A}$ starting at a* is a sequence $p = a_0 r_0 a_1 r_{1,2}, \dots r_{n-1} a_n \in \mathsf{Ind}(\mathcal{A})$ such that $a_0 = a$ and $r_i(a_i, a_{i+1}) \in \mathcal{A}$ for all $i < n$. We use $\mathsf{tail}(p)$ to denote a_n. The *unraveling of $\mathcal{A}$ at a* is the (possibly infinite) ABox whose individuals are the paths in $\mathcal{A}$ starting at a and which contains the assertion $A(p)$ whenever $A(\mathsf{tail}(p)) \in \mathcal{A}$ and $r(p, prb)$ whenever prb is a path in $\mathcal{A}$. The following is well-known [Lutz and Wolter, 2012].

Lemma 27. *Let $Q = (\mathcal{T}, \Sigma, A(x))$ be an OMQ from $(\mathcal{EL}, AQ)$ and $\mathcal{A}$ a Σ-ABox with a distinguished node a. Then $\mathcal{A} \models Q(a)$ iff $\mathcal{A}^u \models Q(a)$ where $\mathcal{A}^u$ is the unraveling of $\mathcal{A}$ at a.*

The *degree* of an ABox $\mathcal{A}$ is the maximum number of successors of any individual in $\mathcal{A}$. The following lemma often allows us to concentrate on ABoxes of small degree. It is proved using a chase procedure.

Lemma 28. *Let $Q = (\mathcal{T}, \Sigma, A_0(x)) \in (\mathcal{EL}, AQ)$ be an OMQ and $\mathcal{A}$ a Σ-ABox such that $\mathcal{A} \models Q(a_0)$. Then there is an $\mathcal{A}' \subseteq \mathcal{A}$ of degree at most $|\mathcal{T}|$ such that $\mathcal{A}' \models Q(a_0)$.*

Proof. (sketch) As a preliminary, consider the chase procedure that, given an ABox $\mathcal{A}$, exhaustively applies to it the following rules:

R1 If $A_1(a), A_2(a) \in \mathcal{A}$, $A_1 \sqcap A_2 \sqsubseteq A_3 \in \mathcal{T}$, and $A_3(a) \notin \mathcal{A}$, then add $A_3(a)$ to $\mathcal{A}$;

R2 If $\top \sqsubseteq A_1 \in \mathcal{T}$, and $A_1(a) \notin \mathcal{A}$ for some $a \in \mathsf{Ind}(\mathcal{A})$, then add $A_1(a)$ to $\mathcal{A}$;

R3 If $r(a, b), A_1(b) \in \mathcal{A}$, $\exists r.A_1 \sqsubseteq A_2 \in \mathcal{T}$, and $A_2(a) \notin \mathcal{A}$, then add $A_2(a)$ to $\mathcal{A}$;

R4 If $A_1(a) \in \mathcal{A}$, $\mathcal{T} \models A_1 \sqsubseteq A_2$, and $A_2(a) \notin \mathcal{A}$, then add $A_2(a)$ to $\mathcal{A}$.

Let exhaustive rule applications result in the (unique) ABox $\mathcal{A}_c$. It is standard to prove that for any concept name A and $a \in \mathsf{Ind}(\mathcal{A})$, we have $\mathcal{A}, \mathcal{T} \models A(a)$ iff $A(a) \in \mathcal{A}_c$.

Now assume $\mathcal{A} \models Q(a_0)$ and let $\mathcal{A}_c$ be the completed ABox produced by the chase. Since $\mathcal{A} \models Q(a_0)$, $A_0(a_0) \in \mathcal{A}_c$. Let $\mathcal{A}'$ be obtained from $\mathcal{A}$ by removing all assertions $r(a, b)$ that did not participate in any application of rule **R3**, and let $\mathcal{A}'_c$ be the result of chasing $\mathcal{A}'$. Clearly, we must have $A_0(a_0) \in \mathcal{A}'_c$. Moreover, it is easy to verify that the degree of $\mathcal{A}'$ is at most $|\mathcal{T}|$. □

B Proofs for Section 3

Lemma 29. *Let $\mathcal{A}_1, \mathcal{A}_2$ be Σ-ABoxes and $\mathcal{T}$ a TBox such that $\mathsf{tp}_{\mathcal{A}_1,\mathcal{T}}(a) = \mathsf{tp}_{\mathcal{A}_2,\mathcal{T}}(a)$ for all $a \in \mathsf{Ind}(\mathcal{A}_1) \cap \mathsf{Ind}(\mathcal{A}_2)$. Then $\mathsf{tp}_{\mathcal{A}_1 \cup \mathcal{A}_2,\mathcal{T}}(a) = \mathsf{tp}_{\mathcal{A}_i,\mathcal{T}}(a)$ for all $a \in \mathsf{Ind}(\mathcal{A}_i)$, $i \in \{1, 2\}$.*

Proof. Let $\mathcal{A}_1$, $\mathcal{A}_2$, and $\mathcal{T}$ be as in the lemma. It clearly suffices to show that $\mathsf{tp}_{\mathcal{A}_1 \cup \mathcal{A}_2,\mathcal{T}}(a) \subseteq \mathsf{tp}_{\mathcal{A}_i,\mathcal{T}}(a)$ for all $a \in \mathsf{Ind}(\mathcal{A}_i)$, $i \in \{1, 2\}$. We show the contrapositive. Thus, assume that $\mathcal{A}_i, \mathcal{T} \not\models A(a)$. We have to show that $\mathcal{A}_1 \cup \mathcal{A}_2, \mathcal{T} \not\models A(a)$. It is well-known [Lutz and Wolter, 2010] that for every $\mathcal{EL}$-TBox $\mathcal{T}$ and ABox $\mathcal{A}$, there is a model $\mathcal{I}$ that is universal in the sense that $a \in A^{\mathcal{I}}$ iff $\mathcal{A}, \mathcal{T} \models A(a)$ for all $a \in \mathsf{Ind}(\mathcal{A})$ and all concept names A. For each $j \in \{1, 2\}$, let $\mathcal{I}_j$ be such a universal model for $\mathcal{T}$ and $\mathcal{A}_j$. We can assume w.l.o.g. that $\Delta^{\mathcal{I}_1} \cap \Delta^{\mathcal{I}_2} = \mathsf{Ind}(\mathcal{A}_1) \cap \mathsf{Ind}(\mathcal{A}_2)$. By assumption and since $\mathsf{tp}_{\mathcal{A}_1,\mathcal{T}}(a) = \mathsf{tp}_{\mathcal{A}_2,\mathcal{T}}(a)$, we must have $a \notin A^{\mathcal{I}_1}$ and $a \notin A^{\mathcal{I}_2}$. Consider the (non-disjoint) union $\mathcal{I}$ of $\mathcal{I}_1$ and $\mathcal{I}_2$. Clearly, $\mathcal{I}$ is a model of $\mathcal{A}_1 \cup \mathcal{A}_2$ and $a \notin A^{\mathcal{I}}$. To show $\mathcal{A}_1 \cup \mathcal{A}_2, \mathcal{T} \not\models A(a)$, it thus remains to prove that $\mathcal{I}$ is a model of $\mathcal{T}$. To do this, we argue that all concept inclusions from $\mathcal{T}$ are satisfied:

- Consider $\exists r.A_1 \sqsubseteq A_2 \in \mathcal{T}$ and $a, b \in \Delta^{\mathcal{I}}$ such that $(a, b) \in r^{\mathcal{I}}$ and $b \in A_1^{\mathcal{I}}$. Then there exist $i, j \in \{1, 2\}$ such that $(a, b) \in r^{\mathcal{I}_i}$ and $b \in A_1^{\mathcal{I}_j}$. If $i = j$, then $a \in A_2^{\mathcal{I}}$, since $\mathcal{I}_i$ is a model of $\mathcal{T}$. Otherwise $b \in \Delta^{\mathcal{I}_1} \cap \Delta^{\mathcal{I}_2} = \mathsf{Ind}(\mathcal{A}_1) \cap \mathsf{Ind}(\mathcal{A}_2)$, so by assumption, $\mathsf{tp}_{\mathcal{A}_1,\mathcal{T}}(b) = \mathsf{tp}_{\mathcal{A}_2,\mathcal{T}}(b)$. It follows that $A_1 \in \mathsf{tp}_{\mathcal{A}_i,\mathcal{T}}(b)$ and thus, $b \in A_1^{\mathcal{I}_i}$. Together with $(a, b) \in r^{\mathcal{I}_i}$ and because $\mathcal{I}_i$ is a model of $\mathcal{T}$, it follows that $a \in A_2^{\mathcal{I}_i} \subseteq A_2^{\mathcal{I}}$. Thus, the inclusion $\exists r.A_1 \sqsubseteq A_2$ is satisfied in $\mathcal{I}$.
- Consider $\top \sqsubseteq A_1 \in \mathcal{T}$ and $a \in \Delta^{\mathcal{I}}$. Then $a \in \Delta^{\mathcal{I}_i}$ for some $i \in \{1, 2\}$. Since $\mathcal{I}_i$ is a model of $\mathcal{T}$, we have $a \in A_1^{\mathcal{I}_i}$, so $a \in A_1^{\mathcal{I}}$ and the inclusion $\top \sqsubseteq A_1$ is satisfied in $\mathcal{I}$.
- Consider $A_1 \sqcap A_2 \sqsubseteq A_3 \in \mathcal{T}$ and $a \in A_1^{\mathcal{I}} \cap A_2^{\mathcal{I}}$. Then there are $i, j \in \{1, 2\}$ such that $a \in A_1^{\mathcal{I}_i}$ and $a \in A_2^{\mathcal{I}_j}$. If $i = j$, then $a \in A_3^{\mathcal{I}}$ follows, since $\mathcal{I}_i$ is a model of $\mathcal{T}$. Otherwise $a \in \Delta^{\mathcal{I}_1} \cap \Delta^{\mathcal{I}_2} = \mathsf{Ind}(\mathcal{A}_1) \cap \mathsf{Ind}(\mathcal{A}_2)$, so by assumption, $\mathsf{tp}_{\mathcal{A}_1,\mathcal{T}}(a) = \mathsf{tp}_{\mathcal{A}_2,\mathcal{T}}(a)$. For sure we have $A_1, A_2 \in \mathsf{tp}_{\mathcal{A}_1,\mathcal{T}}(a)$, so we have $a \in A_1^{\mathcal{I}_1} \cap A_2^{\mathcal{I}_1}$ and since $\mathcal{I}_1$ is a model of $\mathcal{T}$, we conclude $a \in A_3^{\mathcal{I}_1} \subseteq A_3^{\mathcal{I}}$, so the inclusion $A_1 \sqcap A_2 \sqsubseteq A_3$ is satisfied in $\mathcal{I}$.
- Consider $A_1 \sqsubseteq \exists r.A_2 \in \mathcal{T}$ and $a \in A_1^{\mathcal{I}}$. Then $a \in A_1^{\mathcal{I}_i}$ for some $i \in \{1, 2\}$. Since $\mathcal{I}_i$ is a model of $\mathcal{T}$, we have $b \in \Delta^{\mathcal{I}_i}$ and $(a, b) \in r^{\mathcal{I}_i}$, hence also $b \in \Delta^{\mathcal{I}}$ and $(a, b) \in r^{\mathcal{I}}$ and thus, $A_1 \sqsubseteq \exists r.A_2$ is satisfied in $\mathcal{I}$.

□

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

Proof. Let $Q = (\mathcal{T}, \Sigma, A_0(x))$ have the ability to simulate PSA. We show that there is an FO-reduction from PSA to evaluating Q. Let $G = (V, E, S, t)$ be an input to PSA. We construct a Σ-ABox $\mathcal{A}_G$ with a distinguished node a_0 such that $\mathcal{A}_G, \mathcal{T} \models A_0(a_0)$ iff t is accessible in G.

Since Q is able to simulate PSA, there are $\mathcal{A}$, a, b, c, d, t_0, and t_1 satisfying Conditions 1 to 4 from Definition 3. We construct $\mathcal{A}_G$ as follows. Reserve an individual a_v for every $v \in V$. For every $(u, v, w) \in E$, include in $\mathcal{A}_G$ a copy of $\mathcal{A}_\wedge$ and identify a_u, a_v and a_w with c, d and b, respectively. For every $v \in S$, include a copy of $\mathcal{A}_{\mathsf{start}}$ and identify its root with a_v. Finally, include a copy of $\mathcal{A}_{\mathsf{finish}}$ and identify b with a_t. The distinguished node a_0 of $\mathcal{A}_G$ is the root of $\mathcal{A}_{\mathsf{finish}}$. It can be verified that $\mathcal{A}_G$ can be defined from G by a first-order query.

Claim. $\mathcal{A}_G, \mathcal{T} \models A_0(a_0)$ iff t is accessible in G.

First assume that t is accessible in G. Define a sequence $S = S_0 \subseteq S_1 \subseteq \cdots \subseteq V$ by setting

$$S_{i+1} = S_i \cup \{v \in V \mid \text{there is a } (u, w, v) \in E \text{ s.t. } u, w \in S_i\}$$

and let the sequence stabilize at S_n. Clearly, the elements of S_n are exactly the accessible nodes. It can be shown by induction on i that whenever $v \in S_i$, then $\mathcal{A}_G, \mathcal{T} \models t_1(a_v)$. In fact, the induction start follows from $t_1 = \mathsf{tp}_{\mathcal{A},\mathcal{T}}(b)$ and the induction step from Condition 2. (Here we use the fact that in $\mathcal{EL}$, for every tree-shaped ABox $\mathcal{A}$ and every individual b the type $\mathsf{tp}_{\mathcal{A},\mathcal{T}}(b)$ depends only on $\mathcal{A}^b$.) It follows from Conditions 1 and 2 that $\mathcal{A}_{\mathsf{finish}} \cup t_1(b), \mathcal{T} \models A_0(a)$, thus $\mathcal{A}_G, \mathcal{T} \models A_0(a_0)$ as required.

For the other direction, assume that t is not accessible in G. Set

$$\begin{aligned} \mathcal{A}'_G \quad := \quad & \mathcal{A}_G \cup \{t_0(a_v) \mid v \in V \text{ is not accessible}\} \\ & \cup \{t_1(a_v) \mid v \in V \text{ is accessible}\}. \end{aligned}$$

We show that $\mathcal{A}'_G, \mathcal{T} \not\models A_0(a_0)$, which implies $\mathcal{A}_G, \mathcal{T} \not\models A_0(a_0)$.

We have defined $\mathcal{A}'_G$ as an extension of $\mathcal{A}_G$. Alternatively and more suitable for what we want to prove, $\mathcal{A}'_G$ can be obtained by starting with an ABox $\mathcal{A}_0$ that contains only the assertions $t_0(a_v)$ for all inaccessible nodes $v \in V$ as well as $t_1(a_v)$ for all accessible nodes $v \in V$, and then exhaustively applying the following rules in an unspecified order:

- Choose a triple $(u, v, w) \in E$ that has not been chosen before, take a new copy of $\mathcal{A}_\wedge$, rename c to a_u, d to a_v and b to a_w, and add the assertions $t_{\mathsf{acc}(x)}(a_x)$ for $x \in \{u, v, w\}$ where $\mathsf{acc}(x) = 1$ if x is accessible and $\mathsf{acc}(x) = 0$ otherwise. Let this modified copy of $\mathcal{A}_\wedge$ be called $\mathcal{A}_\wedge^{u,v,w}$. Now $\mathcal{A}_{i+1}$ is defined as the union of $\mathcal{A}_i$ and $\mathcal{A}_\wedge^{u,v,w}$.
- Chose a node $v \in S$ that has not been chosen before, introduce a new copy of $\mathcal{A}_{\mathsf{start}}$ and rename b to a_v, add the assertions $t_1(a_v)$. Let this altered copy of $\mathcal{A}_{\mathsf{start}}$ be called $\mathcal{A}_{\mathsf{start}}^v$. Now $\mathcal{A}_{i+1}$ is defined as the union of $\mathcal{A}_i$ and $\mathcal{A}_{\mathsf{start}}^v$.
- Introduce a new copy of $\mathcal{A}_{\mathsf{finish}}$, rename b to a_t and add the assertions $t_0(a_t)$. Let this altered copy of $\mathcal{A}_{\mathsf{finish}}$ be called $\mathcal{A}_{\mathsf{finish}}^t$. Now $\mathcal{A}_{i+1}$ is defined as the union of $\mathcal{A}_i$ and $\mathcal{A}_{\mathsf{finish}}^t$.

Clearly, rule application terminates after finitely many step and results in the ABox $\mathcal{A}'_G$.

Claim. $\mathsf{tp}_{\mathcal{A}_i,\mathcal{T}}(u) = t_0$ if $u \in V$ is inaccessible and $\mathsf{tp}_{\mathcal{A}_i,\mathcal{T}}(v) = t_1$ if $v \in V$ is accessible, for all $i > 0$.

The proof is by induction on i. For $i = 0$, the statement is clear since t_0 and t_1 are $\mathcal{T}$-types. Now assume the statement is true for some i and consider $\mathcal{A}_{i+1}$. If $\mathcal{A}_{i+1}$ was obtained by the first rule, it can be verified using Conditions 2 and 4 from Definition 3 that $\mathsf{tp}_{\mathcal{A}_\wedge^{u,v,w},\mathcal{T}}(x) = t_{\mathsf{acc}(x)}$ for all $x \in \{u, v, w\}$. So with Lemma 29 and since $\mathcal{A}_i$ and $\mathcal{A}_\wedge^{u,v,w}$ share only the individuals u, v, w, the statement follows. If $\mathcal{A}_{i+1}$ was obtained by the second rule it follows from Condition 2 that $\mathsf{tp}_{\mathcal{A}_{\mathsf{start}}^v,\mathcal{T}}(a_v) = t_1$ and with Lemma 29, the statement follows. If $\mathcal{A}_{i+1}$ was obtained by the third rule, it can be verified that $\mathsf{tp}_{\mathcal{A}_{\mathsf{finish}}^t,\mathcal{T}}(a_t) = t_0$ and with Lemma 29, the statement follows. This finishes the proof of the claim.

The claim yields $\mathsf{tp}_{\mathcal{A}'_G,\mathcal{T}}(a_t) = t_0$, so by Condition 3 from Definition 3 it follows that $\mathcal{A}'_G, \mathcal{T} \not\models A_0(a_0)$, as required. □

Our next aim is to show that our two characterizations are equivalent. We first establish a purely combinatorial lemma.

Lemma 30. *Let $k \geq 0$, $n \geq 1$. Every full binary tree of depth $n \cdot k$ whose nodes are colored with n different colors has the full monochromatic binary tree of depth k as a minor.*

Proof. Every full binary tree T whose nodes are colored with n different colors will be associated with the tuple $(m_1, \ldots, m_n)$, where m_i is the smallest positive integer such that T does not have the full binary tree of depth m_i of color i as a minor and $m_i = 0$ if T doesn't contain any nodes of color i. We will prove the following statement:

Claim. Let T be a full binary tree of depth k whose nodes are colored with n different colors. Then $\sum_{i=1}^n m_i \geq k + 1$.

We proof the claim by induction on k. For $k = 0$ there is only one node with a color i_0. Then $m_{i_0} = 1$ and $m_i = 0$ for all $i \neq i_0$. Hence, $\sum_{i=1}^n m_i = 1 \geq 1 = k + 1$.

Now assume the claim holds for a fixed k and consider a tree T of depth $k + 1$. Let the root be called a and the subtrees whose roots are the children of a be T_1 and T_2. Let $(m_1^j, \ldots, m_n^j)$ be the tuple associated with T_j for $j \in \{1, 2\}$. The tuple $(m_1, \ldots, m_n)$ associated with T can be computed in the following way: Let i_0 be the color of a. For every $i \neq i_0$ we have $m_i = \max(m_i^1, m_i^2)$. If $m_{i_0}^1 = m_{i_0}^2$, then $m_{i_0} = m_{i_0}^1 + 1$. Otherwise $m_{i_0} = \max(m_i^1, m_i^2)$. We distinguish two cases:

1. There exists a color j such that $m_j^1 \neq m_j^2$. W.l.o.g. let $m_j^1 < m_j^2$. Then $\sum_{i=1}^n m_i = \sum_{i=1}^n \max(m_i^1, m_i^2) = m_j^2 + \sum_{i \neq j} \max(m_i^1, m_i^2) \geq 1 + m_j^1 + \sum_{i \neq j} \max(m_i^1, m_i^2) \geq 1 + m_j^1 + \sum_{i \neq j} m_i^1 \geq 1 + \sum_{i=1}^n m_i^1 \geq 1 + (k + 1)$.
2. We have $m_i^1 = m_i^2$ for all colors. Then we also have $m_{i_0}^1 = m_{i_0}^2$, so $m_{i_0} = 1 + m_{i_0}^1$. Again, it follows $\sum_{i=1}^n m_i \geq 1 + (k + 1)$.

With the claim proven, the lemma follows easily: Let T be a full binary tree of depth $n \cdot k$ whose nodes are colored with n different colors. Then $\sum_{i=1}^n m_i \geq n \cdot k + 1$. Assume there is no full monochromatic binary tree of depth k as a minor in T, then $m_i \leq k$ for all colors i. This yields $\sum_{i=1}^n m_i \leq k \cdot n$, a contradiction. □

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

Proof. 1 ⇒ 2: Let $k \geq 1$ and $\mathcal{A}, a, b, c, d, t_0$ and t_1 be as in Definition 3. Let $S = \{w \in \{0, 1\}^* \mid |w| \leq k - 1\}$. We will construct an ABox $\mathcal{A}_k$ built up from the following set of ABoxes:

- One copy of $\mathcal{A}_{\mathsf{finish}}$;
- For every $w \in S$, one copy $\mathcal{A}_{\wedge,w}$ of $\mathcal{A}_\wedge$;

- For every $w \in \{0,1\}^k$, one copy $\mathcal{A}_{\mathsf{start},w}$ of $\mathcal{A}_{\mathsf{start}}$.

We identify the individual b of $\mathcal{A}_{\mathsf{finish}}$ with the individual b of $\mathcal{A}_{\wedge,\varepsilon}$. For every $w0 \in S$, we identify the individual b of $\mathcal{A}_{\wedge,w0}$ with the individual c of $\mathcal{A}_{\wedge,w}$ and for every $w1 \in S$, we identify the individual b of $\mathcal{A}_{\wedge,w1}$ with the individual d of $\mathcal{A}_{\wedge,w}$. Finally, for every $w0 \in \{0,1\}^k$, we identify the individual b of $\mathcal{A}_{\mathsf{start},w0}$ with the individual c of $\mathcal{A}_{\wedge,w}$ and for every $w1 \in \{0,1\}^k$, we identify the individual b of $\mathcal{A}_{\mathsf{start},w1}$ with the individual d of $\mathcal{A}_{\wedge,w}$. The resulting ABox is $\mathcal{A}_k$. The root of $\mathcal{A}_k$ is a.

$\mathcal{A}_k$ has the full binary tree of depth k as a minor (induced by the set of roots of all $\mathcal{A}_{\wedge,w}$ and $\mathcal{A}_{\mathsf{start},w}$). From conditions 1 and 2 from Definition 3 follows that $\mathcal{A}_k, \mathcal{T} \models A_0(a)$.

The constructed ABox $\mathcal{A}_k$ does not have to be minimal, but we can minimize it without destroying the full binary tree minor: As long as $\mathcal{A}_k$ is not minimal, remove any concept or role assertion from $\mathcal{A}_k$ such that $\mathcal{A}_k, \mathcal{T} \models A_0(a)$ still holds. We argue that a role assertion connecting two individuals which lie on the same path from a to a root of a $\mathcal{A}_{\mathsf{start},w}$ can never be removed: If the removed role assertion lies in $\mathcal{A}_{\wedge,w}$ on the path from its b to its c, then $\mathsf{tp}_{\mathcal{A}_k,\mathcal{T}}(b) \subseteq \mathsf{tp}_{\mathcal{A}_c^b \cup t_0(c),\mathcal{T}}(b) = t_0$. By iteratively using conditions 3 and 4 from Definition 3, it follows that $\mathcal{A}_k, \mathcal{T} \not\models A_0(a)$. If the removed role assertion lies in $\mathcal{A}_{\wedge,w}$ on the path from its b to its d, the proof is analogous. If the removed role assertion lies in $\mathcal{A}_{\mathsf{finish}}$ on the path from the root to its b, it follows that $\mathsf{tp}_{\mathcal{A}_k,\mathcal{T}}(a) \subseteq \mathsf{tp}_{\mathcal{A}_b \cup t_0(b),\mathcal{T}}(a) \not\ni A_0$.

Since no role assertion on a path from a to a root of an instance of $\mathcal{A}_{\mathsf{start},w}$ will ever be removed, the previously identified minor in form of the full binary tree of depth k still remains in the minimized $\mathcal{A}_k$.

$2 \Rightarrow 1$: Let tp denote the set of all $\mathcal{T}$-types and $m = |\mathsf{tp}|$. Choose $k = m \cdot 2^m \cdot (2m^m + 1)$. Consider a minimal tree-shaped Σ-ABox $\mathcal{A}$ that satisfies $\mathcal{A}, \mathcal{T} \models A_0(a)$, a the root of $\mathcal{A}$, and has the full binary tree of depth k as a minor. We color every $b \in \mathsf{Ind}(\mathcal{A})$ with the color $(\mathsf{tp}_{\mathcal{A},\mathcal{T}}(b), S_b)$ where $S_b \subseteq \mathsf{tp}$ such that $t \in S_b$ whenever $\mathcal{A}_b \cup t(b), \mathcal{T} \models A_0(a)$. There are at most $m \cdot 2^m$ distinct colors, so from Lemma 30 we know that $\mathcal{A}$ has as a minor a monochromatic full binary tree T of depth $2m^m + 1$. Let b be a child of the root of T (to make sure that b is not the root of $\mathcal{A}$) and $T' \subseteq T$ the subtree of T rooted at b, so T' is a full binary tree of depth $2m^m$. We color every $c \in T'$ with the function $f_c : \mathsf{tp} \to \mathsf{tp}$ that is defined by $f_c(t) = \mathsf{tp}_{\mathcal{A}_c \cup t(c),\mathcal{T}}(b)$. There are at most m^m such functions, so again by Lemma 30, there will be the monochromatic binary tree of depth 2 as a minor, especially we have two independent nodes c and d in T that are colored with the same function and that are non-leaves. We will show that $\mathcal{A}$ with the distinguished nodes b, c, d witnesses that Q has the ability to simulate PSA. Condition 1 is true by choice of $\mathcal{A}$. Condition 2 is fulfilled, because b, c and d were colored with the same color by the first coloring (we set $t_1 := \mathsf{tp}_{\mathcal{A},\mathcal{T}}(b)$).

To find t_0, we define a sequence t'_i of $\mathcal{T}$-types where $t'_0 = \emptyset$ and $t'_{i+1} = \mathsf{tp}_{\mathcal{A}_c \cup t'_i(c),\mathcal{T}}(b)$. It follows by induction on i that $t'_i \subseteq t'_{i+1}$ for all i. Let t_0 be the limit of the t'_i, so since c and d were colored with the same function $f_c = f_d$, condition 4 holds. It thus remains to argue that condition 3 holds. We show by induction on i that $\mathcal{A}_c \cup t'_i(c), \mathcal{T} \not\models A_0(a)$. It is clear that $\mathcal{A}_c \cup t'_0(c), \mathcal{T} \not\models A_0(a)$ since c is not a leaf and $\mathcal{A}$ is minimal for $\mathcal{A}, \mathcal{T} \models A_0(a)$. Now assume $\mathcal{A}_c \cup t'_i(c), \mathcal{T} \not\models A_0(a)$ for some i, so we have $\mathcal{A}_c \cup t'_{i+1}(b), \mathcal{T} \not\models A_0(a)$. And since in the first coloring, $S_b = S_c$ has been assured, $\mathcal{A}_c \cup t'_{i+1}(c), \mathcal{T} \not\models A_0(a)$ follows, which completes the induction. So especially we have $\mathcal{A}_c \cup t_0(c), \mathcal{T} \not\models A_0(a)$ and again using $S_b = S_c$, condition 3 follows. □

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 $\mathcal{A} \models \Pi_{Q,k}(a_0)$.*

Proof. Let $\mathcal{A}$ be a tree-shaped ABox with root a_0 that does not have the full binary tree of depth k as a minor and $\mathcal{A}, \mathcal{T} \models A_0(a_0)$. For every $a \in \mathsf{Ind}(\mathcal{A})$, let $\ell(a)$ be the largest integer such that the ABox $\mathcal{A}^a$ contains the full binary tree of depth $\ell(a)$ as a minor. It can be seen that every leaf is labelled by 0 and every non-leaf is labelled by

- the maximum label of its children, if this maximum occurs precisely once among the children and
- the maximum label of its children plus one, otherwise.

We recursively define a sequence in $\mathsf{Ind}(\mathcal{A})$ that will be used to guide the derivation of $\Pi_{Q,k}$ on $\mathcal{A}$, by defining it for all subtrees $\mathcal{A}^a$. If a is a leaf, then $\mathsf{seq}(\mathcal{A}^a) = (a)$. Otherwise, let $\mathcal{A}_1, \ldots, \mathcal{A}_n$ be the subtrees rooted at the children of a and w.l.o.g. let the root of $\mathcal{A}_1$ have the maximum label among the children of a. Then $\mathsf{seq}(\mathcal{A}^a) = \mathsf{seq}(\mathcal{A}_1) \cdot a \cdot \mathsf{seq}(\mathcal{A}_2) \cdot a \cdot \ldots \cdot \mathsf{seq}(\mathcal{A}_n) \cdot a$.

It can be seen that the first individual of such a sequence is a leaf. It can be proved by induction on the depth of $\mathcal{A}^a$ that every individual in $\mathsf{seq}(\mathcal{A}^a)$ besides the first one is either a leaf below the previous individual or the parent of the previous individual.

Let $\mathsf{seq}(\mathcal{A}) = a_1 \ldots a_m$. This sequence gives rise to a sequence of tuples $B_1, \ldots, B_m$ of individuals that will occur together in an IDB fact. We set $B_1 = (a_1)$. If a_{i+1} is a leaf, then $B_{i+1} = B_i \cdot a_{i+1}$. Otherwise, let B'_{i+1} be obtained by replacing the last element of B_i (which is a_i) with a_{i+1}. If the second last element is also a_{i+1}, then B_{i+1} is obtained by removing the last element from B'_{i+1}. Otherwise, B_{i+1} is defined to be B'_{i+1}. For every $1 \leq i \leq m$, let $B_i = (b_{i,1}, \ldots, b_{i,n_i})$.

We prove by induction on i, that for every B_i, we have $\ell(b_{i,1}) > \ell(b_{i,2}) > \ldots > \ell(b_{i,n_i})$ and all these individuals lie on the same path from the root to a leaf. For B_1, there is nothing to show. Assume the statement is true for B_{i-1} and consider B_i. If a_i is a leaf, then it lies somewhere below a_{i-1}. If a_{i-1} had only one child c, then $\mathsf{seq}(\mathcal{A}^{a_{i-1}}) = \mathsf{seq}(\mathcal{A}^c) \cdot a_{i-1}$ and a_i would be the parent of a_{i-1}. So a_{i-1} has at least two children and hence, $\ell(a_{i-1}) > 0$ and $\ell(a_i) = 0$. If a_i is the parent of a_{i-1}, we have to argue that $\ell(b_{i,n_i-1}) > \ell(b_{i,n_i}) = \ell(a_i)$. This follows from the fact that a_i lies below b_{i,n_i-1} and that the subtree rooted at the child of b_{i,n_i-1} with the highest label has been traversed first, so a_i does not lie in this first subtree. So a_i is (a descendant of) a child of b_{i,n_i-1} that has a label smaller than b_{i,n_i-1}.

Since the largest label of an individual in $\mathcal{A}$ is $k-1$, it follows that the tuples B_i have length at most k.

Next, we will describe the derivation that leads to $\mathsf{goal}(a_0)$. For $a \in \mathsf{Ind}(\mathcal{A})$ and any subset C of the children of a, let $\mathcal{A}^a_C$ denote the subset of $\mathcal{A}$ containing only assertions involving a or an individual from $\bigcup_{c\in C} \mathcal{A}^c$.

Claim. For every $i \in \{1,\dots,m\}$, we have $\Pi_{Q,k}, \mathcal{A} \models P_{t_1,\dots,t_n}(B_i)$, where $t_j = \mathsf{tp}_{\mathcal{A}^{b_{i,j}}_{C_j},\mathcal{T}}(b_{i,j})$ and C_j is the set of all children of $b_{i,j}$ that have appeared before a_i in the sequence.

We prove the claim by induction on i. Since a_1 is a leaf and no other individuals have appeared before a_1, the claimed IDB fact is $P_{t_1}(a_1)$, where $t_1 = \mathsf{tp}_{\mathcal{A}^{a_1},\mathcal{T}}(a_1)$. We obtain this fact by using the start rule for a_1.

Assuming the claim holds for a specific i, we prove the claim for $i+1$. For the first case, assume that a_{i+1} is a leaf. We use the extension rule for $P_{t_1,\dots,t_n}(B_i)$ and a_{i+1} to obtain $P_{t_1,\dots,t_{n+1}}(B_{i+1})$, with t_{n+1} as required.

Now consider the case that a_{i+1} is not a leaf. Then a_{i+1} is the parent of a_i. We apply the step rule to obtain $P_{t_1,\dots,t'_n}(B'_{i+1})$ with $t'_n = \mathsf{tp}_{\mathcal{A}^{a_{i+1}}_{\{a_i\}},\mathcal{T}}(a_{i+1})$. If a_{i+1} has not appeared before, we have $P_{t_1,\dots,t'_n}(B'_{i+1}) = P_{t_1,\dots,t'_n}(B_{i+1})$, the claimed fact. If a_{i+1} has appeared before in the sequence, use the consolidation rule to obtain $P_{t_1,\dots,t_{n-2},\mathsf{cl}(t_{n-1}\cup t'_n)}(B_{i+1})$, where $\mathsf{cl}(t_{n-1} \cup t'_n) = \mathsf{tp}_{\mathcal{A}^{a_{i+1}}_{C\cup\{a_i\}},\mathcal{T}}(a_{i+1})$ and $C \cup \{a_i\}$ is now the set of all children of a_{i+1} that have appeared before a_{i+1} in the sequence. Thus, the claim follows.

With the claim proven, consider the claim for the last element of $\mathsf{seq}(\mathcal{A})$, which is the root a_0. All children of a_0 have appeared before. The claim gives $\Pi_{Q,k}, \mathcal{A} \models P_t(a_0)$, where $t = \mathsf{tp}_{\mathcal{A},\mathcal{T}}(a) \ni A_0$, so the goal rule can be applied to derive $\mathsf{goal}(a_0)$. □

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

Proof. Since $\Pi_{Q,k}$ is sound (independently of the value chosen for k), it remains to show completeness. Thus let $\mathcal{A}$ be a Σ-ABox with $\mathcal{A}, \mathcal{T} \models A_0(a)$. Let $\mathcal{A}^u$ be the unraveling of $\mathcal{A}$ at a. Lemma 27 yields $\mathcal{A}^u, \mathcal{T} \models A_0(a)$. Since the branching limit of Q is $k-1$ and by compactness, we find a finite tree-shaped $\mathcal{A}' \subseteq \mathcal{A}^u$ such that $\mathcal{A}', \mathcal{T} \models A_0(a)$ and $\mathcal{A}'$ does not have the full binary tree of depth k as a minor. Lemma 9 yields $\mathcal{A}' \models \Pi_{Q,k}$, thus $\mathcal{A}^u \models \Pi_{Q,k}$. Since there is a homomorphism from $\mathcal{A}^u$ to $\mathcal{A}$ and answers to Datalog programs are preserved under homomorphisms, this implies $\mathcal{A} \models \Pi_{Q,k}$. □

Lemma 12. $(d-1)^k(n-k)^k \le \ell^k_d(n) \le (k+1)(d-1)^k n^k$ *for all $d, k \ge 0$ and $n \ge 2k$.*

Proof. We aim to show that for all $d, k \ge 0$ and $n \ge 2k$,

$$\ell^k_d(n) = \sum_{i=0}^{k} (d-1)^i \binom{n}{i} \qquad (*)$$

From $(*)$, the lower bound stated in the lemma is obtained by considering only the summand for $i = k$ and the upper bound is obtained by replacing every summand with the largest summand, which is the summand for $i = k$ if $n \ge 2k$.

Towards proving $(*)$, we first observe that for all $n \ge 1$ and $k \ge 1$:

$$\ell^k_d(n) = \ell^k_d(n-1) + (d-1)\ell^{k-1}_d(n-1) \qquad (**)$$

Let T be a tree with degree d and depth n that does not contain the full binary tree of depth $k+1$ as a minor and that has the largest possible number of leaves. It can easily be seen that the root of T has degree d and that T contains the full binary tree of depth k as a minor. Consider the subtrees $T_1, \dots, T_d$ whose roots are the children of the root of T. There must be one of them that also has the full binary tree of depth k as a minor and all of them must have the full binary tree of depth $k-1$ as a minor, otherwise T would not have the maximum number of leaves. Moreover, there cannot be two subtrees that both have the full binary tree of depth k as a minor, since then T would have a minor of depth $k+1$. Since the number of leaves of T is the sum of the leaves of all T_j, $(**)$ follows.

Now we prove $(*)$ by induction on n. First observe that $\ell^k_d(0) = \ell^0_d(n) = 1$ for all d, k, n, thus $(*)$ holds for all cases where $k = 0$ or $n = 0$. From now on, let $k \ge 1$ and $n \ge 1$ and assume that $(*)$ holds for $\ell^k_d(n)$ and for $\ell^{k-1}_d(n)$. We show that it also holds for $\ell^k_d(n+1)$:

$$\begin{aligned}
\ell^k_d(n+1) &= \ell^k_d(n) + (d-1)\cdot \ell^{k-1}_d(n) \\
&= \sum_{i=0}^{k} (d-1)^i \binom{n}{i} + (d-1)\sum_{i=0}^{k-1} (d-1)^i \binom{n}{i} \\
&= \sum_{i=0}^{k} (d-1)^i \binom{n}{i} + \sum_{i=1}^{k} (d-1)^i \binom{n}{i-1} \\
&= 1 + \sum_{i=1}^{k} (d-1)^i \binom{n+1}{i} \\
&= \sum_{i=0}^{k} (d-1)^i \binom{n+1}{i}
\end{aligned}$$

□

A *derivation of* $\Pi(a)$ *in* $\mathcal{A}$ is a labelled directed tree (V, E, ℓ) where

1. $\ell(x_0) = \mathsf{goal}(a)$ for x_0 the root node;
2. for each $x \in V$ with children $y_1, \dots, y_k$, $k > 0$, there is a rule $S(\mathbf{y}) \leftarrow q(\mathbf{x})$ in Π and a substitution σ of variables by individuals from $\mathcal{A}$ such that $\ell(x) = S(\sigma\mathbf{y})$ and $\ell(y_1), \dots, \ell(y_k)$ are exactly the facts in $q(\sigma\mathbf{x})$;
3. if x is a leaf, then $\ell(x) \in \mathcal{A}$.

Note that all leaves are labelled with Σ-assertions from $\mathcal{A}$ and all inner nodes with Σ_I-assertions, that is, assertions of the form $P(\mathbf{a})$ with $P \in \Sigma_I$ and $\mathbf{a}$ a tuple of individuals from $\mathsf{Ind}(\mathcal{A})$. It is well known that $\mathcal{A} \models \Pi(a)$ iff there is a derivation of $\Pi(a)$ in $\mathcal{A}$

We associate with each derivation $D = (V, E, \ell)$ of $\Pi(a)$ in $\mathcal{A}$ an instance $\mathcal{A}_D$. In fact, we first associate an instance $\mathcal{A}_x$ with every $x \in V$ and then set $\mathcal{A}_D := \mathcal{A}_{x_0}$ for x_0 the root of D. If $x \in V$ is a leaf, then $\ell(x) \in \mathcal{A}$ and we set $\mathcal{A}_x = \{\ell(x)\}$. If $x \in V$ has children $y_1, \dots, y_k$, $k > 0$, such that $y_1, \dots, y_\ell$ are labelled with Σ_I-assertions and $y_{\ell+1}, \dots, y_k$ with Σ-assertions, then $\mathcal{A}_x$ is obtained by starting with the facts from $\ell(y_{\ell+1}), \dots, \ell(y_k)$ and then adding a copy of $\mathcal{A}_{y_i}$, for $1 \leq i \leq \ell$, in which all individuals except those in $\ell(x)$ are substituted with fresh individuals.

The following lemma, stated also in the main part of the paper, is straightforward to verify.

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.*

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

Proof. Let $Q = (\mathcal{T}, \Sigma, A_0(x))$ be unboundedly branching. Assume to the contrary of what we have to show that Q is rewritable into a linear Datalog program Π. Let d be the diameter of Π. To establish a contradiction, choose $k_0 > 0$ very large (we will make this precise later) and let $\mathcal{A}$ be a minimal tree-shaped Σ-ABox with root a_0 such that $\mathcal{A} \models Q(a_0)$ and $\mathcal{A}$ has the full binary tree T of depth k_0 as a minor. We can assume w.l.o.g. that there is a constant ℓ (which is independent of k_0) such that the depth of $\mathcal{A}$ is bounded by $\ell \cdot k_0$: since Q is unboundedly branching, by Proposition 6 we must find tree-shaped Σ-ABoxes $\mathcal{A}_{\mathsf{start}}$, $\mathcal{A}_\wedge$ and $\mathcal{A}_{\mathsf{finish}}$ as in Definition 3 (ability to simulate PSA) from which we can assemble the desired ABox $\mathcal{A}$ as in the construction from the proof of the "$\Rightarrow$" direction of Proposition 6. As ℓ, we can then use the sum of the depths of $\mathcal{A}_{\mathsf{start}}$, $\mathcal{A}_\wedge$ and $\mathcal{A}_{\mathsf{finish}}$.

We have $\mathcal{A} \models \Pi(a_0)$ and thus there is a derivation D of $\Pi(a_0)$ in $\mathcal{A}$. Consider the ABox $\mathcal{A}_D$. By Lemma 13, we have the following:

1. $\mathcal{A}_D \models \Pi(a_0)$;
2. there is a homomorphism h from $\mathcal{A}_D$ to $\mathcal{A}$ with $h(a_0) = a_0$;
3. $\mathcal{A}_D$ has pathwidth at most d.

We manipulate $\mathcal{A}_D$ as follows

- Restrict the degree to $|\mathcal{T}|$ by taking a subset according to Lemma 28.
- Remove all assertions that involve an individual a which is not reachable from a_0 along role edges, that is, for which there are no assertions $r_1(a_0, a_1), \dots, r_n(a_{n-1}, a_n)$ with $a_n = a$.

We use $\mathcal{B}$ to denote the resulting ABox. It can be verified that Conditions 1 to 3 still hold when $\mathcal{A}_D$ is replaced with $\mathcal{B}$. In particular, this is true for Condition 1 since $\mathcal{A}_D \models \Pi(a_0)$ iff $\mathcal{A}_D, \mathcal{T} \models A_0(a_0)$ iff $\mathcal{B}, \mathcal{T} \models A_0(a_0)$ iff $\mathcal{B} \models \Pi(a_0)$. The second equivalence is easy to establish by showing how a model witnessing $\mathcal{B}, \mathcal{T} \not\models A_0(a_0)$ can be transformed into a model that witnesses $\mathcal{A}_D, \mathcal{T} \not\models A_0(a_0)$.

Choose a homomorphism h from $\mathcal{B}$ to $\mathcal{A}$ with $h(a_0) = a_0$. Then h must be surjective since otherwise, the restriction $\mathcal{A}^-$ of $\mathcal{A}$ to the individuals in the range of h would satisfy $\mathcal{A}^-, \mathcal{T} \models A_0(a_0)$, contradicting the minimality of $\mathcal{A}$. Since $\mathcal{A}$ contains the full binary tree of depth k_0 as a minor, we find in $\mathcal{A}$ distinct leaves $a_1, \dots, a_{2^{k_0}}$. For each a_i, choose a b_i with $h(b_i) = a_i$. Clearly, all individuals $b_1, \dots, b_{2^{k_0}}$ must be distinct, since $a_1, \dots, a_{2^{k_0}}$ are. Moreover, $\mathcal{B}$ contains no assertion of the form $r(b_i, c)$, for any b_i.

By construction, $\mathcal{B}$ is connected. By Condition 2, and since $\mathcal{A}$ is tree-shaped, $\mathcal{B}$ must have the form of a DAG (a directed acyclic graph), meaning that the directed graph $G_{\mathcal{A}}$ is a DAG. We proceed to exhaustively remove assertions from $\mathcal{B}$ as follows: whenever $r(c_1, c), r(c_2, c) \in \mathcal{B}$ with $c_1 \neq c_2$, then choose and remove one of these two edges. Using the fact that every individual in $\mathcal{B}$ is reachable from a_0, it can be proved by induction on the number of edge removals that the obtained ABoxes

(i) remain connected and

(ii) contain the same individuals as $\mathcal{B}$, that is, edge removal never results in the removal of an individual.

Point (i) and the fact that we start from a DAG-shaped ABox means that the ABox $\mathcal{B}_t$ ultimately obtained by this manipulation is tree-shaped. By construction of $\mathcal{B}_t$, h is still a homomorphism from $\mathcal{B}_t$ to $\mathcal{A}$, $\mathcal{B}_t$ has pathwidth at most d, and the individuals $b_1, \dots, b_{2^{k_0}}$ are leaves in $\mathcal{B}_t$ (and thus $\mathcal{B}_t$ has at least 2^{k_0} leaves). From the former, it follows that the depth of $\mathcal{B}_t$ is at most $\ell \cdot k_0$.

We now show that it is possible to choose a value for k_0 such that, because $\mathcal{B}_t$ has 2^{k_0} leaves, it must contain the full binary tree of a certain depth k as a minor, and consequently has pathwidth exceeding d, which contradicts the fact that $\mathcal{B}_t$ has pathwidth at most d. Set $k = 2d + 3$. By Lemma 12, $\mathcal{B}_t$ contains the full binary tree of depth k as a minor, unless the number of leaves of $\mathcal{B}_t$ is bounded by $k(|T| - 1)^{k-1}(\ell \cdot k_0)^{k-1} = c \cdot k_0^{k-1}$ where c is a constant that is independent of k_0. So if we choose k_0 large enough so that $2^{k_0} > c \cdot k_0^{k-1}$, then the existence of a full binary tree of depth k in $\mathcal{B}_t$ is guaranteed. But it is well-known that the full binary tree of depth k has pathwidth $\lceil k/2 \rceil$ [Scheffler, 1989] and consequently every tree that has this tree as a minor has at least the same pathwidth. It follows that the pathwidth of $\mathcal{B}_t$ is at least $d + 1$, the desired contradiction. □

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

Proof. By Proposition 7, it suffices to show that the Q_k are boundedly branching. Let $\mathcal{A}$ be a tree-shaped ABox that derives Q_k at its root and is minimal with that property. We show that $\mathcal{A}$ does not have the full binary tree of depth $k + 1$ as a minor.

We start with an analysis of the sets $\mathsf{tp}_{\mathcal{A},\mathcal{T}_k}(a)$, $a \in \mathsf{Ind}(\mathcal{A})$, and of the structure of $\mathcal{A}$. Since $\top \sqsubseteq A_0 \in \mathcal{T}_k$, none of the sets $\mathsf{tp}_{\mathcal{A},\mathcal{T}_k}(a)$ is empty. It is easy to verify that $\mathcal{T}_k \models A_i \sqsubseteq A_{i-1}$ and $\mathcal{T}_k \models B_{x,i} \sqsubseteq B_{x,i-1}$ for $1 \leq i \leq k$ and $x \in$

$\{r, s, t, u\}$, so $\mathsf{tp}_{\mathcal{A},\mathcal{T}_k}(a)$ is thus of the form $\{A_0, \ldots, A_i\} \cup B$, $i \geq 0$ and $B \subseteq \{B_{x,j} \mid x \in \{r, s, t, u\} \text{ and } 0 \leq j \leq k\}$. We say that a *has type* i if i is the largest integer such that $A_i \in \mathsf{tp}_{\mathcal{A},\mathcal{T}_k}(a)$ and that a *has* x*-type* j_x if j_x is the largest integer such that $B_{x,j_x} \in \mathsf{tp}_{\mathcal{A},\mathcal{T}_k}(a)$. We argue that every individual in $\mathcal{A}$ has no more than one outgoing edge of the same role. Assume there would be distinct individuals a, b, c and assertions $x(a, b), x(a, c) \in \mathcal{A}$ for some $x \in \{r, s, t, u\}$. Let b have type j and x-type ℓ and c have type m and x-type n, where $0 \leq j, \ell, m, n \leq k$. This will derive $B_{x,j}$, $B_{x,\ell}$, $B_{x,m}$ and $B_{x,n}$ at a, but since $\mathcal{T}_k \models B_{x,i} \sqsubseteq B_{x,i-1}$ for $1 \leq i \leq k$, these four concept names are already implied by $B_{x,\mathsf{max}(j,\ell,m,n)}$, so one of the individuals b, c can be removed without altering the result of the query. We will refer to non-leaves by the combination of role names of their outgoing edges, e.g. an rs-node is an individual that has one outgoing r-edge, one outgoing s-edge and no other outgoing edges. Using minimality of $\mathcal{A}$, it can be argued that for every x-node a with $x \in \{r, s, t, u\}$, we have some $x(b, a) \in \mathcal{A}$ and it follows that a path from one branching point to the next is always a chain of the same role. It can be seen that every individual with degree greater than one is either an rs-node or a tu-node. All other combinations do not appear due to the minimality of $\mathcal{A}$. As an example, assume there is an rst-node a. Then there will $B_{r,j}, B_{s,\ell}, B_{t,m}$ derived at a, with j, ℓ, m being maximal. If a is the root of $\mathcal{A}$, then the t-edge could be removed. If a is not the root, it must be connected to its parent by a t-edge, since otherwise, the t-edge below a could be removed. If $m \leq \mathsf{min}(j, \ell)$, the t-edge below a could be removed as well. So the final case is that $m > \mathsf{min}(j, \ell)$, but then both the r-edge and the s-edge could be removed. In any case, $\mathcal{A}$ would not be minimal, so it does not contain an rst-edge. All other combinations of roles can be argued similarly, so $\mathcal{A}$ has no individuals of outdegree three or higher.

For all $a \in \mathsf{Ind}(\mathcal{A})$, let $\mathsf{tdep}(a)$ denote the maximum depth of a full binary tree that can be found as a minor in the subtree of $\mathcal{A}$ rooted at a.

Claim. a is of type i iff $\mathsf{tdep}(a) = i$ for all $a \in \mathsf{Ind}(\mathcal{A})$ that are leaves or of degree two.

We prove the claim by induction on the number n of leaves of the subtree rooted at a, starting with $n = 1$. For the induction start consider a node a that has one leaf below it. If a would be of degree two, it would have more leaves below, so a is a leaf itself and the statement follows easily. Now let $n > 1$ and a an individual of degree two with n leaves below it. The two outgoing paths from a both either end in a leaf or in another node of degree two. In any case, the induction hypothesis applies for the two endpoints of the paths. Let b and c be the endpoints of the two paths and b have type i and c have type j, so $\mathsf{tdep}(b) = i$ and $\mathsf{tdep}(c) = j$. It follows that $\mathsf{tdep}(a) = i + 1$ if $i = j$ and $\mathsf{tdep}(a) = \mathsf{max}(i, j)$ otherwise. Using the minimality of $\mathcal{A}$ we can see that in the first case, a must be a rs-node and has type $i + 1$. In the second case, also using minimality, a must be a tu-node, i and j have to differ by one with the u path leading to the node with higher type, and it follows that a has type $\mathsf{max}(i, j)$.

From the claim, it immediately follows that Q_k is boundedly branching: In every minimal tree-shaped minimal ABox $\mathcal{A}$ that derives Q_k at the root, the root is either of degree two or is a leaf, since nodes of degree one always have type 0 and $k \geq 1$. So the claim says that k is the largest integer such that $\mathcal{A}$ contains the full binary tree of depth k as a minor. Thus, Q_k is boundedly branching. □

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 $\mathfrak{C}_i$.*

Proof. Let $i \geq 0$. For $n \geq k \geq 1$, we explicitly construct tree-shaped ABoxes $\mathcal{A}_k^n \in \mathfrak{C}_i$ that are minimal for Q_k, have depth at most $n(i + 1)$ and that have a large number of leaves. If $n = k = 1$, then $\mathcal{A}_k^n$ has only three individuals, the root being an rs-node. If $n = k > 1$, then take the disjoint union of two copies of $\mathcal{A}_{k-1}^{n-1}$, introduce a new rs-node as the root, with an outgoing r-path of length $i + 1$ leading to the root of the first copy and an outgoing s-path of length $i + 1$ leading to the root of the second copy. If $n > k = 1$, then $\mathcal{A}_k^n$ consists of a root that is a tu-node with an outgoing t-edge ending in a leaf and an outgoing u-path of length $i + 1$ leading to the root of a copy of $\mathcal{A}_k^{n-1}$. Finally, for $n > k > 1$, take the disjoint union of $\mathcal{A}_{k-1}^{n-1}$ and $\mathcal{A}_k^{n-1}$ and introduce a new tu-node as the root, with an outgoing t-path of length $i + 1$ pointing to the root of $\mathcal{A}_{k-1}^{n-1}$ and an outgoing u-path of length $i + 1$ pointing to the root of $\mathcal{A}_k^{n-1}$.

The structure of $\mathcal{A}_k^n$ resembles the structure of the largest (in terms of number of leaves) binary tree of depth n that does not have the full binary tree of depth $k + 1$ as a minor, precisely: It can be proven by induction on the construction of the $\mathcal{A}_k^n$, that $\mathcal{A}_k^n$ has precisely $\ell_2^k(n)$ leaves, so from Lemma 12 it follows that $\mathcal{A}_k^n$ has at least $(n - k)^k$ leaves.

From now, the proof is similar to the one of Lemma 14. For the sake of contradiction, assume that there is a $k \geq 1$, such that Q_{2k+3} is rewritable into a linear Datalog program Π of diameter k on the class $\mathfrak{C}_i$. Choose n very large (we will make this precise later) and let $\mathcal{A} = \mathcal{A}_{2k+3}^n$, so $\mathcal{A}$ is a minimal tree-shaped ABox for Q_{2k+3} with root a_0, depth at most $n(i + 1)$ and that has at least $(n - 2k - 3)^{2k+3}$ leaves.

We have $\mathcal{A}, \mathcal{T} \models \Pi(a_0)$ and thus there is a derivation of $\Pi(a_0)$ in $\mathcal{A}$. Consider the ABox $\mathcal{A}_D$. By Lemma 13, we have the following:

1. $\mathcal{A}_D \models \Pi(a_0)$;
2. there is a homomorphism h from $\mathcal{A}_D$ to $\mathcal{A}$ with $h(a_0) = a_0$;
3. $\mathcal{A}_D$ has pathwidth at most k.

We manipulate $\mathcal{A}_D$ as follows:

- Restrict the degree to $|\mathcal{T}|$ by taking a subset according to Lemma 28.
- Remove all assertions that involve an individual a which is not reachable from a_0 along role edges, that is, for which there are no assertions $r_1(a_0, a_1), \ldots, r_n(a_{n-1}, a_m)$ with $a_m = a$.

We use $\mathcal{B}$ to denote the resulting ABox. It can be verified that Conditions 1 to 3 still hold when $\mathcal{A}_D$ is replaced with $\mathcal{B}$. In particular, this is true for Condition 1 since $\mathcal{A}_D \models \Pi(a_0)$ iff

$\mathcal{A}_D, \mathcal{T} \models A_0(a_0)$ iff $\mathcal{B}, \mathcal{T} \models A_0(a_0)$ iff $\mathcal{B} \models \Pi(a_0)$. The second equivalence is easy to establish by showing how a model witnessing $\mathcal{B}, \mathcal{T} \not\models A_0(a_0)$ can be transformed into a model that witnesses $\mathcal{A}_D, \mathcal{T} \not\models A_0(a_0)$.

Choose a homomorphism h from $\mathcal{B}$ to $\mathcal{A}$ with $h(a_0) = a_0$. Then h must be surjective since otherwise, the restriction $\mathcal{A}^-$ of $\mathcal{A}$ to the individuals in the range of h would satisfy $\mathcal{A}^-, \mathcal{T} \models A_0(a_0)$, contradicting the minimality of $\mathcal{A}$. Let $a_1, \dots, a_m$ be the leaves of $\mathcal{A}$, $m \geq (n - 2k - 3)^{2k+3}$. For each a_i, choose a b_i with $h(b_i) = a_i$. Clearly, all individuals in $b_1, \dots, b_m$ must be distinct.

By construction, $\mathcal{B}$ is connected. Since $\mathcal{B} \to \mathcal{A}$, $\mathcal{B}$ must further be a DAG (directed acyclic graph). We proceed to exhaustively remove assertions from $\mathcal{B}$ as follows: whenever $r(c_1, c), r(c_2, c) \in \mathcal{B}$ with $c_1 \neq c_2$, then choose and remove one of these two assertions. Using the fact that every individual in $\mathcal{B}$ is reachable from a_0, it can be proved by induction on the number of edge removals that the obtained ABoxes

(i) remain connected and

(ii) contain the same individuals as $\mathcal{B}$, that is, edge removal never results in the removal of an individual.

Point (i) and the fact that we start from a DAG-shaped ABox means that the ABox $\mathcal{B}_t$ ultimately obtained by this manipulation is tree-shaped. By construction of $\mathcal{B}_t$, h is still a homomorphism from $\mathcal{B}_t$ to $\mathcal{A}$, $\mathcal{B}_t$ has pathwidth at most k, and the individuals $b_1, \dots, b_m$ are leaves in $\mathcal{B}_t$ (and thus $\mathcal{B}_t$ has at least $(n - 2k - 3)^{2k+3}$ leaves. From the former, it follows that the depth of $\mathcal{B}_t$ is at most $n(i + 1)$.

Assume that $\mathcal{B}_t$ does not contain the full binary tree of depth $2k + 3$ as a minor. Then by Lemma 12, the number of leaves of $\mathcal{B}_t$ is at most $(2k + 3)(|\mathcal{T}| - 1)^{2k+2}(n(i + 1))^{2k+2}$, which is polynomial of degree $2k + 2$ in n. So if n was chosen large enough such that $(n - 2k - 3)^{2k+3} > (2k + 3)(|\mathcal{T}| - 1)^{2k+2}(n(i + 1))^{2k+2}$ in the beginning, this leads to a contradiction. Hence, $\mathcal{B}_t$ must contain as a minor the full binary tree of depth at least $2k + 3$. But it is well-known that any such tree has pathwidth at least $k + 1$, in contradiction to $\mathcal{B}_t$ having pathwidth at most k. □

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

Proof. Assume to the contrary of what we have to show that $Q_{8\ell+13}$ is rewritable into a linear Datalog program Π_0 of width ℓ. Let k be the diameter of Π_0. Clearly, Π_0 is also a rewriting of $Q_{8\ell+13}$ on the class of ABoxes $\mathfrak{C}_k$.

We carry out a sequence of three rewriting steps on Π_0. In the first step, let Π_1 be obtained from Π_0 by replacing every rule $S(\mathbf{x}) \leftarrow q(\mathbf{y})$ in Π_0 with the set of all rules $S(\mathbf{x}') \leftarrow q(\mathbf{y}')$ that can be obtained from the original rule by consistenly identifying variables in the rule body and head such that the restriction of $q(\mathbf{y}')$ to EDB relations (that is, concept and role names in Σ) takes the form of a forest in which every tree branches at most once. This step preserves equivalence on $\mathfrak{C}_k$ since every homomorphism from the body of a rule in Π into an ABox from $\mathfrak{C}_k$ (and also to the extension of such an ABox with IDB relations) induces a variable identification that identifies a corresponding rule produced in the rewriting.

In the next step, we rewrite Π_1 into a linear Datalog program Π_2, as follows. Let $S(\mathbf{x}) \leftarrow q(\mathbf{y})$ be a rule in Π_1 and let V be the set of *special* variables in $q(\mathbf{y})$, that is, variables that occur also in $\mathbf{x}$ or in the IDB atom in $q(\mathbf{y})$, if existant. We obtain a new rule body $q'(\mathbf{y}')$ from $q(\mathbf{y})$ in the following way:

1. remove the IDB atom (if existant), obtaining a forest-shaped rule body;
2. remove all trees that do not contain a special variable;
3. re-add the IDB atom (if existant).

In Π_2, we replace $S(\mathbf{x}) \leftarrow q(\mathbf{y})$ with $S(\mathbf{x}) \leftarrow q'(\mathbf{y}')$.

We argue that, on the class of ABoxes $\mathfrak{C}_k$, Π_2 is equivalent to Π_1. Thus let $\mathcal{A}$ be an ABox from $\mathfrak{C}_k$ and $a \in \mathsf{Ind}(\mathcal{A})$ such that $\mathcal{A} \models \Pi_2(a)$. We have to show that $\mathcal{A} \models \Pi_1(a)$. Let $q_1(\mathbf{x}_1), \dots, q_p(\mathbf{x}_p)$ be all trees that have been removed from a rule body during the construction of Π_2. Let $\mathcal{A}_i$ be $q_1(\mathbf{x}_1)$ viewed as a Σ-ABox, $1 \leq i \leq p$. Note that each $\mathcal{A}_i$ must be in $\mathfrak{C}_k$. Let $\mathcal{B}$ be the disjoint union of the ABoxes $\mathcal{A}, \mathcal{A}_1, \dots, \mathcal{A}_p$, assuming that these ABoxes do not share any individual names, and note that $\mathcal{B}$ is in $\mathfrak{C}_k$. Since $\mathcal{A} \models \Pi_2(a)$, we must have $\mathcal{B} \models \Pi_2(a)$. By construction of $\mathcal{B}$, this clearly implies $\mathcal{B} \models \Pi_1(a)$. Consequently, $\mathcal{B} \models Q_{8\ell+13}(a)$. Since answers to OMQs from $(\mathcal{EL}, \text{AQ})$ depend only on the reachable part of ABoxes, we obtain that $\mathcal{A} \models Q_{8\ell+13}(a)$, thus $\mathcal{A} \models \Pi_1(a)$ as required.

At this point, let us sum up the most important properties of the linear Datalog program Π_2: it is a rewriting of $Q_{8\ell+13}$ on $\mathfrak{C}_k$, has width at most ℓ and diameter at most k, and

($*$) the restriction of the rule body to EDB relations is a forest that consists of at most 2ℓ trees.

Note that the upper bound of 2ℓ is a consequence of the fact that, by construction of Π_2, each of the relevant trees contains at least one special variable.

We now rewrite Π_2 into a final linear Datalog program Π_3 that is equivalent to Π_2, has width at most $4\ell+2$, and diameter at most $4\ell + 5$. Thus Π_3 is a rewriting of $Q_{8\ell+13}$ on $\mathfrak{C}_k$ of diameter $4\ell + 5$, which is a contradiction to Lemma 17.

It thus remains to give the construction of Π_3. Let $\rho = S(\mathbf{x}) \leftarrow q(\mathbf{y})$ be a rule in Π_2 and let $\mathbf{y}' \subseteq \mathbf{y}$ be the set of variables x that are special or a branching variable where the latter means that $q(\mathbf{y})$ contains atoms of the form $r(x, y_1)$, $s(x, y_2)$ with $y_1 \neq y_2$. Due to ($*$), $\mathbf{y}'$ contains at most 4ℓ variables. Let $q'(\mathbf{y}')$ be the restriction of $q(\mathbf{y})$ to the variables in $\mathbf{y}'$. By construction of Π_2, it can be verified that $q(\mathbf{y})$ is the union of $q'(\mathbf{y}')$ and path-shaped $q_1(\mathbf{y}_1), \dots, q_p(\mathbf{y}_p)$ such that for $1 \leq i \leq p$,

- $q_i(\mathbf{y}_i)$ contains only EDB atoms,
- there are at most two variables shared by $q_i(\mathbf{y}_i)$ and $q'(\mathbf{y}')$, denoted by $\mathbf{x}_i$, and
- the queries $q_1(\mathbf{y}_1), \dots, q_p(\mathbf{y}_p)$ do not share any variables.

We thus find linear Datalog programs $\Gamma_1, \dots, \Gamma_p$ that are at most binary, of width at most two and diameter at most three such that for any Σ-ABox $\mathcal{A}$ and $\mathbf{a} \subseteq \mathsf{Ind}(\mathcal{A})$, $\mathcal{A} \models \Gamma_i(\mathbf{a})$ iff there is a homomorphism h_i from $q_i(\mathbf{y}_i)$ to $\mathcal{A}$ such that $h_i(\mathbf{x}_i) = \mathbf{a}$. Let the goal relations of $\Gamma_1, \dots, \Gamma_p$ be $G_1, \dots, G_p$. We assume w.l.o.g. that the programs $\Gamma_1, \dots, \Gamma_p$

do not share variables or IDB relations, and neither do they share variables or IDB relations with Π_2. In Π_3, we replace $S(\mathbf{x}) \leftarrow q(\mathbf{y})$ with the following rules:

- for any rule $P(\mathbf{x}) \leftarrow p(\mathbf{z})$ in Γ_1 where $p(\mathbf{z})$ contains only EDB atoms, the rule $X^P_\rho(\mathbf{y}', \mathbf{x}) \leftarrow q'(\mathbf{y}') \wedge p(\mathbf{z})$;
- for any rule $P(\mathbf{x}) \leftarrow p(\mathbf{z})$ in Γ_i, $1 < i \leq n$, where $p(\mathbf{z})$ contains only EDB atoms, the rule $X^P_\rho(\mathbf{y}', \mathbf{x}) \leftarrow X^{G_{i-1}}_\rho(\mathbf{y}', \mathbf{x}_{i-1}) \wedge p(\mathbf{z})$;
- for any rule $P(\mathbf{x}) \leftarrow p(\mathbf{z})$ in Γ_i, $1 \leq i \leq n$, where $p(\mathbf{z})$ contains the IDB atom $Q(\mathbf{u})$, the rule $X^P_\rho(\mathbf{y}', \mathbf{x}) \leftarrow X^Q_\rho(\mathbf{y}', \mathbf{u}) \wedge p(\mathbf{z})$;
- the rule $S(\mathbf{x}) \leftarrow X^{G_p}_\rho(\mathbf{y}', \mathbf{x}_p)$.

where the goal relations of $\Gamma_1, \dots, \Gamma_p$ become normal IDB relations. It can be verified that Π_3 is as required. □

C Proofs for Section 4

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

Proof. We use a pumping argument. Let $Q = (\mathcal{T}, \Sigma, A_0(x)) \in (\mathcal{EL}, \text{AQ})$ have unbounded depth. Set $k := 3^{|\mathcal{T}|} + 1$. Since Q has unbounded depth, there is a tree-shaped ABox $\mathcal{A}$ of depth k with root a such that $\mathcal{A}, \mathcal{T} \models A_0(a)$ and $\mathcal{A}$ is minimal with this property. $\mathcal{A}$ contains a path from the root to a leaf that is of length at least k. Let $\mathcal{A}'$ denote the ABox obtained from $\mathcal{A}$ by removing all assertions involving the leaf in this path. Since $\mathcal{A}$ is minimal, $\mathcal{A}', \mathcal{T}' \not\models A_0(a)$. For every individual b on the remaining path, we consider the pair (t'_b, t_b) where $t'_b = \mathsf{tp}_{\mathcal{A}',\mathcal{T}}(b)$ and $t_b = \mathsf{tp}_{\mathcal{A},\mathcal{T}}(b)$. Observing $t'_b \subseteq t_b$, we obtain $3^{|\mathcal{T}|} = k - 1$ as an upper bound for the number of different pairs that may occur on the path. The remaining path has k individuals, so by the pigeonhole principle there are distinct individuals b and c with $(t'_b, t_b) = (t'_c, t_c)$. W.l.o.g., let c be a descendant of b. We set $t_0 = t'_b$ and $t_1 = t_b$. It is now easy to verify that $\mathcal{A}$, a, b, c, t_0 and t_1 satisfy Conditions 1 to 4 from Definition 21. □

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

Proof. Let $(\mathcal{T}, \Sigma, A_0(x)) \in (\mathcal{EL}, \text{AQ})$ have the ability to simulate REACH. Then there is an ABox $\mathcal{A}$ with root a, distinguished individuals b and c, and sets $t_0 \subsetneq t_1$ of concept names as in Definition 21. We reduce REACH to Q. Let $G = (V, E)$ be a directed graph, $s \in V$ a source node and $t \in V$ a target node. We construct a Σ-ABox $\mathcal{A}_G$. Reserve an individual a_v for every node $v \in V$. For every $(u, v) \in E$, include in $\mathcal{A}_G$ a copy $\mathcal{A}_{u,v}$ of $\mathcal{A}_{\mathsf{edge}}$ and identify c with a_u and b with a_v. Further include in $\mathcal{A}_G$ one copy of $\mathcal{A}_{\mathsf{finish}}$ and identify b with a_t and one copy of $\mathcal{A}_{\mathsf{start}}$ and identify c with a_s. Let a_0 be the root of the copy of $\mathcal{A}_{\mathsf{finish}}$ in $\mathcal{A}_G$. It can be verified that $\mathcal{A}_G$ can be constructed from G using an FO-query. It thus remains to show the following.

Claim. t is reachable from s in G iff $a_0 \in \mathsf{cert}_\mathcal{T}(A_0(x), \mathcal{A}_G)$.

Let t be reachable from s. Then there is a path $s = v_0, \dots, v_n = t$ in G. By definition of $\mathcal{A}_G$, there is a copy of $\mathcal{A}_{\mathsf{start}}$ whose root is a_s, so Condition 2 from Definition 21 yields $t_1 \subseteq \mathsf{tp}_{\mathcal{A},\mathcal{T}}(a_s)$. Between any two $a_{v_i}, a_{v_{i+1}}$ there is a copy of $\mathcal{A}_{\mathsf{edge}}$, so we inductively obtain $t_1 \subseteq \mathsf{tp}_{\mathcal{A},\mathcal{T}}(a_{v_i})$ for all i. In particular, $t_1 \subseteq \mathsf{tp}_{\mathcal{A},\mathcal{T}}(a_t)$. Finally, there is a copy of $\mathcal{A}_{\mathsf{finish}}$ in which b is identified with a_t. By Condition 1, we have $A_0 \in \mathsf{tp}_{\mathcal{A},\mathcal{T}}(a_0)$, i.e. $a_0 \in \mathsf{cert}_\mathcal{T}(A_0(x), \mathcal{A}_G)$.

For the other direction, assume that t is not reachable from s. Set

$$\begin{aligned} \mathcal{A}'_G \quad := \quad & \mathcal{A}_G \cup \{t_0(a_v) \mid v \in V \text{ is not reachable from } s\} \\ & \cup \{t_1(a_v) \mid v \in V \text{ is reachable from } s\}. \end{aligned}$$

We show that $\mathcal{A}'_G, \mathcal{T} \not\models A_0(a_0)$, which implies $\mathcal{A}_G, \mathcal{T} \not\models A_0(a_0)$.

We have defined $\mathcal{A}'_G$ as an extension of $\mathcal{A}_G$. Alternatively and more suitable for what we want to prove, $\mathcal{A}'_G$ can be obtained by starting with an ABox $\mathcal{A}_0$ that contains only the assertions $t_0(a_v)$ for all unreachable nodes $v \in V$ as well as $t_1(a_v)$ for all reachable nodes $v \in V$, and then exhaustively applying the following rules in an unspecified order:

- Choose an edge $(u, v) \in E$ that has not been chosen before, take a new copy of $\mathcal{A}_{\mathsf{edge}}$, rename c to a_u and b to a_v, and add the assertions $M_{\mathsf{reach}(x)}(a_x)$ for $x \in \{u, v\}$ where $\mathsf{reach}(x) = 1$ if x is reachable from s and $\mathsf{reach}(x) = 0$ otherwise. Let this modified copy of $\mathcal{A}_{\mathsf{edge}}$ be called $\mathcal{A}^{u,v}_{\mathsf{edge}}$. Now $\mathcal{A}_{i+1}$ is defined as the union of $\mathcal{A}_i$ and $\mathcal{A}^{u,v}_{\mathsf{edge}}$.
- Introduce a new copy of $\mathcal{A}_{\mathsf{start}}$, rename b to a_s and add the assertions $t_1(a_s)$. Let this altered copy of $\mathcal{A}_{\mathsf{start}}$ be called $\mathcal{A}^s_{\mathsf{start}}$. Now $\mathcal{A}_{i+1}$ is defined as the union of $\mathcal{A}_i$ and $\mathcal{A}^s_{\mathsf{start}}$.
- Introduce a new copy of $\mathcal{A}_{\mathsf{finish}}$, rename b to a_t and add the assertions $t_0(a_t)$. Let this altered copy of $\mathcal{A}_{\mathsf{finish}}$ be called $\mathcal{A}^t_{\mathsf{finish}}$. Now $\mathcal{A}_{i+1}$ is defined as the union of $\mathcal{A}_i$ and $\mathcal{A}^t_{\mathsf{finish}}$.

Clearly, rule application terminates after $|E| + 2$ steps and results in the ABox $\mathcal{A}'_G$.

Claim. $\mathsf{tp}_{\mathcal{A}_i,\mathcal{T}}(u) = t_0$ if $u \in V$ is unreachable and $\mathsf{tp}_{\mathcal{A}_i,\mathcal{T}}(v) = t_1$ if $v \in V$ is reachable, for all $i > 0$.

The proof is by induction over i. For $i = 0$, the statement is clear since t_0 and t_1 are $\mathcal{T}$-types. Now assume the statement is true for some i and consider $\mathcal{A}_{i+1}$. If $\mathcal{A}_{i+1}$ was obtained by the first rule, it can be verified using conditions 2 and 3 from Definition 21 that $\mathsf{tp}_{\mathcal{A}^{u,v}_{\mathsf{edge}},\mathcal{T}}(v) = M_{\mathsf{acc}(v)}$ for all $x \in \{u, v\}$. So with Lemma 29 and since $\mathcal{A}_i$ and $\mathcal{A}^{u,v}_{\mathsf{edge}}$ share only the individuals u, v, the statement follows. If $\mathcal{A}_{i+1}$ was obtained by the second rule it follows from Condition 2 that $\mathsf{tp}_{\mathcal{A}^s_{\mathsf{start}},\mathcal{T}}(a_s) = t_1$ and with Lemma 21, the statement follows. If $\mathcal{A}_{i+1}$ was obtained by the third rule, it can be verified that $\mathsf{tp}_{\mathcal{A}^t_{\mathsf{finish}},\mathcal{T}}(a_t) = t_0$ and with Lemma 29, the statement follows. This finishes the proof of the claim.

The claim yields $\mathsf{tp}_{\mathcal{A}'_G,\mathcal{T}}(a_t) = t_0$, so by Condition 4 from Definition 21 it follows that $\mathcal{A}'_G, \mathcal{T} \not\models A_0(a_0)$, as required. □

D Proofs for Section 5

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*).*

Proof. In (the appendix of) [Bienvenu *et al.*, 2013], it is proved that FO-rewritability in $(\mathcal{EL}, \text{AQ})$ is EXPTIME-hard. The proof is by a reduction of the word problem of a polynomially space bounded alternating Turing machine (ATM) M that solves an EXPTIME-complete problem. The reduction exhibits a polynomial time algorithm that constructs, given an input w to M, an OMQ $Q = (\mathcal{T}, \Sigma, B(x)) \in (\mathcal{EL}, \text{AQ})$ such that Q is not FO-rewritable iff M accepts w. A careful inspection of the construction of Q and of the "if" part of the proof reveals that

($*$) if M accepts w, then Q is unboundedly branching, thus not linear Datalog rewritable, PTIME-hard, NL-hard, and not in NL (unless NL = PTIME).

Conversely, if M does not accept w, then FO-rewritability of Q implies that Q is

- in L and thus neither NL-hard (unless L = NL) nor PTIME-hard (unless L = PTIME).
- linear Datalog rewritable (since every FO-rewritable OMQ from $(\mathcal{EL}, \text{AQ})$ is rewritable into a UCQ [Bienvenu *et al.*, 2013]).

The stated hardness results follow.

We expand a little bit on ($*$), using the terminology from [Bienvenu *et al.*, 2013]. Assume that M accepts w. We have to argue that for any $k \geq 0$, there is a Σ-ABox $\mathcal{A}$ such that $\mathcal{A}, \mathcal{T} \models B(a)$, a the root of $\mathcal{A}$, $\mathcal{A}$ is minimal with this property, and $\mathcal{A}$ contains as a minor the full binary tree of depth k. Thus fix a $k > 0$. Since M accepts w, there is an accepting computation tree T of M on w. T can be converted into a tree-shaped ABox $\mathcal{A}_T$ in a straightforward way: introduce one individual name for each configuration, use the concept names from $\mathcal{C}$ to describe the actual configurations at their corresponding individual names (see [Bienvenu *et al.*, 2013]), and use the role names r_1 and r_2 to connect configurations in the intended way. Let m be the number of leaf nodes in $\mathcal{A}_T$. By starting with $\mathcal{A}_T$ and then repeatedly appending copies of $\mathcal{A}_T$ to leaf nodes, we can construct an ABox $\mathcal{A}$ that is a "full m-ary tree of $\mathcal{A}_T$-trees of depth k". At each leaf of $\mathcal{A}$, we add the concept name Start. Clearly, $\mathcal{A}$ contains as a minor the full binary tree of depth k. It can be verified that $\mathcal{A}, \mathcal{T} \models B(a)$, a the root of $\mathcal{A}$, and that $\mathcal{A}$ is minimal with this property, that is, removing any assertion from $\mathcal{A}$ result in $B(a)$ no longer being entailed. □

We now introduce alternating parity automata on finite trees (APTAs). A *tree* is a non-empty (and potentially infinite) set $T \subseteq (\mathbb{N} \setminus 0)^*$ closed under prefixes. We say that T is m*-ary* if for every $n \cdot i \in T$, we have $i \leq m$. For any $n \in (\mathbb{N} \setminus 0)^*$, as a convention we set $n \cdot 0 := n$. For an alphabet Γ, a Γ*-labelled tree* is a pair (T, L) with T a tree and $L : T \to \Gamma$ a node labeling function. For any set X, let $\mathcal{B}^+(X)$ denote the set of all positive Boolean formulas over X, i.e., formulas built using conjunction and disjunction over the elements of X used as propositional variables, and where the special formulas true and false are allowed as well. An *infinite path* P of a tree T is a prefix-closed set $P \subseteq T$ such that for every $i \geq 0$, there is a unique $n \in P$ with $|n| = i$.

Definition 31 (APTA)**.** An *alternating parity tree automaton (APTA) on finite* m*-ary trees* is a tuple $\mathfrak{A} = (S, \Gamma, \delta, s_0, c)$ where S is a finite set of *states*, Γ is a finite alphabet, $\delta : S \times \Gamma \to \mathcal{B}^+(\mathsf{tran}(\mathfrak{A}))$ is the *transition function* with $\mathsf{tran}(\mathfrak{A}) = \{\langle i \rangle s,\ [i]s \mid 0 \leq i \leq m \text{ and } s \in S\}$ the set of *transitions* of $\mathfrak{A}$, $s_0 \in S$ is the *initial state*, and $c : S \to \mathbb{N}$ is the *parity condition* that assigns to each state a *priority*.

Intuitively, a transition $\langle i \rangle s$ with $i > 0$ means that a copy of the automaton in state s is sent to the i-th successor of the current node, which is then required to exist. Similarly, $\langle 0 \rangle s$ means that the automaton stays at the current node and switches to state s. Transitions $[i]s$ mean that a copy of the automaton in state s is sent to the relevant successor if that successor exists (which is not required).

Definition 32 (Run, Acceptance)**.** A *run* of a APTA $\mathfrak{A} = (S, \Gamma, \delta, s_0, c)$ on a finite Γ-labelled tree (T, L) is a $T \times S$-labelled tree (T_r, r) such that the following conditions are satisfied:

1. $r(\varepsilon) = (\varepsilon, s_0)$;
2. if $m \in T_r$, $r(m) = (n, s)$, and $\delta(s, L(n)) = \varphi$, then there is a (possibly empty) set $S \subseteq \mathsf{tran}(\mathfrak{A})$ such that S (viewed as a propositional valuation) satisfies φ as well as the following conditions:
   (a) if $\langle i \rangle s' \in S$, then $n \cdot i \in T$ and there is a node $m \cdot j \in T_r$ such that $r(m \cdot j) = (n \cdot i, s')$;
   (b) if $[i]s' \in S$ and $n \cdot i \in T$, then there is a node $m \cdot j \in T_r$ such that $r(m \cdot j) = (n \cdot i, s')$.

We say that (T_r, r) is *accepting* if on all infinite paths $\varepsilon = n_1 n_2 \cdots$ of T_r, the maximum priority that appears infinitely often is even. A finite Γ-labelled tree (T, L) is *accepted* by $\mathfrak{A}$ if there is an accepting run of $\mathfrak{A}$ on (T, L). We use $L(\mathfrak{A})$ to denote the set of all finite Γ-labelled tree accepted by $\mathfrak{A}$.

Note that, although input trees are finite, runs can be infinite because of transitions of the form $[0]s$; it thus makes sense to use a parity condition. It is known (and easy to see) that APTAs are closed under complementation and intersection, and that these constructions involve only a polynomial blowup.

Let $Q = (\mathcal{T}, \Sigma, A_0(x))$ be an OMQ. We are going to construct, for every pair of $\mathcal{T}$-types t_0, t_1, an automaton $\mathfrak{A}_{t_0,t_1}$ such that Q has the ability to simulate PSA iff $L(\mathfrak{A}_{t_0,t_1}) \neq \emptyset$ for at least one pair t_0, t_1. Thus, let t_0, t_1 be $\mathcal{T}$-types.

Let Γ be the alphabet that consists of all subsets of $\Sigma \cup \{b, c, d\}$ that contain exactly one role name and not more than a single element of $\{b, c, d\}$. A Γ-labelled tree is *proper* if it has exactly one node whose label contains b, and likewise for c and d. We are going to use proper Γ-labelled trees to represent Σ-ABoxes, where each tree node corresponds to an ABox individual and the role name in each node label represents the role via which the current tree node is reachable from its predecessor (except at the root). Because APTAs run on trees of fixed maximum degree, we need to restrict the degree

of ABoxes that witness the ability to simulate PSA. We are going to use $m = 4|\mathcal{T}|$ and shall argue later that this choice is without loss of generality. The elements b, c, d of node labels serve the purpose of marking the distinguished individuals b, c, d in ABoxes that witness the ability to simulate PSA. Formally, a proper Γ-labelled tree (T, L) represents the ABox

$$\begin{array}{rcl}\mathcal{A}_{(T,L)} & := & \{A(n) \mid A \in L(n)\} \cup \\ & & \{r(n,m) \mid r \in L(m) \text{ and } m \text{ child of } n\}.\end{array}$$

We characterize entailment of OMQs in terms of derivation trees. A *derivation tree* for $Q(a_0)$ in a Σ-ABox $\mathcal{A}$ is a finite $\mathsf{Ind}(\mathcal{A}) \times \mathsf{N_C}$-labelled tree (T, V) that satisfies the following conditions:

1. $V(\varepsilon) = (a_0, A_0)$;
2. if $V(x) = (a, A)$ and neither $A(a) \in \mathcal{A}$ nor $\top \sqsubseteq A \in \mathcal{T}$, then one of the following holds:
   - x has successors $y_1, \dots, y_k$, $k \geq 1$ with $V(y_i) = (a, A_i)$ for $1 \leq i \leq k$ and $\mathcal{T} \models A_1 \sqcap \dots \sqcap A_k \sqsubseteq A$;
   - x has a single successor y with $V(y) = (b, B)$ and there is an $\exists r.B \sqsubseteq A \in \mathcal{T}$ such that $r(a, b) \in \mathcal{A}$.

Note that the first item of Point 2 above requires $\mathcal{T} \models A_1 \sqcap \dots \sqcap A_k \sqsubseteq A$ instead of $A_1 \sqcap A_2 \sqsubseteq A \in \mathcal{T}$ to 'shortcut' anonymous parts of the canonical model. In fact, the derivation of A from $A_1 \sqcap \dots \sqcap A_k$ by $\mathcal{T}$ can involve the introduction of anonymous elements. The main property of derivation trees is the following.

Lemma 33. *Let $\mathcal{A}$ be a Σ-ABox. Then $\mathcal{A}, \mathcal{T} \models Q(a)$ iff there is a derivation tree for $Q(a)$ in $\mathcal{A}$.*

The proof is a minor variation of an analogous result for the more expressive description logic $\mathcal{ELI}_\bot$ in [Bienvenu *et al.*, 2013], proof details are omitted.

We are going to construct $\mathfrak{A}_{t_0,t_1}$ as the intersection of six automata $\mathfrak{A}_0, \mathfrak{A}_1, \dots, \mathfrak{A}_5$, where $\mathfrak{A}_0$ ensures that the input tree (T, L) is proper and for $1 \leq i \leq 5$, $\mathfrak{A}_i$ ensures that the corresponding condition i from the definition of the ability to simulate PSA is satisfied, where the two equalities in condition 4 count as two conditions. We start with the automaton $\mathfrak{A}_1$, which has to make sure that $\mathcal{A}_{(T,L)}, \mathcal{T} \models A_0(a)$ with a the root of $\mathcal{A}_{(T,L)}$. We are going to use derivation trees and Lemma 33. Set $\mathfrak{A}_1 = (S, \Gamma, \delta, s_0, c)$ where

$$S = \{s_A \mid A \in \Sigma \cap \mathsf{N_C}\} \cup \{s_r \mid r \in \Sigma \cap \mathsf{N_R}\},$$

$s_0 = s_{A_0}$, and c assigns one to all states, that is, the accepting runs are exactly the finite runs. The transition function δ is defined as follows:

$$\delta(s_A, \sigma) = \mathsf{true} \qquad \text{if } A \in \sigma \text{ or } \top \sqsubseteq A \in \mathcal{T}$$

$$\delta(s_A, \sigma) = \bigvee_{\mathcal{T} \models A_1 \sqcap \dots \sqcap A_n \sqsubseteq A} \big(\langle 0 \rangle s_{A_1} \wedge \dots \wedge \langle 0 \rangle s_{A_n}\big) \vee \bigvee_{\exists r.B \sqsubseteq A \in \mathcal{T}} \bigvee_{i \in 1..m} \langle i \rangle (s_B \wedge s_r) \qquad \text{if } A \notin \sigma \text{ and } \top \sqsubseteq A \notin \mathcal{T}$$

$$\delta(s_r, \sigma) = \mathsf{true} \qquad \text{if } r \in \sigma$$

$$\delta(s_r, \sigma) = \mathsf{false} \qquad \text{if } r \notin \sigma$$

It should be obvious how the above transitions verify the existence of a derivation tree. Note that the tree must be finite since runs are required to be finite. This finishes the definition of $\mathfrak{A}_1$. The automata $\mathfrak{A}_2, \dots, \mathfrak{A}_5$ are variations of $\mathfrak{A}_1$. We will give some crucial intuitions, but refrain from working out full details. Regarding $\mathfrak{A}_2$, we present a simplified version that only ensures $\mathsf{tp}_{\mathcal{A}_{(T,L)},\mathcal{T}}(b) = t_1$ (but not, as also required, $\mathsf{tp}_{\mathcal{A}_{(T,L)},\mathcal{T}}(c) = t_1$ and $\mathsf{tp}_{\mathcal{A}_{(T,L)},\mathcal{T}}(d) = t_1$). Set $\mathfrak{A}_2 = (S, \Gamma, \delta, s_0, c)$ where

$$S = \{s_A, s_{\overline{A}} \mid A \in \Sigma \cap \mathsf{N_C}\} \cup \{s_r, s_{\overline{r}} \mid r \in \Sigma \cap \mathsf{N_R}\},$$

and c assigns one to s_0 and all states of the form s_A and zero to all states of the form $s_{\overline{A}}$. The transition function δ contains all transitions from $\mathfrak{A}_1$, plus the following:

$$\delta(s_0, \sigma) = \bigwedge_{A \in t_1} [0] s_A \wedge \bigwedge_{A \in (\Sigma \cap \mathsf{N_C}) \setminus t_1} [0] s_{\overline{A}} \qquad \text{if } b \in \sigma$$

$$\delta(s_0, \sigma) = \bigwedge_{1 \leq i \leq m} [i] s_0 \qquad \text{if } b \notin \sigma$$

$$\delta(s_{\overline{A}}, \sigma) = \mathsf{false} \qquad \text{if } A \in \sigma \text{ or } \top \sqsubseteq A \in \mathcal{T}$$

$$\delta(s_{\overline{A}}, \sigma) = \bigwedge_{\mathcal{T} \models A_1 \sqcap \dots \sqcap A_n \sqsubseteq A} \big([0] s_{\overline{A}_1} \vee \dots \vee [0] s_{\overline{A}_n}\big) \wedge \bigwedge_{\exists r.B \sqsubseteq A \in \mathcal{T}} \bigwedge_{i \in 1..m} [i](s_{\overline{B}} \vee s_{\overline{r}}) \qquad \text{if } A \notin \sigma \text{ and } \top \sqsubseteq A \notin \mathcal{T}$$

$$\delta(s_{\overline{r}}, \sigma) = \mathsf{false} \qquad \text{if } r \in \sigma$$

$$\delta(s_{\overline{r}}, \sigma) = \mathsf{true} \qquad \text{if } r \notin \sigma$$

Note that the transitions for $s_{\overline{A}}$ and $s_{\overline{r}}$ are simply the duals of those for s_A and s_r. This together with the definition of the acceptance condition implies that a state $s_{\overline{A}}$ can be assigned to a tree node n iff $\mathcal{A}_{(T,L)}, \mathcal{T} \not\models A(n)$. This finishes the definition of $\mathfrak{A}_2$. The remaining automata $\mathfrak{A}_3, \dots, \mathfrak{A}_5$ mainly differ from the previous ones in that instead of considering the ABox $\mathcal{A}_{(T,L)}$, they need to consider an ABox obtained from $\mathcal{A}_{(T,L)}$ by replacing the subtree rooted at one of the nodes marked b (or c or d) with the assertions $\{A(b) \mid A \in t_0\}$. This requires to modify the transitions present in $\mathfrak{A}_1$ and $\mathfrak{A}_2$, using the markers b, c, d in the input tree. For example, the first two transitions from $\mathfrak{A}_1$ could be replaced with the following

transitions:

$$\delta(s_A,\sigma) = \mathsf{true} \qquad \text{if } A \in \sigma \text{ or } \top \sqsubseteq A \in \mathcal{T} \text{ or } (b \in \sigma \text{ and } A \in t_0)$$

$$\delta(s_A,\sigma) = \bigvee_{\mathcal{T} \models A_1 \sqcap \cdots \sqcap A_n \sqsubseteq A} \big(\langle 0\rangle s_{A_1} \wedge \cdots \wedge \langle 0\rangle s_{A_n}\big) \vee \bigvee_{\exists r.B \sqsubseteq A \in \mathcal{T}} \bigvee_{i \in 1..m} \langle i\rangle (s_B \wedge s_r) \qquad \text{if } A \notin \sigma \text{ and } \top \sqsubseteq A \notin \mathcal{T} \text{ and } b \notin \sigma$$

$$\delta(s_A,\sigma) = \bigvee_{\mathcal{T} \models A_1 \sqcap \cdots \sqcap A_n \sqsubseteq A} \big(\langle 0\rangle s_{A_1} \wedge \cdots \wedge \langle 0\rangle s_{A_n}\big) \qquad \text{if } b \in \sigma \text{ and } A \notin t_0$$

We omit further details of the automata $\mathfrak{A}_3, \ldots, \mathfrak{A}_5$.

Lemma 34. *Q has the ability to simulate* PSA *iff $L(\mathfrak{A}_{t_0,t_1}) \neq \emptyset$ for some t_0, t_1.*

Proof. First assume that there are $\mathcal{T}$-types t_0, t_1 such that $L(\mathfrak{A}_{t_0,t_1}) \neq \emptyset$. Then there is a Γ-labelled tree $(T, L) \in L(\mathfrak{A}_{t_0,t_1})$. Using the construction of $\mathfrak{A}_{t_0,t_1}$ it can be verified that $\mathcal{A} = \mathcal{A}_{(T,L)}$, t_0, and t_1 satisfy Conditions 1 to 4 from Definition 3, thus Q has the ability to simulate PSA.

Conversely, assume that Q has the ability to simulate PSA. Then there are $\mathcal{T}$-types t_0, t_1 and a tree-shaped Σ-ABox $\mathcal{A}$ such that Conditions 1 to 4 from Definition 3 are satisfied. Using exactly the same arguments as in the proof of Lemma 28, we find a sub-ABox $\mathcal{A}' \subseteq \mathcal{A}$ of degree at most $m = 4|\mathcal{T}|$ that still satisfies Conditions 1 to 4 from Definition 3; the factor of 4 stems from the fact that these conditions mention not only $\mathcal{A}$, but also three sub-ABoxes of $\mathcal{A}$. It is now easy to encode $\mathcal{A}$ as a Γ-labelled tree (T, L) and to verify that $(T, L) \in \mathfrak{A}_{t_0,t_1}$, which finishes the proof. □

It is not hard to verify that the number of states of the resulting overall automaton $\mathfrak{A}_{t_0,t_1}$ is polynomial in the size of $\mathcal{O}$. Since the emptiness of APTAs can be decided in time single exponential in the number of states (and polynomial in the size of all other components of the automaton) and since we need to build at most single exponentially many automata $\mathfrak{A}_{t_0,t_1}$, we obtain Theorem 25.