A Complete Classification of the Complexity and Rewritability of Ontology-Mediated Queries based on the Description Logic $\mathcal{EL}$

Carsten Lutz and Leif Sabellek

Department of Computer Science, University of Bremen, Germany

Abstract

We provide a fine-grained analysis of the data complexity and rewritability of ontology-mediated queries (OMQs) based on an $\mathcal{EL}$ ontology and a conjunctive query (CQ). 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 in terms of bounded depth and (un)bounded pathwidth of certain minimal ABoxes on which the OMQ yields an answer. We also show that each of the associated meta problems such as deciding whether a given OMQ is rewritable into linear Datalog is ExpTime-complete. We also 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.

Keywords: description logic, ontology-mediated querying, complexity classification, rewritability, linear datalog

1. Introduction

An important application of ontologies is to enrich data with a semantics and with domain knowledge while also extending the vocabulary that is available for query formulation beyond the relation symbols that occur in the database [1, 2, 3, 4]. In a medical application, for example, a database could record the concrete diseases of patients such as asthma while the ontology enriches the vocabulary by adding general categories of diseases such as respiratory disease and cardiovascular disease, as well as the knowledge that asthma is a respiratory disease. This would then enable to query for all patients with a respiratory disease despite the fact that general categories such as 'respiratory disease' do not occur in the vocabulary of the database.

In this context, it is common to view the combination of an ontology and a database query as a compound query, commonly referred to as an *ontology-mediated query (OMQ)* [3]. An *OMQ language* $(\mathcal{L}, \mathcal{Q})$ is then constituted by an ontology language $\mathcal{L}$ and a query language $\mathcal{Q}$. Prominent choices for $\mathcal{L}$ include many description logics (DLs) such as $\mathcal{EL}$, Horn-$\mathcal{SHIQ}$, and $\mathcal{ALC}$ [5] while the most common choices for $\mathcal{Q}$ are conjunctive queries (CQs), unions thereof (UCQs), and the simple atomic queries (AQs) which are of the form $A(x)$ with A a monadic relation symbol. Substantial research efforts have been invested into understanding the properties of the resulting OMQ languages, with two important topics being

1. the *data complexity* of OMQ evaluation, where data complexity means that only the data is considered to be the input while the OMQ is fixed [6, 7, 8, 9, 3], and

2. the *rewritability* of OMQs into more traditional database query languages such as SQL (which in this context is often equated with first-order logic) and Datalog [10, 11, 3, 12, 13, 14].

Note that both topics are connected to practical concerns. It is considered to be a necessary condition for efficient query evaluation in practice that the data complexity is in PTime, preferably even lower. Moreover, most database systems are unaware of ontologies, and thus rewriting OMQs into standard database query languages provides an important avenue for implementing OMQ execution in practical applications [1, 15, 16, 17]. Data complexity and rewritability 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. We remark that FO-rewritability of an OMQ implies rewritability into a UCQ and thus into Datalog [3]. From now on, when speaking about complexity we always mean data complexity.

Regarding compexity and rewritability, modern DLs can roughly be divided into two families: 'expressive DLs' such as $\mathcal{ALC}$ and $\mathcal{SHIQ}$ that result in OMQ languages with coNP complexity and where rewritability is guaranteed neither into FO nor into Datalog [3, 17, 14], and 'Horn DLs' such as $\mathcal{EL}$ and Horn-$\mathcal{SHIQ}$ that typically result in OMQ languages with PTime complexity and where rewritability into (monadic) Datalog is guaranteed, but FO-rewritability is not [11, 15, 18]. From a practical perspective, however, it does not seem necessary to guarantee that *all* OMQs formulated in the OMQ language used have good properties regarding complexity and rewritability. Instead, it is much more relevant to understand whether the *concrete* ontologies and OMQs that are of interest for the application are well-behaved. This potentially makes a difference given that ontology engineers tend to use 'potentially expensive' language features, but might well use them in a 'computationally harmless' way. Initiated in [19, 3], this perspective has led to studies of data complexity and rewritability that are much more fine-grained than the analysis of entire ontology languages. The ultimate aim is to understand, for relevant OMQ languages $(\mathcal{L}, \mathcal{Q})$, the exact complexity and rewritability status of every OMQ from $(\mathcal{L}, \mathcal{Q})$.

The aim of this paper is to carry out such an ultimately fine-grained analysis of the data complexity and rewritability of OMQs from the languages $(\mathcal{EL}, \text{CQ})$

and $(\mathcal{EL}, \text{AQ})$ where $\mathcal{EL}$ is a fundamental and widely known Horn DL for which standard reasoning problems such as subsumption can be solved in PTime and that is at the core of the OWL EL profile of the OWL 2 ontology language [20]. In fact, we completely settle the complexity and rewritability status of each OMQ from $(\mathcal{EL}, \text{CQ})$. Our first main result is a trichotomy for data complexity: Every OMQ from $(\mathcal{EL}, \text{CQ})$ is in AC^0, NL-complete, or PTime-complete, and all three complexities actually occur already in $(\mathcal{EL}, \text{AQ})$. We consider this a remarkable sparseness of complexities. Let us illustrate the trichotomy using an example. In description logics, a TBox is used to formulate an ontology while an ABox plays the role of the database. An OMQ from $(\mathcal{EL}, \text{CQ})$ is then a triple $(\mathcal{T}, \Sigma, q)$ with $\mathcal{T}$ an $\mathcal{EL}$ TBox that represents the ontology, q a CQ, and Σ an ABox signature, that is, a set of concept and role names that can occur in the data.

Example 1. Consider an ontology that represents knowledge about genes. A person carries $\mathsf{Gene1}$ if both parents carry $\mathsf{Gene1}$, and a person carries $\mathsf{Gene2}$ if at least one parent carries $\mathsf{Gene2}$ (dominant and recessive inheritance, respectively). We use the TBox

$$\begin{aligned}\mathcal{T} = \{\ \exists \mathsf{hasFather}.\mathsf{Gene1Carrier} \sqcap \exists \mathsf{hasMother}.\mathsf{Gene1Carrier} &\sqsubseteq \mathsf{Gene1Carrier}\\ \exists \mathsf{hasFather}.\mathsf{Gene2Carrier} &\sqsubseteq \mathsf{Gene2Carrier}\\ \exists \mathsf{hasMother}.\mathsf{Gene2Carrier} &\sqsubseteq \mathsf{Gene2Carrier}\ \}.\end{aligned}$$

For $\Sigma = \{\mathsf{Gene1Carrier}, \mathsf{Gene2Carrier}, \mathsf{hasMother}, \mathsf{hasFather}\}$, the OMQ $Q_1 = (\mathcal{T}, \Sigma, \mathsf{Gene1Carrier}(x))$ is PTime-complete and not rewritable into linear Datalog, while $Q_2 = (\mathcal{T}, \Sigma, \mathsf{Gene2Carrier}(x))$ is NL-complete and rewritable into linear Datalog. Intuitively, Q_2 is easier than Q_1 because yes-instances are witnessed by a *path* to an ancestor who is a $\mathsf{Gene2Carrier}$, whereas for Q_1, a yes-instance is witnessed by a *tree* of ancestors where every ancestor on a leaf of the tree is a $\mathsf{Gene1Carrier}$. Now consider $Q_3 = (\mathcal{T}, \Sigma, \exists y\, \mathsf{hasMother}(x, y))$. This OMQ is even rewritable into first-order logic and can be evaluated in AC^0, since it does not rely on any predicate that is propagated recursively.

Our second main result is that for OMQs from $(\mathcal{EL}, \text{CQ})$, evaluation in NL coincides with rewritability into linear Datalog. It is known that evaluation in AC^0 coincides with FO-rewritability [18] and thus each of the three occurring complexities coincides with rewritability into a well-known database language: AC^0 corresponds to FO, NL to linear Datalog, and PTime to monadic Datalog. We also show that there is no constant bound on the arity of IDB relations in linear Datalog rewritings, that is, we find a sequence of OMQs from $(\mathcal{EL}, \text{CQ})$ (and in fact, even from $(\mathcal{EL}, \text{AQ})$) that are all rewritable into linear Datalog, but require higher and higher arities of IDB relations.[1]

We remark that rewritability into linear Datalog might also be interesting from a practical perspective. In fact, the equation "SQL = FO" often adopted in

[1]In Datalog, the relations in the database are called EDB or *extensional* while the relations that occur in rule heads are called IDB or *intensional*.

ontology-mediated querying 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 provides a natural abstraction of SQL that includes linear recursion, despite the fact that it does not contain full FO. Indeed, the fact that all OMQs from $(\mathcal{EL}, \text{CQ})$ that are FO-rewritable are also UCQ-rewritable indicates 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.

The second main result is proved using a characterization of linear Datalog rewritability in terms of bounded pathwidth that may be of independent interest. It is easiest to state for $(\mathcal{EL}, \text{AQ})$: an OMQ Q is rewritable into linear Datalog (equivalently: can be evaluated in NL) if the class $\mathcal{M}_Q$ of the following ABoxes $\mathcal{A}$ has bounded pathwidth: $\mathcal{A}$ is tree-shaped, delivers the root as an answer to Q, and is minimal w.r.t. set inclusion regarding the latter property. For $(\mathcal{EL}, \text{CQ})$, we have to replace in $\mathcal{M}_Q$ tree-shaped ABoxes with pseudo tree-shaped ones in which the root is an ABox that can have any relational structure, but whose size is bounded by the size of the actual query in q. These results are closely related to results on bounded pathwidth obstructions of CSPs, see for example [21, 22, 23].

Finally, we consider the meta problems associated to the studied properties of OMQs, such as whether a given OMQ is rewritable into linear Datalog, whether it is NL-hard, PTime-hard, etc. Each of these problems turns out to be ExpTime-complete, both in $(\mathcal{EL}, \text{CQ})$ and in $(\mathcal{EL}, \text{AQ})$. In the case of linear Datalog rewritability, our results provide a way of constructing a concrete rewriting when it exists.

The paper is organized as follows. We introduce preliminaries in Section 2 and then start with considering the OMQ language $(\mathcal{EL}, \text{conCQ})$ where conCQ refers to the class of CQs that are connected when viewed as a graph; these CQs might have any arity, including 0.

In Section 3, we show that $(\mathcal{EL}, \text{conCQ})$ enjoys a dichotomy between AC^0 and NL, using a notion of bounded depth that was introduced in [18]. In particular, it was shown in [18] that when the ABoxes in $\mathcal{M}_Q$ have bounded depth, then Q can be evaluated in AC^0. We prove that otherwise, we find certain gadget ABoxes (we say that *Q has the ability to simulate* REACH) that allow us to reduce the reachability problem in directed graphs, thus showing NL-hardness.

In Section 4, we prove a dichotomy between NL and PTime for $(\mathcal{EL}, \text{conCQ})$. We first show that if $\mathcal{M}_Q$ has unbounded pathwidth, then we can find certain gadget ABoxes (we say that *Q has the ability to simulate* PSA) that allow us to reduce the path accessibility problem, thus showing PTime-hardness. This result is similar to, but substantially more difficult than the NL-hardness result in Section 3. We then proceed by showing that if $\mathcal{M}_Q$ has bounded pathwidth, then we can construct a two-way alternating word automaton that accepts suitable representations of pairs $(\mathcal{A}, \mathbf{a})$ where $\mathcal{A}$ is an ABox of low pathwidth and $\mathbf{a}$ and answer to Q on $\mathcal{A}$. We further show how to convert this automaton into a linear Datalog rewriting, which yields NL complexity.

Section 5 is concerned with extending both of our dichotomies to potentially disconnected CQs. In Section 6, we prove that there is a sequence of OMQs that are linear Datalog rewritable but for which the width of IDB relations in linear Datalog rewritings is not bounded by a constant. This strengthens a result by [22] who establish an analogous statement for CSPs. In Section 7, we prove decidability and ExpTime-completeness of the meta problems. The upper bounds are established using the ability to simulate psa from Section 4 and alternating tree automata.

Related Work. Going back at least to [24], studying the data complexity of OMQs based on description logic ontologies has a long history. While it was observed in [24] that expressive DLs have coNP data complexity, a serious consideration of Horn DLs and their PTime data complexity seems to have started only later with [6]. Other relevant publications on the subject of data complexity include [7, 8, 9], the survey [4] has more references. The interest in FO-rewritability started with the proposal of the DL-Lite family of description logics [25], designed so that every OMQ with a DL-Lite ontology is FO-rewritable. For almost all other DLs, FO-rewritability is not guaranteed. This has led to the study of FO-rewritability as a decision problem, for Horn DLs in [11, 15, 18, 26] and for expressive DLs in [3, 14]. Rewritability into Datalog has also received considerable attention. For Horn DLs, it is guaranteed and has been used for efficient implementation of query evaluation [10, 16, 17] and for expressive DLs, it has been considered as a decision problem [14, 27]. Other authors have considered fragments of expressive DLs for which rewritability into Datalog is guaranteed [12] and rewritability into extensions of Datalog with disjunction and negation as failure [13]. We remark that FO-rewritability of OMQs based on Horn DLs is closely related to boundedness of monadic datalog programs, see [28, 29, 30].

As already noted, the classification of the complexity and rewritability of individual OMQs started with [19, 3]. For expressive DLs, this question turned out to be closely related to the complexity classification of constraint satisfaction problems (CSPs) with a fixed template [31]. Very important progress has been made in this area with the proof that CSPs enjoy a dichotomy between PTime and NP [32, 33]. Via the results in [3], this implies that OMQ evaluation in languages such as $(\mathcal{ALC}, \text{UCQ})$ enjoys a dichotomy between PTime and coNP. However, the picture is still far from being fully understood. For example, neither in CSP nor in expressive OMQ languages it is known whether there is a dichotomy between NL and PTime, and whether containment in NL coincides with rewritability into linear Datalog. In [34], a tetrachotomy between AC^0, NL, PTime, and coNP is obtained for the very restricted (yet technically non-trivial) case of (non-Horn) ontologies of the form $\{A \sqsubseteq F \sqcup T, F \sqcap T \sqsubseteq \bot\}$ and for Boolean CQs that are directed paths. A related problem is studied in [35]. Finally, a complexity classification has been undertaken for extensions of DL-Lite and $\mathcal{EL}$ in which predicates can be declared to be closed (which brings in a form of disjunction, thus is a non-Horn setup), again exhibiting connections to (variants of) CSPs [36, 37, 38].

This paper is an extended version of [39]. The main differences are that [39] only treats atomic queries but no conjunctive queries, does not provide characterizations in terms of bounded pathwidth, and achieves less optimal bounds on the width of IDB relations in constructed linear Datalog programs. To support readability, many proofs have been moved to the appendix.

2. Preliminaries

We introduce description logics, ontology-mediated queries, central technical notions such as universal models and the pathwidth of ABoxes, as well as linear Datalog and a fundamental glueing construction for ABoxes. We refer to [5] for more extensive information on description logics and to [40] for background in database theory.

TBoxes and Concepts. In description logic, an ontology is formalized as a TBox. Let $\mathsf{N_C}$, $\mathsf{N_R}$, and $\mathsf{N_I}$ be disjoint countably infinite sets of *concept names*, *role names*, and *individual names*. Concept names should be viewed as unary relations in the sense of first-order logic while role names correspond to binary relations and individual names to constants.

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. While this paper focuses on $\mathcal{EL}$, there are some places where we also consider the extension $\mathcal{ELI}$ of $\mathcal{EL}$ with inverse roles. An $\mathcal{ELI}$*-concept* is built according to the syntax rule $C, D ::= \top \mid A \mid C \sqcap D \mid \exists r.C \mid \exists r^-.C$, the symbol ranges being as in the case of $\mathcal{EL}$-concepts. An expression of the form r^- is an *inverse role*. An $\mathcal{EL}$*-TBox* ($\mathcal{ELI}$-TBox, resp.) is a finite set of *concept inclusions (CIs)* of the form $C \sqsubseteq D$, C and D $\mathcal{EL}$-concepts ($\mathcal{ELI}$-concepts, resp.).

The TBox in Example 1 in the introduction is an $\mathcal{EL}$-TBox. It uses concept names $\mathsf{Gene1Carrier}$ and $\mathsf{Gene2Carrier}$ and role names $\mathsf{hasMother}$ and $\mathsf{hasFather}$. We could use inverse roles, for example, to express that every female parent is the mother of someone using the concept inclusion $\mathsf{Female} \sqcap \mathsf{Parent} \sqsubseteq \exists \mathsf{hasMother}^-.\top$. Adding this CI yields an $\mathcal{ELI}$-TBox.

The *size* of a TBox, a concept, or any other syntactic object O, denoted $|O|$, is the number of symbols needed to write O, with each concept and role name counting as one symbol.

ABoxes. An *ABox* is the DL way to store data. Formally, it is defined as a finite set of *concept assertions* $A(a)$ and *role assertions* $r(a, b)$ where A is a concept name, r is a role name, and a, b are individual names. Every ABox $\mathcal{A}$ is associated with a directed graph $G_\mathcal{A}$ with nodes $\mathsf{ind}(\mathcal{A})$ and edges $\{(a, b) \mid r(a, b) \in \mathcal{A}\}$. Two example ABoxes are

$$\begin{aligned}
\mathcal{A}_1 &= \{\mathsf{hasMother}(a, b), \mathsf{hasFather}(a, c), \mathsf{hasMother}(b, d), \mathsf{hasFather}(b, e) \\
&\quad\ \mathsf{Gene2Carrier}(e)\} \\
\mathcal{A}_2 &= \{\mathsf{hasMother}(a, c), \mathsf{hasFather}(a, d), \mathsf{hasMother}(b, c), \mathsf{hasFather}(b, d) \\
&\quad\ \mathsf{Gene1Carrier}(b), \mathsf{Gene2Carrier}(d)\}.
\end{aligned}$$

Their graphs are shown in Figure 1.

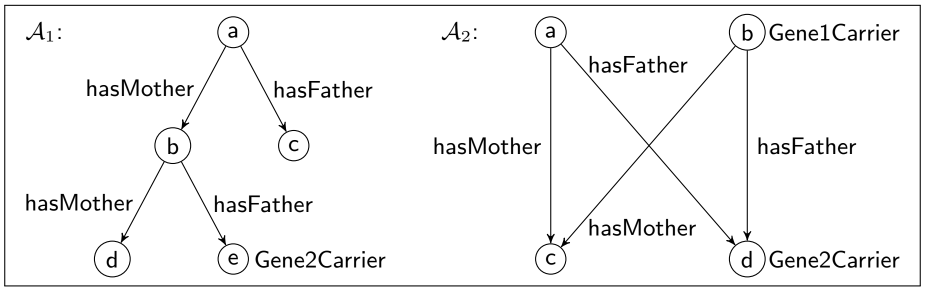

Figure 1: Visual representation of two ABoxes. The ABox $\mathcal{A}_1$ is tree-shaped, while $\mathcal{A}_2$ is not.

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*. We remark that the ABox signature plays the same role as a schema in the database literature [40]. Let $\mathcal{A}$ be an ABox and $a, b \in \mathsf{ind}(\mathcal{A})$. A *path in $\mathcal{A}$ from a to b of length k* is a sequence of assertions $r_0(a_0, a_1), \dots, r_{k-1}(a_{k-1}, a_k) \in \mathcal{A}$ with $a = a_0$ and $b = a_k$. For $S \subseteq \mathsf{ind}(\mathcal{A})$, we use $\mathcal{A}|_S$ to denote the restriction of $\mathcal{A}$ to the assertions that only use individual names from S. A *homomorphism* from an ABox $\mathcal{A}_1$ to an ABox $\mathcal{A}_2$ is a 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 $r(h(a), h(b)) \in \mathcal{A}_2$.

A directed graph G is a *tree* if it is connected, every node has indegree at most 1, and there is a unique node with indegree 0, which is then called the *root* of G. An ABox $\mathcal{A}$ is *tree-shaped* if $G_{\mathcal{A}}$ is a tree and there are no multi-edges, that is, $r(a, b) \in \mathcal{A}$ implies $s(a, b) \notin \mathcal{A}$ for all $s \neq r$. The *root* of a tree-shaped ABox $\mathcal{A}$ is the root of $G_{\mathcal{A}}$ and we call an individual b a *descendant* of an individual a if $a \neq b$ and the unique path from the root to b contains a. For example, the above ABox $\mathcal{A}_1$ is tree-shaped while $\mathcal{A}_2$ is not.

Semantics. An *interpretation* is a tuple $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is a non-empty set, called the *domain* of $\mathcal{I}$, and $\cdot^{\mathcal{I}}$ is a function that assigns to every concept name A a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and to every role name r a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The function $\cdot^{\mathcal{I}}$ can be inductively extended to assign to every $\mathcal{ELI}$ concept C a set $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ in the following way.

$$\begin{aligned}
\top^{\mathcal{I}} &= \Delta^{\mathcal{I}} \\
(C_1 \sqcap C_2)^{\mathcal{I}} &= C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}} \\
(\exists r.C_1)^{\mathcal{I}} &= \{d \in \Delta^{\mathcal{I}} \mid \exists\, e \in \Delta^{\mathcal{I}} : r(d,e) \wedge C_1(e)\} \\
(\exists r^-.C_1)^{\mathcal{I}} &= \{e \in \Delta^{\mathcal{I}} \mid \exists\, d \in \Delta^{\mathcal{I}} : r(d,e) \wedge C_1(d)\}
\end{aligned}$$

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 $(a, b) \in r^{\mathcal{I}}$. Note that we adopt the *standard names assumption* here, meaning that individual names are directly treated as constants rather than being interpreted by $\mathcal{I}$. This is the most common semantics in the context of ontology-mediated querying. It implies the *unique name assumption* which requires distinct individual name to be interpreted as distinct elements in interpretations.

An interpretation is a *model* of a TBox $\mathcal{T}$ if it satisfies all CIs in it and a *model* of an ABox $\mathcal{A}$ if it satisfies all assertions in it. For an interpretation $\mathcal{I}$ and $\Delta \subseteq \Delta^{\mathcal{I}}$, we use $\mathcal{I}|_{\Delta}$ to denote the restriction of $\mathcal{I}$ to the elements in Δ.

Conjunctive queries. A *conjunctive query (CQ)* is of the form $q = \exists \mathbf{y} \phi(\mathbf{x}, \mathbf{y})$, ϕ a conjunction of relational atoms, that uses only unary and binary relations that must be from $\mathsf{N_C}$ and $\mathsf{N_R}$, respectively. An example for a CQ is

$$\exists y\, \mathsf{Gene1Carrier}(x) \wedge \mathsf{hasFather}(x, y) \wedge \mathsf{Gene1Carrier}(y).$$

A *CQ with equality atoms* is a CQ where additionally, atoms of the form $x = y$ are allowed. The variables in $\mathbf{x}$ are called *answer variables* whereas the variables in $\mathbf{y}$ are called *quantified variables.* We set $\mathsf{var}(q) = \mathbf{x} \cup \mathbf{y}$. We also interpret q as the set of its atoms. Every CQ q can be viewed as an ABox $\mathcal{A}_q$ by viewing (answer and quantified) variables as individual names. A CQ is *connected* if $G_{\mathcal{A}_q}$ is connected and *rooted* if every connected component of $G_{\mathcal{A}_q}$ contains at least one answer variable. A CQ is *tree-shaped* if $\mathcal{A}_q$ is. The CQ displayed above is tree-shaped. If q is a CQ and $V \subseteq \mathsf{var}(q)$, then we use $q|_V$ to denote the restriction of q to the atoms that only use variables from V (this may drop answer variables from q). An *atomic query (AQ)* is a CQ of the form $A(x)$.

A *union of conjunctive queries (UCQ)* q is a disjunction of CQs that have the same answer variables $\mathbf{x}$. We write $q(\mathbf{x})$ to emphasize that $\mathbf{x}$ are the answer variables in q. The *arity* of a (U)CQ q, denoted $\mathsf{ar}(q)$, is the number of its answer variables. We say that q is *Boolean* if $\mathsf{ar}(q) = 0$. Slightly overloading notation, we write CQ to denote the set of all CQs, $\text{CQ}^=$ to denote the set of all CQs where equality atoms are allowed, conCQ for the set of all connected CQs, AQ for the set of all AQs, and UCQ for the set of all UCQs.

Let $q(\mathbf{x})$ be a CQ and $\mathcal{I}$ an interpretation. A tuple $\mathbf{a} \in (\Delta^{\mathcal{I}})^{\mathsf{ar}(q)}$ is an *answer to q on $\mathcal{I}$*, denoted $\mathcal{I} \models q(\mathbf{a})$, if there is a *homomorphism* h from q to $\mathcal{I}$ with $h(\mathbf{x}) = \mathbf{a}$, that is, a function $h : \mathsf{var}(q) \to \Delta^{\mathcal{I}}$ such that $A(x) \in q$ implies $h(x) \in A^{\mathcal{I}}$ and $r(x, y) \in q$ implies $(h(x), h(y)) \in r^{\mathcal{I}}$. If q is a UCQ, then $\mathbf{a}$ is an answer to q on $\mathcal{I}$ if it is an answer to some CQ in q on $\mathcal{I}$.

Ontology-mediated queries. An *ontology-mediated query (OMQ)* is a triple $Q = (\mathcal{T}, \Sigma, q)$ that consists of a TBox $\mathcal{T}$, an ABox signature Σ and a query q such as a CQ or a UCQ. Let $\mathcal{A}$ be a Σ-ABox and $\mathbf{a} \in \mathsf{ind}(\mathcal{A})^{\mathsf{ar}(q)}$. We say that $\mathbf{a}$ is an *answer to Q on $\mathcal{A}$*, denoted $\mathcal{A} \models Q(\mathbf{a})$, if for every common model $\mathcal{I}$ of $\mathcal{A}$ and $\mathcal{T}$, $\mathbf{a}$ is an answer to q on $\mathcal{I}$. We may also write $\mathcal{A}, \mathcal{T} \models q(\mathbf{a})$ instead of $\mathcal{A} \models Q(\mathbf{a})$. For an ontology language $\mathcal{L}$ and query language $\mathcal{Q}$, we use $(\mathcal{L}, \mathcal{Q})$ to denote the OMQ language in which TBoxes are formulated in $\mathcal{L}$ and the actual queries are from $\mathcal{Q}$; we also identify this language with the set of all OMQs that it admits. In this paper, we mainly concentrate on the OMQ languages $(\mathcal{EL}, \text{CQ})$ and $(\mathcal{EL}, \text{AQ})$. Three examples of OMQs are given in Example 1. The OMQs Q_1 and Q_2 given there are from $(\mathcal{EL}, \text{AQ})$ while the OMQ Q_3 is from $(\mathcal{EL}, \text{CQ})$. Considering the ABox $\mathcal{A}_1$ from Figure 1, we have $\mathcal{A}_1 \models Q_2(a)$.

For an OMQ $Q = (\mathcal{T}, \Sigma, q)$, we use $\textsc{eval}(Q)$ to denote the following problem: given a Σ-ABox $\mathcal{A}$ and a tuple $\mathbf{a} \in \mathsf{ind}(\mathcal{A})^{\mathsf{ar}(q)}$, decide whether $\mathcal{A} \models Q(\mathbf{a})$.

TBox normal form. 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 and r is a role name or, in the case of $\mathcal{ELI}$-TBoxes, an inverse role. Every TBox $\mathcal{T}$ can be converted into a TBox $\mathcal{T}'$ in normal form in linear time [41], 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 $\mathcal{T}'$ by interpreting the fresh concept names. Consequently, all OMQs of the form $Q = (\mathcal{T}, \Sigma, q)$ and $Q' = (\mathcal{T}', \Sigma, q)$ 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 its data complexity nor rewritability into linear Datalog (or any other language). The TBox from Example 1 is not in normal form. It could be converted into normal form by replacing

$$\exists \mathsf{hasFather}.\mathsf{Gene1Carrier} \sqcap \exists \mathsf{hasMother}.\mathsf{Gene1Carrier} \sqsubseteq \mathsf{Gene1Carrier}$$

by the three CIs

$$\exists \mathsf{hasFather}.\mathsf{Gene1Carrier} \sqsubseteq A,\ \exists \mathsf{hasMother}.\mathsf{Gene1Carrier} \sqsubseteq B,\ A \sqcap B \sqsubseteq \mathsf{Gene1Carrier},$$

where A and B are fresh concept names.

First order Rewritability. When speaking about first-order (FO) formulas, we mean formulas without function symbols and constants that use relation symbols of arity one and two only, drawing unary relation symbols from $\mathsf{N_C}$ and binary relation symbols from $\mathsf{N_R}$. Equality is admitted. Let $Q = (\mathcal{T}, \Sigma, q) \in (\mathcal{EL}, \text{CQ})$ be an OMQ. We say that Q is *FO-rewritable* if there exists an FO formula $\varphi(\mathbf{x})$ such that for every ABox $\mathcal{A}$ and every tuple $\mathbf{a}$ of individuals from $\mathsf{ind}(\mathcal{A})$, we have $\mathcal{A} \models Q(\mathbf{a})$ if and only if $\mathcal{A} \models \varphi(\mathbf{a})$ where $\mathcal{A}$ is interpreted as a relational structure over Σ. We call $\varphi(\mathbf{x})$ an *FO-rewriting*. As an example, consider the OMQ $(\mathcal{T}, \Sigma, B(x))$ where $\mathcal{T} = \{\exists r.A \sqsubseteq B\}$ and $\Sigma = \{r, A, B\}$. This OMQ is FO-rewritable with $\varphi(x) = B(x) \vee (\exists y\, r(x, y) \wedge A(y))$ being an FO-rewriting. Also, the OMQ Q_3 from Example 1 is FO-rewritable with FO-rewriting $\varphi(x) = \exists y\, \mathsf{hasMother}(x, y)$.

Linear Datalog Rewritability. A *Datalog rule* ρ has the form $S(\mathbf{x}) \leftarrow R_1(\mathbf{y}_1) \wedge \cdots \wedge R_n(\mathbf{y}_n)$, $n > 0$, where $S, R_1, \ldots, R_n$ are relation symbols of any arity and $\mathbf{x}, \mathbf{y}_i$ denote tuples of variables that match the arity of the relation symbol that they are used with. 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 *goal relation* goal that does not occur in rule bodies. The *arity of* Π, denoted $\mathsf{ar}(\Pi)$, is the arity of the goal relation. 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 no IDB relations from Π and a tuple $\mathbf{a} \in \mathsf{ind}(\mathcal{A})^{\mathsf{ar}(\Pi)}$, we write $\mathcal{A} \models \Pi(a)$ if a is an answer to Π on $\mathcal{A}$, defined in the usual way [40]: $\mathcal{A} \models \Pi(a)$ if $\mathsf{goal}(a)$ is a logical consequence of $\mathcal{A} \cup \Pi$ viewed as a set of first-order sentences (all variables in rules quantified universally).

We also admit body atoms of the form $\top(x)$ that are vacuously true. This is just syntactic sugar since any rule with body atom $\top(x)$ can equivalently be replaced by a set of rules obtained by replacing $\top(x)$ in all possible ways with an atom $R(x_1, \dots, x_n)$ where R is an EDB relation and where $x_i = x$ for some i and all other x_i are fresh variables.

A Datalog program Π over EDB signature Σ is a *rewriting* of an OMQ $Q = (\mathcal{T}, \Sigma, q)$ if $\mathcal{A} \models Q(\mathbf{a})$ iff $\mathcal{A} \models \Pi(\mathbf{a})$ for all Σ-ABoxes $\mathcal{A}$ and all $\mathbf{a} \in \mathsf{ind}(\mathcal{A})$. 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 all OMQs from $(\mathcal{EL}, \text{CQ})$ are Datalog-rewritable. It follows from the results in this paper that there are rather simple OMQs $Q = (\mathcal{T}, \Sigma, q)$ that are not linear Datalog-rewritable, for example Q_1 from Example 1. On the other hand, Q_2 is linear Datalog-rewritable, using two unary IDBs G and goal and the following rules:

$$\begin{aligned} G(x) &\leftarrow \mathsf{Gene2Carrier}(x) \\ G(x) &\leftarrow \mathsf{hasMother}(x, y) \wedge G(y) \\ G(x) &\leftarrow \mathsf{hasFather}(x, y) \wedge G(y) \\ \mathsf{goal}(x) &\leftarrow G(x). \end{aligned}$$

Types. Let $\mathcal{T}$ be a TBox in normal form. A *$\mathcal{T}$-type t* is a set 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 an 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 t is a $\mathcal{T}$-type and a an individual name, we may use $t(a)$ we denote the ABox $\{A(a) \mid A \in t\}$.

Universal models. It is well known [42] that for every $\mathcal{ELI}$-TBox $\mathcal{T}$ and ABox $\mathcal{A}$ there is a *universal model* $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ with certain nice properties. These are summarized in the following lemma. Homomorphisms between interpretations are defined in the expected way, ignoring individual names.

Lemma 2. *Let $\mathcal{T}$ be an $\mathcal{ELI}$-TBox in normal form and $\mathcal{A}$ an ABox. Then there is an interpretation $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ such that*

1. *$\mathcal{U}_{\mathcal{A},\mathcal{T}}$ is a model of $\mathcal{A}$ and $\mathcal{T}$;*
2. *for every model $\mathcal{I}$ of $\mathcal{A}$ and $\mathcal{T}$, there is a homomorphism from $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ to $\mathcal{I}$ that is the identity on $\mathsf{ind}(\mathcal{A})$;*
3. *for all CQs q and $\mathbf{a} \in \mathsf{ind}(\mathcal{A})^{\mathsf{ar}(q)}$, $\mathcal{A}, \mathcal{T} \models q(\mathbf{a})$ iff $\mathcal{U}_{\mathcal{A},\mathcal{T}} \models q(\mathbf{a})$.*

It follows e.g. from results in [43] that $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ can be constructed using a standard chase procedure, as described next. We define a sequence of ABoxes $\mathcal{A}_0, \mathcal{A}_1, \dots$ by setting $\mathcal{A}_0 = \mathcal{A}$ and then letting $\mathcal{A}_{i+1}$ be $\mathcal{A}_i$ extended as follows:

(i) If $\exists r.B \sqsubseteq A \in \mathcal{T}$ and $r(a,b), B(b) \in \mathcal{A}_i$, then add $A(a)$ to $\mathcal{A}_{i+1}$;

(ii) If $\exists r^-.A \sqsubseteq B \in \mathcal{T}$ and $r(a,b), A(a) \in \mathcal{A}_i$, then add $B(b)$ to $\mathcal{A}_{i+1}$;

(iii) if $\top \sqsubseteq A \in \mathcal{T}$ and $a \in \mathsf{ind}(\mathcal{A}_i)$, then add $A(a)$ to $\mathcal{A}_{i+1}$;

(iv) if $B_1 \sqcap B_2 \sqsubseteq A \in \mathcal{T}$ and $B_1(a), B_2(a) \in \mathcal{A}_i$, then add $A(a)$ to $\mathcal{A}_{i+1}$;

(v) if $A \sqsubseteq \exists r.B \in \mathcal{T}$, $A(a) \in \mathcal{A}_i$ and there is no $b \in \mathsf{ind}(\mathcal{A}_i)$ such that $r(a,b)$ and $B(b)$, then take a fresh individual b and add $r(a,b)$ and $B(b)$ to $\mathcal{A}_{i+1}$;

(vi) if $B \sqsubseteq \exists r^-.A \in \mathcal{T}$, $B(b) \in \mathcal{A}_i$ and there is no $a \in \mathsf{ind}(\mathcal{A}_i)$ such that $r(a,b)$ and $A(a)$, then take a fresh individual a and add $r(a,b)$ and $A(a)$ to $\mathcal{A}_{i+1}$.

Let $\mathcal{A}_\omega = \bigcup_{i \geq 0} \mathcal{A}_i$. We define $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ to be the interpretation that corresponds to $\mathcal{A}_\omega$. The properties in Lemma 2 are standard to prove, see for example [43, 44].

Note that $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ can be infinite and that its shape is basically the shape of $\mathcal{A}$, but with a (potentially infinite) tree attached to every individual in $\mathcal{A}$. The domain elements in these trees are introduced by Rules (v) and (vi), and we refer to them as *anonymous elements*. It should thus also be clear what we mean when speaking about the anonymous elements in $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ *below* some $a \in \mathsf{ind}(\mathcal{A})$: the restriction of $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ to a and those anonymous individuals that can be reached from a without passing through an ABox individual. The following is an immediate consequence of the definition of universal models. We omit a formal proof.

Lemma 3. *Let $\mathcal{T}$ be a TBox in normal form and let $\mathcal{A}_1$ and $\mathcal{A}_2$ be ABoxes and $a_i \in \mathsf{ind}(\mathcal{A}_i)$ for $i \in \{1,2\}$. If $\mathsf{tp}_{\mathcal{A}_1,\mathcal{T}}(a_1) = \mathsf{tp}_{\mathcal{A}_2,\mathcal{T}}(a_2)$, then the restriction of $\mathcal{U}_{\mathcal{A}_1,\mathcal{T}}$ to a_1 and the anonymous elements below it is homomorphically equivalent to the restriction of $\mathcal{U}_{\mathcal{A}_2,\mathcal{T}}$ to a_2 and the anonymous elements below it.*

Pathwidth. A *path decomposition* of a (directed or undirected) graph $G = (V,E)$ is a sequence $V_1, \dots, V_n$ of subsets of V, such that

- $V_i \cap V_k \subseteq V_j$ for $1 \leq i \leq j \leq k \leq n$ and
- for every edge $e \in E$ there exists an i such that $e \subseteq V_i$.

A path decomposition $V_1, \dots, V_n$ is an (ℓ, k)-*path decomposition* if $\ell = \max\{|V_i \cap V_{i+1}| \mid 1 \leq i < n\}$ and $k = \max\{|V_i| \mid 1 \leq i \leq n\}$. The *pathwidth* of G, denoted $\mathsf{pw}(G)$, is the smallest integer k, such that G has a $(\ell, k+1)$-path decomposition for some ℓ. For an ABox $\mathcal{A}$, a sequence $V_1, \dots, V_n$ of subsets of $\mathsf{ind}(\mathcal{A})$ is a path decomposition of $\mathcal{A}$ if $V_1, \dots, V_n$ is a path decomposition of $G_\mathcal{A}$. We assign a pathwidth to $\mathcal{A}$ by setting $\mathsf{pw}(\mathcal{A}) := \mathsf{pw}(G_\mathcal{A})$.

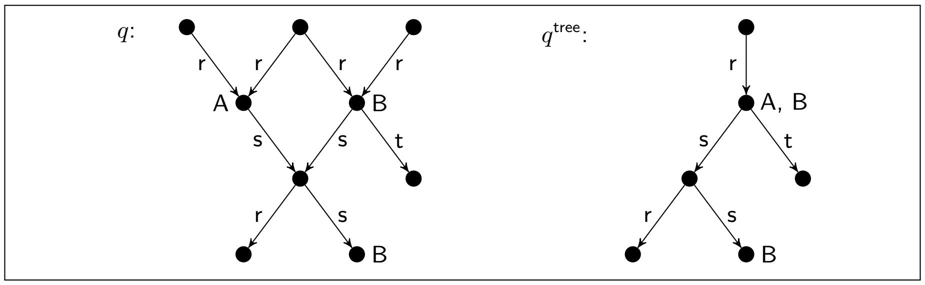

Figure 2: Treeifiable CQ q and tree-shaped CQ q^{tree} resulting from fork elimination. The latter can be viewed as the $\mathcal{EL}$-concept $C_{q^{\mathsf{tree}}} = \exists r.(A \sqcap B \sqcap \exists t.\top \sqcap \exists s.(\exists r.\top \sqcap \exists s.B))$.

Treeifying CQs. A Boolean CQ q is *treeifiable* if there exists a homomorphism from q into a tree-shaped interpretation. With every treeifiable Boolean CQ q, we associate a tree-shaped CQ q^{tree} that is obtained by starting with q and then exhaustively *eliminating forks*, that is, identifying x_1 and x_2 whenever there are atoms $r(x_1, y)$ and $r(x_2, y)$. Informally, one should think of q^{tree} as the least constrained treeification of q. It is known that a Boolean CQ q is treeifiable if and only if the result of exhaustively eliminating forks is tree-shaped [45]. Consequently, it can be decided in polynomial time whether a Boolean CQ is treeifiable. Treeification is useful because every tree-shaped Boolean CQ q can be viewed as an $\mathcal{EL}$-concept C_q in a straightforward way. Figure 2 shows a treeifiable CQ, the tree-shaped CQ resulting from fork elimination, and the corresponding $\mathcal{EL}$-concept.

A pair of variables (x, y) from a CQ q is *guarded* if q contains an atom of the form $r(x, y)$. For every guarded pair (x, y) and every $i \geq 0$, define $\mathsf{reach}^i(x, y)$ to be the smallest set such that

1. $x \in \mathsf{reach}^0(x, y)$ and $y \in \mathsf{reach}^1(x, y)$;
2. if $z \in \mathsf{reach}^i(x, y)$, $i > 0$, and $r(z, u) \in q$, then $u \in \mathsf{reach}^{i+1}(x, y)$;
3. if $u \in \mathsf{reach}^{i+1}(x, y)$, $i > 0$ and $r(z, u) \in q$, then $z \in \mathsf{reach}^i(x, y)$.

Moreover, $\mathsf{reach}(x, y) = \bigcup_i \mathsf{reach}^i(x, y)$. We use $\mathsf{trees}(q)$ to denote the set of all (tree-shaped) CQs p^{tree} such that $p = q|_{\mathsf{reach}(x,y)}$ for some guarded pair (x, y) with p treeifiable.

It is easy to verify that the number of CQs in $\mathsf{trees}(q)$ is linear in $|q|$. We briefly argue that $\mathsf{trees}(q)$ can be computed in polynomial time. The number of guarded pairs is linear in $|q|$. For each guarded pair (x, y), $\mathsf{reach}(x, y)$ can clearly be computed in polynomial time. Moreover, exhaustively eliminating forks on $p = q|_{\mathsf{reach}(x,y)}$ takes only polynomial time, which tells us whether p is treeifiable and constructs p^{tree} if this is the case.

Pseudo tree-shaped ABoxes. Throughout the paper, we often concentrate on ABoxes that take a restricted, almost tree-shaped form. These are called pseudo tree-shaped ABoxes, first introduced in [18]. An ABox $\mathcal{A}$ is *pseudo tree-shaped with core individuals* $I \subseteq \mathsf{ind}(\mathcal{A})$ if removing from $\mathcal{A}$ all role assertions

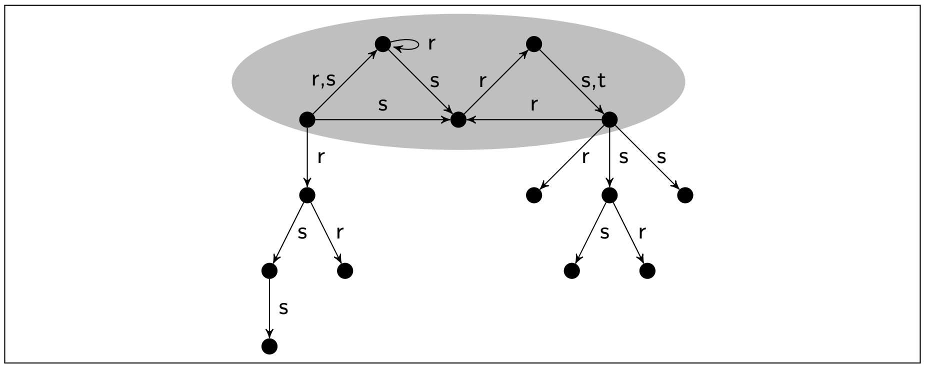

Figure 3: A pseudo tree-shaped ABox with core size 5 and 2 trees. Inside the core (highlighted in gray), cycles, multi-edges and self-loops are allowed. In the trees, there are no multi-edges, no self-loops and all roles are directed away from the core.

that only use individuals from I yields a disjoint union of tree-shaped ABoxes. We refer to these as the *trees of* $\mathcal{A}$, to $\mathcal{A}|_I$ as the *core* of $\mathcal{A}$, and to the number of individuals in I as the *core size* of $\mathcal{A}$. Note that every tree-shaped ABox is pseudo tree-shaped with core size 1 and a single tree, I is then the root of the tree. Figure 3 shows a pseudo tree-shaped ABox. Also note that every ABox $\mathcal{A}$ is pseudo tree-shaped with core individuals $\mathsf{ind}(\mathcal{A})$, but we are usually interested in the case where I is a proper subset of $\mathsf{ind}(\mathcal{A})$.

We introduce some succinct notation for removing parts of pseudo tree-shaped ABox that is used throughout the paper. For every pseudo tree-shaped ABox $\mathcal{A}$ and a non-core individual $a \in \mathsf{ind}(\mathcal{A})$, we use $\mathcal{A}^a$ to denote the tree-shaped ABox rooted at a. Moreover, we use $\mathcal{A}_a$ to denote the pseudo tree-shaped ABox $\mathcal{A} \setminus \mathcal{A}^a$, 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 $((\mathcal{A}^a)_b)_c$.

The following lemma describes the central property of pseudo tree-shaped ABoxes. It essentially says that if $\mathbf{a}$ is an answer to an OMQ Q based on a connected CQ q on an ABox $\mathcal{A}$, then one can unravel $\mathcal{A}$ into a pseudo tree-shaped ABox $\mathcal{A}'$ of core size at most $|q|$ that homomorphically maps to $\mathcal{A}$ and such that $\mathbf{a}$ is an answer to Q on $\mathcal{A}'$, witnessed by a homomorphism from q to $\mathcal{U}_{\mathcal{A}',\mathcal{T}}$ that satisfies the additional property of being within or at least 'close to' the core of $\mathcal{A}'$.

Lemma 4. *Let $Q = (\mathcal{T}, \Sigma, q) \in (\mathcal{EL}, \text{conCQ})$, $\mathcal{A}$ a Σ-ABox and $\mathbf{a} \in \mathsf{ind}(\mathcal{A})^{\mathsf{ar}(q)}$ such that $\mathcal{A} \models Q(\mathbf{a})$. Then there is a pseudo tree-shaped Σ-ABox $\mathcal{A}'$ of core size at most $|q|$ and with $\mathbf{a}$ in its core that satisfies the following conditions:*

1. *there is a homomorphism from $\mathcal{A}'$ to $\mathcal{A}$ that is the identity on $\mathbf{a}$;*
2. *$\mathcal{A}' \models Q(\mathbf{a})$, witnessed by a homomorphism from q to $\mathcal{U}_{\mathcal{A}',\mathcal{T}}$ whose range consists solely of core individuals and of anonymous elements in a tree rooted in a core individual.*

We shall often be interested in pseudo tree-shaped ABoxes $\mathcal{A}$ that give an answer $\mathbf{a}$ to an OMQ Q and that are minimal with this property regarding set inclusion, that is, no strict subset of $\mathcal{A}$ supports $\mathbf{a}$ as an answer to Q. We introduce some convenient notation for this. Let $Q = (\mathcal{T}, \Sigma, q) \in (\mathcal{EL}, \mathrm{CQ})$. We use $\mathcal{M}_Q$ to denote the set of all pseudo tree-shaped Σ-ABoxes $\mathcal{A}$ of core size at most $|q|$ such that for some tuple $\mathbf{a}$ in the core of $\mathcal{A}$, $\mathcal{A} \models Q(\mathbf{a})$ while $\mathcal{A}' \not\models Q(\mathbf{a})$ for any $\mathcal{A}' \subsetneq \mathcal{A}$.

Finally, we note that when the TBox is formulated in $\mathcal{EL}$, then the concept names derived at some non-core individual a of a pseudo tree-shaped ABox only depend on $\mathcal{A}^a$.[2] The proof is by a straightforward analysis of the construction of universal models.

Lemma 5. *Let $\mathcal{T}$ be an $\mathcal{EL}$-TBox, $\mathcal{A}$ an pseudo tree-shaped ABox, and $a \in \mathsf{ind}(\mathcal{A})$ a non-core individual. Then $\mathcal{A}, \mathcal{T} \models B(a)$ if and only if $\mathcal{A}^a, \mathcal{T} \models B(a)$ for every concept name $B \in \mathsf{N_C}$.*

Fundamental ABox Manipulations. 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. We state it only for $(\mathcal{EL}, \mathrm{AQ})$, since we only use it for this OMQ language. The proof is by a careful removal of role assertions that are not required for the construction of the universal model.

Lemma 6. *Let $Q = (\mathcal{T}, \Sigma, A(x)) \in (\mathcal{EL}, \mathrm{AQ})$ be an OMQ and $\mathcal{A}$ a Σ-ABox such that $\mathcal{A} \models Q(a)$. Then there exists $\mathcal{A}' \subseteq \mathcal{A}$ of degree at most $|\mathcal{T}|$ such that $\mathcal{A}' \models Q(a)$.*

We next introduce a fundamental construction for merging ABoxes. Let $\mathcal{T}$ be an $\mathcal{ELI}$-TBox. The following lemma allows us to glue together ABoxes under certain conditions or to replace a subset of an ABox by another ABox without changing the types that are derived. The proof is by careful manipulation of models.

Lemma 7. *Let $\mathcal{A}_1, \mathcal{A}_2$ be Σ-ABoxes and $\mathcal{T}$ an $\mathcal{ELI}$-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 $a \in \mathsf{ind}(\mathcal{A}_i)$, $i \in \{1, 2\}$.*

Throughout the paper, we shall often replace a part of an ABox by a different ABox. Let $\mathcal{A}$ be a pseudo tree-shaped ABox, $b \in \mathsf{ind}(\mathcal{A})$ a non-core individual and $\mathcal{A}'$ a tree-shaped ABox with root b and $\mathsf{ind}(\mathcal{A}) \cap \mathsf{ind}(\mathcal{A}') = \{b\}$. When we speak about the ABox obtained from $\mathcal{A}$ *by replacing the subtree below b by $\mathcal{A}'$*, we mean the ABox that can be obtained as follows: Remove all concept assertions of the form $A(b)$ and all assertions that involve a descendant of b from $\mathcal{A}$, and then take the ABox $\mathcal{A} \cup \mathcal{A}'$. The following corollary allows us to replace parts of pseudo tree-shaped ABoxes.

[2]The result generalizes to ABoxes that are not pseudo tree-shaped, but for our purposes the given formulation suffices.

Corollary 8. *Let $\mathcal{T}$ be an $\mathcal{ELI}$-TBox, $\mathcal{A}$ a pseudo tree-shaped ABox, $b \in \mathsf{ind}(\mathcal{A})$ not in the core of $\mathcal{A}$, and $\mathcal{A}'$ a tree-shaped ABox with root b such that $\mathsf{ind}(\mathcal{A}) \cap \mathsf{ind}(\mathcal{A}') = \{b\}$ and $\mathsf{tp}_{\mathcal{A}',\mathcal{T}}(b) = \mathsf{tp}_{\mathcal{A},\mathcal{T}}(b)$. If $\mathcal{A}''$ is the ABox obtained from $\mathcal{A}$ by replacing the subtree rooted at b by $\mathcal{A}'$, then $\mathsf{tp}_{\mathcal{A},\mathcal{T}}(a) = \mathsf{tp}_{\mathcal{A}'',\mathcal{T}}(a)$ for all $a \in \mathsf{ind}(\mathcal{A}) \cap \mathsf{ind}(\mathcal{A}'')$.*

3. AC^0 versus NL for Connected CQs

We prove a dichotomy between AC^0 and NL for $(\mathcal{EL}, \mathrm{conCQ})$ and show that for OMQs from this language, evaluation in AC^0 coincides with FO rewritability. The dichotomy does not depend on assumptions from complexity theory since it is known that $\mathrm{AC}^0 \neq \mathrm{NL}$ [46]. We generalize the results obtained here to potentially disconnected CQs in Section 5.

FO-rewritability of OMQs in $(\mathcal{EL}, \mathrm{CQ})$ has been characterized in [18] by a property called bounded depth. Informally, an OMQ Q has bounded depth if it looks only boundedly far into the ABox. To obtain our results, we show that unbounded depth implies NL-hardness. Formally, bounded depth is defined as follows. The *depth* of a tree-shaped ABox $\mathcal{A}$ is the largest number k such that there exists a path of length k starting from the root in $G_\mathcal{A}$. The *depth* of a pseudo tree-shaped ABox is the maximum depth of its trees. We say that an OMQ $Q \in (\mathcal{EL}, \mathrm{CQ})$ has *bounded depth* if there is a k such that every $\mathcal{A} \in \mathcal{M}_Q$ has depth at most k. If there is no such k, then Q has *unbounded depth.*

In Example 1, the OMQ Q_3 has bounded depth, while Q_1 and Q_2 have unbounded depth. Unbounded depth of Q_2 is witnessed by the ABoxes

$$\mathcal{A}_k = \{\mathsf{hasMother}(a_i, a_{i+1}) \mid 0 \leq i < k\} \cup \{\mathsf{Gene2Carrier}(a_k)\}$$

with $k \geq 1$. It is not hard to verify that $\mathcal{A}_k \in \mathcal{M}_{Q_2}$ for every $k \geq 1$ and that the depth of $\mathcal{A}_k$ is k. The following is the main result of this section.

Theorem 9. *Let $Q \in (\mathcal{EL}, \mathrm{conCQ})$. The following are equivalent:*

(i) Q has bounded depth.

(ii) Q is FO-rewritable.

(iii) $\textsc{eval}(Q)$ is in AC^0.

If these conditions do not hold, then $\textsc{eval}(Q)$ is NL*-hard under FO reductions.*

The equivalence (ii) $\Leftrightarrow$ (iii) is closely related to a result in CSP. In fact, every OMQ of the form $(\mathcal{T}, \Sigma, \exists x\, A(x))$ with A a concept name and $\mathcal{T}$ formulated in $\mathcal{ELI}$ is equivalent to the complement of a CSP [3] and it is a known result in CSP that FO-rewritability coincides with AC^0 [47]. Conjunctive queries, however, go beyond the expressive power of (complements of) CSPs and thus we cannot derive (ii) $\Leftrightarrow$ (iii) from the CSP connection.

The equivalence (i) $\Leftrightarrow$ (ii) follows from Theorem 9 in [18]. Further, the implication (ii) $\Rightarrow$ (iii) is clear because first order formulas can be evaluated in

AC^0. It thus suffices to show the implication (iii) $\Rightarrow$ (i) and the last sentence of the theorem. We prove that unbounded depth implies NL-hardness, which establishes both since $\mathrm{AC}^0 \neq \mathrm{NL}$.

We first give a rough sketch of the proof. Let $Q = (\mathcal{T}, \Sigma, q) \in (\mathcal{EL}, \mathrm{conCQ})$ be an OMQ of unbounded depth. We reduce REACH, the reachability problem in directed graphs, to EVAL(Q). Note that REACH is NL-complete under FO reductions [48]. An input to REACH is a tuple $G = (V, E, s, t)$ where (V, E) is a directed graph, $s \in V$ a *source node* and $t \in V$ a *target node*. Such a tuple is a yes-instance if there exists a path from s to t in the graph (V, E). To simplify the reduction, we further assume w.l.o.g. that $s \neq t$, that the indegree of s and the outdegree of t are both 0, and that there are no nodes without incident edges.

The reduction has to translate a tuple $G = (V, E, s, t)$ into a Σ-ABox $\mathcal{A}_G$ and a tuple $\mathbf{a}$ such that $\mathcal{A}_G \models Q(\mathbf{a})$ if and only if there is a path in G from s to t. We show that any ABox from $\mathcal{M}_Q$ of sufficiently large depth can be used to construct ABoxes $\mathcal{A}_{\mathsf{source}}$, $\mathcal{A}_{\mathsf{edge}}$ and $\mathcal{A}_{\mathsf{target}}$ that can serve as gadgets in the reduction. More precisely, the ABox $\mathcal{A}_G$ has (among others) one individual a_v for every node $v \in V$, the edges of (V, E) will be represented using copies of $\mathcal{A}_{\mathsf{edge}}$, and the source and target nodes will be marked using the ABoxes $\mathcal{A}_{\mathsf{source}}$ and $\mathcal{A}_{\mathsf{target}}$, respectively. The bottom half of Figure 5 illustrates how $\mathcal{A}_G$ is constructed from G using the gadgets. As part of the reduction, we identify two $\mathcal{T}$-types t_0 and t_1 such that for every node $v \in V$, $\mathsf{tp}_{\mathcal{A}_G,\mathcal{T}}(a_v) = t_1$ if v is reachable from s in G and $\mathsf{tp}_{\mathcal{A}_G,\mathcal{T}}(a_v) = t_0$ otherwise. The tuple $\mathbf{a}$ is then chosen such that $\mathcal{A}_G, \mathcal{T} \models q(\mathbf{a})$ if and only if $\mathsf{tp}_{\mathcal{A}_G,\mathcal{T}}(a_t) = t_1$. The purpose of $\mathcal{A}_{\mathsf{source}}$ is to produce the type t_1, the purpose of $\mathcal{A}_{\mathsf{edge}}$ is to propagate the type further, and the purpose of $\mathcal{A}_{\mathsf{target}}$ is to make the query true whenever it receives t_1 as 'input'.

We next define a property of Q, called the *ability to simulate* REACH, that makes the properties of $\mathcal{A}_{\mathsf{source}}$, $\mathcal{A}_{\mathsf{edge}}$, and $\mathcal{A}_{\mathsf{target}}$ precise, as well as those of the $\mathcal{T}$-types t_0 and t_1. We then show that Q having unbounded depth implies the ability to simulate REACH and that this, in turn, implies NL-hardness of EVAL(Q) via a reduction from REACH.

For the rest of this section, we assume w.l.o.g. that in any OMQ $Q = (\mathcal{T}, \Sigma, q) \in (\mathcal{EL}, \mathrm{conCQ})$, the TBox $\mathcal{T}$ has been modified as follows: for every $p \in \mathsf{trees}(q)$, introduce a fresh concept name A_p and add the concept inclusion $C_p \sqsubseteq A_p$ to $\mathcal{T}$ where C_p is p viewed as an $\mathcal{EL}$-concept. Then normalize $\mathcal{T}$ again. It is easy to see that the OMQ resulting from this modification is equivalent to the original OMQ Q. The extension is still useful since its types are more informative, now potentially containing also the freshly introduced concept names.

In the reduction, Boolean queries require some special care since they can be made true by homomorphisms to anywhere in the universal model of $\mathcal{A}_G$ and $\mathcal{T}$, rather than only to the neighborhood of the answer tuple $\mathbf{a}$ (recall that we work with connected CQs). To address this issue, we make use of what we call core close homomorphisms. Let $\mathcal{A}$ be a pseudo tree-shaped Σ-ABox and $\mathbf{a}$ a tuple from $\mathsf{ind}(\mathcal{A})$. We call a homomorphism h from q to $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ *core close* if there is some variable x in q such that $h(x)$ is a core individual or an anonymous element in a tree that the chase has generated below a core individual. If $\mathsf{ar}(q) > 0$ and $\mathbf{a}$ is from the core of $\mathcal{A}$, then every homomorphism from q to $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ must be core close. Boolean CQs q, in contrast, may admit homomorphisms to $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ that are not core close. However, the following lemma shows that this cannot happen when $\mathcal{A} \in \mathcal{M}_Q$.

Lemma 10. *Let $Q = (\mathcal{T}, \Sigma, q) \in (\mathcal{EL}, \text{conCQ})$ be Boolean and $\mathcal{A} \in \mathcal{M}_Q$. Then every homomorphism from q to $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ is core close.*

Proof. Since $\mathcal{A} \in \mathcal{M}_Q$, $\mathcal{A}$ is minimal with the property that $\mathcal{A} \models Q$. Assume that there is a homomorphism h from q to $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ that is not core close. Then there is no path in $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ from any element in the range of h to any individual in the core of $\mathcal{A}$ (though a path in the converse direction might exist). By Lemma 5, we can remove all assertions in $\mathcal{A}$ that involve a core individual and the resulting ABox $\mathcal{A}'$ satisfies $\mathcal{A}' \models Q$, contradicting the minimality of $\mathcal{A}$. $\square$

If M is a set of concept names (such as a type), then $M(a)$ denotes the ABox $\{A(a) \mid A \in M\}$. We write $\mathcal{A}, \mathcal{T} \models M(a)$ to mean that $\mathcal{A}, \mathcal{T} \models A(a)$ for all $A \in M$. We are now ready to define formally the ability to simulate REACH.

Definition 11. An OMQ $Q = (\mathcal{T}, \Sigma, q) \in (\mathcal{EL}, \text{conCQ})$ has *the ability to simulate* REACH if there exist

- a pseudo tree-shaped Σ-ABox $\mathcal{A}$ of core size at most $|q|$,
- a tuple $\mathbf{a}$ of individuals from the core of $\mathcal{A}$ of length $\mathsf{ar}(q)$,
- a tree $\mathcal{A}_i$ of $\mathcal{A}$ with two distinguished individuals b, c such that b has distance at least $|q|$ from the core and c is a descendant of b that has distance at least $|q|$ from b and
- $\mathcal{T}$-types $t_0 \subsetneq t_1$

such that

1. $\mathcal{A} \models Q(\mathbf{a})$,
2. $t_1 = \mathsf{tp}_{\mathcal{A},\mathcal{T}}(b) = \mathsf{tp}_{\mathcal{A},\mathcal{T}}(c)$,
3. $\mathsf{tp}_{\mathcal{A}_c \cup t_0(c),\mathcal{T}}(b) = t_0$,
4. $\mathcal{A}_b \cup t_0(b) \not\models Q(\mathbf{a})$ and
5. if q is Boolean, then every homomorphism h from q to $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ is core close.

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

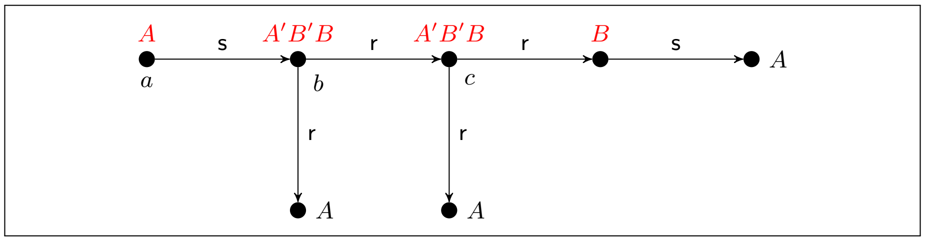

Figure 4: The ABox $\mathcal{A}$ displayed in black witnesses that the OMQ from Example 12 has the ability to simulate REACH. The assertions in red are derived from $\mathcal{A}$ using $\mathcal{T}$, showing that $t_1 = \mathsf{tp}_{\mathcal{A},\mathcal{T}}(b) = \mathsf{tp}_{\mathcal{A},\mathcal{T}}(c)$.

To understand the essence of Definition 11, it is worthwhile to consider the special case where q is an AQ $A(x)$. In this case, Q has the ability to simulate REACH if there is a tree-shaped Σ-ABox $\mathcal{A}$ with root $a = \mathbf{a}$, two distinguished non-root individuals $b, c \in \mathsf{ind}(\mathcal{A})$, c a descendant of b, and $\mathcal{T}$-types $t_0 \subsetneq t_1$ such that Conditions (1)-(4) of Definition 11 are satisfied. All remaining parts of Definition 11 should be thought of as technical complications induced by replacing AQs with CQs. Conditions (1)-(4) make sure that the gadgets $\mathcal{A}_{\mathsf{source}}$, $\mathcal{A}_{\mathsf{edge}}$ and $\mathcal{A}_{\mathsf{target}}$ behave in the expected way. In particular, Condition (2) says that $\mathcal{A}_{\mathsf{source}}$ produces the type t_1 and that $\mathcal{A}_{\mathsf{edge}}$ propagates that type further. Condition (3) says that if the 'input' for $\mathcal{A}_{\mathsf{edge}}$ is t_0, then the 'output' will also be t_0. Condition (4) says that the type t_0 as input for $\mathcal{A}_{\mathsf{target}}$ is not sufficient to imply the query. The top half of Figure 5 shows schematically how the ABoxes $\mathcal{A}_{\mathsf{source}}$, $\mathcal{A}_{\mathsf{edge}}$ and $\mathcal{A}_{\mathsf{target}}$ are defined from an ABox that has the ability to simulate REACH. In the following example, we give a concrete OMQ that has the ability to simulate REACH and an ABox witnessing this.

Example 12. Let

$$\mathcal{T} = \{\exists s.A \sqsubseteq B, \exists r.A \sqsubseteq A', \exists r.B \sqsubseteq B', A' \sqcap B' \sqsubseteq B, \exists s.B \sqsubseteq A\}$$

and $\Sigma = \{r, s, A\}$. Figure 4 shows a tree-shaped ABox that witnesses that the OMQ $(\mathcal{T}, \Sigma, A(x))$ has the ability to simulate REACH, via types $t_1 = \{A', B', B\}$ and $t_0 = \{A'\}$, and with $\mathbf{a} = a$.

In Proposition 13 below, we show that unbounded depth implies the ability to simulate REACH and in Proposition 14 we show that the ability to simulate REACH enables a reduction from the reachability problem for directed graphs.

Proposition 13. *Let $Q \in (\mathcal{EL}, \text{conCQ})$. If Q has unbounded depth, then Q has the ability to simulate* REACH.

Proof. We use a pumping argument. Let $Q = (\mathcal{T}, \Sigma, q) \in (\mathcal{EL}, \text{conCQ})$ have unbounded depth. There must be a pseudo tree-shaped ABox $\mathcal{A} \in \mathcal{M}_Q$ and a tuple $\mathbf{a}$ from its core such that $\mathcal{A} \models Q(\mathbf{a})$ and such that one of its trees, say $\mathcal{A}_i$, has depth at least $k := (|q|+1) \cdot 3^{|\mathcal{T}|} + |q| + 1$. Consider a path of length at least k from the root of $\mathcal{A}_i$ to a leaf. Let $\mathcal{A}'$ denote the ABox obtained from $\mathcal{A}$ by

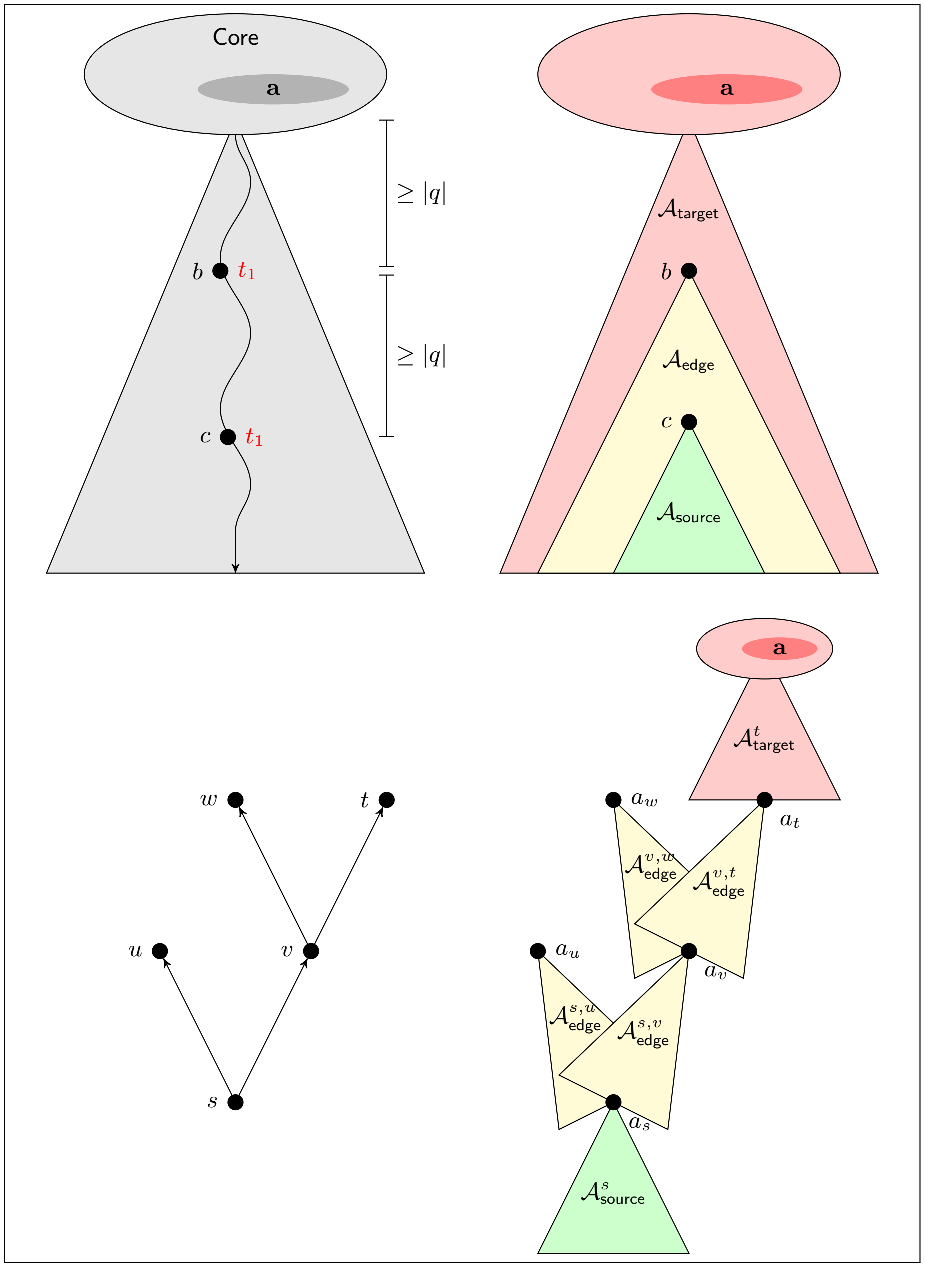
Core
$\mathbf{a}$
$\geq |q|$
$\geq |q|$
b
t_1
c
t_1
$\mathbf{a}$
$\mathcal{A}_{\mathsf{target}}$
b
$\mathcal{A}_{\mathsf{edge}}$
c
$\mathcal{A}_{\mathsf{source}}$
w
t
u
v
s
$\mathbf{a}$
$\mathcal{A}^t_{\mathsf{target}}$
a_w
a_t
$\mathcal{A}^{v,w}_{\mathsf{edge}}$
$\mathcal{A}^{v,t}_{\mathsf{edge}}$
a_u
a_v
$\mathcal{A}^{s,u}_{\mathsf{edge}}$
$\mathcal{A}^{s,v}_{\mathsf{edge}}$
a_s
$\mathcal{A}^s_{\mathsf{source}}$

Figure 5: Top left: Schematically, an ABox that has the ability to simulate REACH. Top right: The same ABox, divided into $\mathcal{A}_{\mathsf{source}}$, $\mathcal{A}_{\mathsf{edge}}$ and $\mathcal{A}_{\mathsf{target}}$. Bottom: In the reduction, an instance G for REACH (bottom left) is translated into the ABox $\mathcal{A}_G$ (bottom right).

removing all assertions that involve the leaf in this path. Since $\mathcal{A}$ is minimal, $\mathcal{A}' \not\models Q(\mathbf{a})$. Now, every individual b on the remaining path that has distance at least $|q|$ from the core is colored with 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 $3^{|\mathcal{T}|}$ as an upper bound for the number of different colors (t'_b, t_b) that may occur on the path. But the number of individuals on this path with distance at least $|q|$ from the core is at least $k - |q| = (|q|+1)\cdot 3^{|\mathcal{T}|}+1$, so by the pigeonhole principle there is one color (t', t) that appears at least $|q| + 1$ times on the path. Then there must be distinct individuals b and c that have distance at least $|q|$ from each other and such that $(t'_b, t_b) = (t'_c, t_c)$. W.l.o.g. let c be a descendant of b. We set $t_0 = t'_b$ and $t_1 = t_b$.

For this choice of $\mathcal{A}$, $\mathbf{a}$, b, c, t_0 and t_1, Conditions 1 and 2 from Definition 11 are immediately clear. With Corollary 8, we can replace $\mathcal{A}^c$ in $\mathcal{A}'$ by $t_0(c)$, so Condition 3 holds. Furthermore, we have $\mathcal{A}' \not\models Q(\mathbf{a})$ and $\mathsf{tp}_{\mathcal{A}',\mathcal{T}}(b) = t_0$, so again by Corollary 8, if we replace $\mathcal{A}^b$ with $t_0(b)$, the types derived in the remaining ABox do not change, thus Condition 4 holds. Condition 5 follows from Lemma 10. $\square$

Now for the reduction from REACH to EVAL(Q) when Q has the ability to simulate REACH.

Proposition 14. *Let $Q \in (\mathcal{EL}, \text{conCQ})$. If Q has the ability to simulate* REACH*, then* EVAL(Q) *is* NL*-hard under FO reductions.*

To prove Proposition 14, let $Q = (\mathcal{T}, \Sigma, q) \in (\mathcal{EL}, \text{conCQ})$ have the ability to simulate REACH. Then there is a pseudo tree-shaped ABox $\mathcal{A}$, a tuple $\mathbf{a}$ in its core, distinguished individuals b and c, and types $t_0 \subsetneq t_1$ as in Definition 11. Fix such $\mathcal{A}$, $\mathbf{a}$, b, c, t_0, and t_1 for what follows. This gives rise to ABoxes $\mathcal{A}_{\mathsf{source}}$, $\mathcal{A}_{\mathsf{target}}$, and $\mathcal{A}_{\mathsf{edge}}$ as in Definition 11. We reduce REACH to EVAL(Q).

Let $G = (V, E, s, t)$ be an input tuple for REACH. We use (copies of) $\mathcal{A}_{\mathsf{source}}$, $\mathcal{A}_{\mathsf{target}}$, and $\mathcal{A}_{\mathsf{edge}}$ as building blocks to construct a Σ-ABox $\mathcal{A}_G$ with a designated tuple of individual names $\mathbf{a}$ such that $\mathcal{A}_G \models Q(\mathbf{a})$ if and only if there is a path in G from s to t. We reserve an individual a_v for every node $v \in V$, to be used in $\mathcal{A}_G$ along with additional individuals. We define

$$\mathcal{A}_G := \mathcal{A}^s_{\mathsf{source}} \cup \mathcal{A}^t_{\mathsf{target}} \cup \bigcup_{(u,v)\in E} \mathcal{A}^{u,v}_{\mathsf{edge}}$$

where

- $\mathcal{A}^s_{\mathsf{source}}$ is a copy of $\mathcal{A}_{\mathsf{source}}$ where c is renamed to a_s and all other individuals are fresh;
- $\mathcal{A}^t_{\mathsf{target}}$ is a copy of $\mathcal{A}_{\mathsf{target}}$ where b is renamed to a_t, all individuals from $\mathbf{a}$ retain their original name, and all other individuals are fresh;[3]

[3]As a consequence, all individuals from $\mathbf{a}$ are part of $\mathcal{A}_G$.

- for every $(u,v) \in E$, $\mathcal{A}^{u,v}_{\mathsf{edge}}$ is a copy of $\mathcal{A}_{\mathsf{edge}}$ where c is renamed to a_u, b is renamed to a_v and all other individuals are fresh.

The bottom half of Figure 5 visualizes this construction.

It can be verified that $\mathcal{A}_G$ can be constructed from G using an FO reduction. For such a reduction one views both $\mathcal{A}_G$ and G as an FO structure and then constructs $\mathcal{A}_G$ from G using one FO query to define the domain and one FO query for each relation symbol to define the extension of that symbol; see [48] for more information. It is very tedious to describe FO reductions in full formal detail, so we avoid that here. We note, however, that the kind of reduction that we use here, where edges in some graph are replaced with gadgets of constant size, are a standard example of FO reductions. We refer the interested reader to Example 2.19 in [48] where an FO reduction of SAT to CLIQUE is presented in full detail.

It remains to show the correctness of the reduction.

Lemma 15. *t is reachable from s in G if and only if $\mathcal{A}_G \models Q(\mathbf{a})$.*

The proof of Lemma 15 relies on various technical properties of the ABoxes $\mathcal{A}_{\mathsf{source}}$, $\mathcal{A}_{\mathsf{target}}$, and $\mathcal{A}_{\mathsf{edge}}$ that we summarize next.

Lemma 16. *Let $Q = (\mathcal{T}, \Sigma, q) \in (\mathcal{EL}, \text{conCQ})$ have the ability to simulate* REACH *and let $\mathcal{A}$, $\mathbf{a}$, b, c, t_0, t_1, $\mathcal{A}_{\mathsf{source}}$, $\mathcal{A}_{\mathsf{edge}}$ and $\mathcal{A}_{\mathsf{target}}$ be as in Definition 11. Then*

1. $\mathsf{tp}_{\mathcal{A}_{\mathsf{source}},\mathcal{T}}(c) = t_1$;
2. $\mathsf{tp}_{\mathcal{A}_{\mathsf{edge}} \cup t_1(c),\mathcal{T}}(b) = t_1$;
3. $\mathcal{A}_{\mathsf{target}} \cup t_1(b), \mathcal{T} \models Q(\mathbf{a})$.
4. $\mathsf{tp}_{\mathcal{A}_{\mathsf{source}} \cup t_1(c),\mathcal{T}}(c) = t_1$;
5. $\mathsf{tp}_{\mathcal{A}_{\mathsf{edge}} \cup t_i(b) \cup t_j(c),\mathcal{T}}(c) = t_j$ *for all* $i,j \in \{0,1\}$;
6. $\mathsf{tp}_{\mathcal{A}_{\mathsf{edge}} \cup t_i(b) \cup t_j(c),\mathcal{T}}(b) = t_{\max\{i,j\}}$ *for all* $i,j \in \{0,1\}$;
7. $\mathsf{tp}_{\mathcal{A}_{\mathsf{target}} \cup t_0(b),\mathcal{T}}(b) = t_0$;
8. $\mathsf{tp}_{\mathcal{A}_{\mathsf{source}} \cup t_1(c),\mathcal{T}}(a) = \mathsf{tp}_{\mathcal{A},\mathcal{T}}(a)$ *for all* $a \in \mathsf{ind}(\mathcal{A}_{\mathsf{source}})$;
9. $\mathsf{tp}_{\mathcal{A}_{\mathsf{edge}} \cup t_1(b) \cup t_1(c),\mathcal{T}}(a) = \mathsf{tp}_{\mathcal{A},\mathcal{T}}(a)$ *for all* $a \in \mathsf{ind}(\mathcal{A}_{\mathsf{edge}})$;
10. $\mathsf{tp}_{\mathcal{A}_{\mathsf{target}} \cup t_1(b),\mathcal{T}}(a) = \mathsf{tp}_{\mathcal{A},\mathcal{T}}(a)$ *for all* $a \in \mathsf{ind}(\mathcal{A}_{\mathsf{target}})$.

With Lemma 16 at our disposal, we can now prove Lemma 15.

Proof of Lemma 15. For the more straightforward "$\Rightarrow$" direction, let t be reachable from s. Then there is a path $s = v_0, \ldots, v_n = t$ in G. Since $\mathcal{A}^s_{\mathsf{source}} \subseteq \mathcal{A}_G$, Point 1 of Lemma 16 and monotonicity[4] yield $t_1 \subseteq \mathsf{tp}_{\mathcal{A}_G,\mathcal{T}}(a_s)$. This provides the induction start for showing that $t_1 \subseteq \mathsf{tp}_{\mathcal{A}_G,\mathcal{T}}(a_{v_i})$ for $0 \leq i \leq n$, by induction on i. For the induction step, assume that $t_1 \subseteq \mathsf{tp}_{\mathcal{A}_G,\mathcal{T}}(a_{v_i})$ with $0 \leq i < n$. Since $\mathcal{A}^{v_i,v_{i+1}}_{\mathsf{edge}} \subseteq \mathcal{A}_G$, Point 2 of Lemma 16 and monotonicity yield $t_1 \subseteq \mathsf{tp}_{\mathcal{A}_G,\mathcal{T}}(a_{v_{i+1}})$. Finally, since $\mathcal{A}^t_{\mathsf{target}} \subseteq \mathcal{A}_G$, Point 3 of Lemma 16 and monotonicity yield $\mathcal{A}_G \models Q(\mathbf{a})$.

The "$\Leftarrow$" direction is more laborious. Assume that t is not reachable from s. We define an extension $\mathcal{A}'_G \supseteq \mathcal{A}_G$ and then show that $\mathcal{A}'_G \not\models Q(\mathbf{a})$, which implies $\mathcal{A}_G \not\models Q(\mathbf{a})$ as desired. The reason for working with $\mathcal{A}'_G$ in place of $\mathcal{A}_G$ is that the former lends itself better towards inductive proofs. Note that we could not have used $\mathcal{A}'_G$ to define the reduction since it uses symbols that are not in Σ and cannot be constructed by an FO-reduction since this requires deciding reachability in directed graphs.

For all $v \in V$, let $\mathsf{reach}(v) = 1$ if v is reachable from s in G and $\mathsf{reach}(v) = 0$ otherwise. Further let $(u_1, v_1), \dots, (u_m, v_m)$ be an enumeration of all edges in E. We inductively define a sequence of ABoxes $\mathcal{A}_0, \mathcal{A}_1, \dots, \mathcal{A}_m$ as follows:

$$\mathcal{A}_0 = \mathcal{A}^s_{\mathsf{source}} \cup t_1(a_s) \cup \mathcal{A}^t_{\mathsf{target}} \cup t_0(a_t)$$

and if $\mathcal{A}_{i-1}$ has already been defined, let

$$\mathcal{A}_i = \mathcal{A}_{i-1} \cup \mathcal{A}^{u_i,v_i}_{\mathsf{edge}} \cup t_{\mathsf{reach}(u_i)}(a_{u_i}) \cup t_{\mathsf{reach}(v_i)}(a_{v_i}).$$

Finally, we define $\mathcal{A}'_G = \mathcal{A}_m$. It is straightforward to verify that

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

Set $\mathsf{ind}_V = \{a_v \mid v \in V\}$. With the *components* of $\mathcal{A}'_G$, we mean the ABoxes $\mathcal{A}^+_{\mathsf{source}} := \mathcal{A}^s_{\mathsf{source}} \cup t_1(a_s)$, $\mathcal{A}^+_{\mathsf{target}} := \mathcal{A}^t_{\mathsf{target}} \cup t_0(a_t)$, and $\mathcal{A}^+_{u,v} := \mathcal{A}^{u,v}_{\mathsf{edge}} \cup t_{\mathsf{reach}(u)}(a_u) \cup t_{\mathsf{reach}(v)}(a_v)$ for all $(u, v) \in E$. By Points 4 to 7 of Lemma 16, the following types are realized in components:

- $\mathsf{tp}_{\mathcal{A}^+_{\mathsf{source}},\mathcal{T}}(a_s) = t_1 = t_{\mathsf{reach}(s)}$,
- $\mathsf{tp}_{\mathcal{A}^+_{\mathsf{target}},\mathcal{T}}(a_t) = t_0 = t_{\mathsf{reach}(t)}$,
- $\mathsf{tp}_{\mathcal{A}^+_{u,v},\mathcal{T}}(a_u) = t_{\mathsf{reach}(u)}$, and $\mathsf{tp}_{\mathcal{A}^+_{u,v},\mathcal{T}}(a_v) = t_{\mathsf{reach}(v)}$.

We next show the following.

Claim. Let $0 \leq i \leq m$. Then $\mathsf{tp}_{\mathcal{A}_i,\mathcal{T}}(a) = \mathsf{tp}_{\mathcal{C},\mathcal{T}}(a)$ for every component $\mathcal{C}$ of $\mathcal{A}'_G$ with $\mathcal{C} \subseteq \mathcal{A}_i$ and all $a \in \mathsf{ind}(\mathcal{C})$.

We prove the claim by induction on i. For $i = 0$, the only relevant components $\mathcal{C}$

[4]We mean that if $\mathcal{A} \subseteq \mathcal{A}'$ then $\mathcal{T}, \mathcal{A} \models q(\mathbf{a})$ implies $\mathcal{T}, \mathcal{A}' \models q(\mathbf{a})$ for all $\mathcal{T}$, q, and $\mathbf{a}$.

are $\mathcal{A}^+_{\mathsf{source}}$ and $\mathcal{A}^+_{\mathsf{target}}$. Note that they are disjoint since $s \neq t$, and thus the statement is clearly satisfied.

For the induction step, assume that the statement has already been proven for some $i \geq 0$. There is a component $\mathcal{C}_i = \mathcal{A}^+_{u,v}$ such that $\mathcal{A}_{i+1} = \mathcal{A}_i \cup \mathcal{C}_i$. Note that $\mathcal{A}_i$ and $\mathcal{C}_i$ might only share the individuals a_u and a_v. We have $\mathsf{tp}_{\mathcal{C}_i,\mathcal{T}}(a_u) = t_{\mathsf{reach}(u)}$ and $\mathsf{tp}_{\mathcal{C}_i,\mathcal{T}}(a_v) = t_{\mathsf{reach}(v)}$. By induction hypothesis, $\mathsf{tp}_{\mathcal{A}_i,\mathcal{T}}(a_u) = t_{\mathsf{reach}(u)}$ if a_u is shared and $\mathsf{tp}_{\mathcal{A}_i,\mathcal{T}}(a_v) = t_{\mathsf{reach}(v)}$ if a_v is shared. The statement thus follows from Lemma 7 for $\mathcal{C} = \mathcal{C}_i$, and from Lemma 7 and induction hypothesis for all $\mathcal{C} \neq \mathcal{C}_i$. This finishes the proof of the claim. Since $\mathcal{A}'_G = \mathcal{A}_m$, we may use the claim with $\mathcal{A}'_G$ in place of $\mathcal{A}_i$.

We now use the claim to show that $\mathcal{A}'_G, \mathcal{T} \not\models q(\mathbf{a})$. Assume to the contrary that $\mathcal{A}'_G, \mathcal{T} \models q(\mathbf{a})$, that is, there is a homomorphism h from $q(\mathbf{x})$ to $\mathcal{U}_{\mathcal{A}'_G,\mathcal{T}}$ such that $h(\mathbf{x}) = \mathbf{a}$. Note that any two distinct individuals from ind_V have distance at least $|q|$ in $\mathcal{A}'_G$ since the same is true for individuals b and c in $\mathcal{A}_{\mathsf{edge}}$. Since q is connected, there can thus be at most one such individual in the range of h.

If no individual from ind_V is in the range of h, then q being connected implies that there is a single component $\mathcal{C}$ of $\mathcal{A}'_G$ such that the range of h contains only individuals from $\mathcal{C}$ and anonymous elements below them.

First assume that $\mathcal{C}$ is $\mathcal{A}^+_{\mathsf{target}}$. We can construct a homomorphism h' from q to $\mathcal{U}_{\mathcal{A}^+_{\mathsf{target}},\mathcal{T}}$ by setting $h'(x) = h(x)$ if $h(x) \in \mathsf{ind}(\mathcal{A}'_G)$ and using the claim and Lemma 3 to define $h'(x)$ if $h(x)$ is an anonymous element. Thus $\mathcal{A}^+_{\mathsf{target}}, \mathcal{T} \models q(\mathbf{a})$, in contradiction to Condition (4) of Definition 11.

Now assume that $\mathcal{C}$ is $\mathcal{A}^+_{\mathsf{source}}$ or $\mathcal{A}^+_{u,v}$. We only treat the former as the latter is completely analogous. Since the constants from $\mathbf{a}$ are not in $\mathcal{A}^+_{\mathsf{source}}$, q is Boolean. We can construct a homomorphism h' from q to $\mathcal{U}_{\mathcal{A}^+_{\mathsf{source}},\mathcal{T}}$ as in the previous case. We can further construct a homomorphism h'' from $\mathcal{U}_{\mathcal{A}^+_{\mathsf{source}},\mathcal{T}}$ to $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ by starting with the obvious homomorphism from $\mathcal{A}^+_{\mathsf{source}}$ to $\mathcal{A}$ (map every $a \in \mathsf{ind}(\mathcal{A}^+_{\mathsf{source}})$ to the individual in $\mathcal{A}$ that a is a copy of) and using Points 8 to 10 of Lemma 16 and Lemma 3 to define $h''(d)$ for anonymous elements d. The composition g of h' with h'' is a homomorphism from q to $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ that is not core close since its range falls in the subtree of $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ rooted at c and the distance of c from the core is at least $|q|$. This contradicts Condition 5 of Definition 11.

Now assume that the range of h contains the individual a_v. Let $X_0 = \{x \in \mathsf{var}(q) \mid h(x) = a_v\}$ and let $X^{\downarrow}$ be the set of $x \in \mathsf{var}(q)$ such that $h(x)$ is some $a \in \mathsf{ind}(\mathcal{A}'_G)$ or in an anonymous tree below such an a such that there exists a path of length at least one and at most $|q|$ from a_v to a in $\mathcal{A}'_G$. Let $X^{\uparrow} = \mathsf{var}(q) \setminus (X_0 \cup X^{\downarrow})$. We distinguish three cases.

Case 1: $v = t$, the target node. Since we assume that t has outdegree 0 in G and $s \neq t$, $h(X^{\uparrow})$ contains only individuals from $\mathcal{A}^+_{\mathsf{target}}$ and anonymous elements below them while $h(X^{\downarrow})$ contains only individuals from (potentially multiple) components $\mathcal{A}^+_{u,v}$ and anonymous elements below them. There is then an obvious homomorphism from $\mathcal{A}'_G|_{h(\mathsf{var}(q))}$ to $\mathcal{A}$, and consequently we can construct a homomorphism g from q to $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ as above. Then b is in the range of g since a_v is a copy of b. Since the distance of b from the core of $\mathcal{A}$ is at least $|q|$, g is not core close, contradicting Condition 5 of Definition 11.

Case 2: $v = s$, the source node. Since we assume that s has indegree 0 in G and $s \neq t$, $h(X^{\downarrow})$ contains only individuals from $\mathcal{A}^+_{\mathsf{source}}$ or anonymous elements below them and $h(X^{\uparrow})$ contains only individuals from (potentially multiple) components $\mathcal{A}^+_{u,v}$ and anonymous elements below them. We can proceed as in the previous case to obtain a homomorphism from q to $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ whose range contains c, contradicting Condition 5 of Definition 11.

Case 3: $v \notin \{s,t\}$. Then the range of h contains only individuals from (potentially multiple) components $\mathcal{A}^+_{u,v}$ and anonymous elements below them. Note that unlike in the previous two case, there is no obvious homomorphism from $\mathcal{A}'_G|_{h(\mathsf{var}(q))}$ to $\mathcal{A}$, and thus we need a more refined argument. We can construct a homomorphism g from $q|_{X^{\uparrow}\cup X_0}$ to $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ as above that maps every $x \in \mathsf{var}(q)$ with $h(x) \in \mathsf{ind}(\mathcal{A}'_G)$ to the individual in $\mathcal{A}$ that $h(x)$ is a copy of. Then $g(x) = c$ for all $x \in X_0$. It remains to extend g to all of q. Let q' be obtained from q by identifying $x_1, x_2 \in \mathsf{var}(q)$ whenever $r(x_1,y), r(x_2,y) \in q$ and $x_1, x_2 \in X_0 \cup X^{\downarrow}$. By construction, the restriction of $\mathcal{A}'_G$ to the individuals between a_v and $\{h(x) \mid x \in X^{\downarrow}\}$ is a directed tree. Consequently and since the anonymous elements in the universal model also form directed trees, h is a homomorphism from q' to $\mathcal{U}_{\mathcal{A}'_G,\mathcal{T}}$. Moreover, $q' \setminus q|_{X^{\uparrow}\cup X_0}$ is the union of tree-shaped CQs $q_1,\dots,q_n$ that all share the same root x_0 and are otherwise variable disjoint. Each q_i can be viewed as an $\mathcal{EL}$-concept $\exists r.C$. By the extension of $\mathcal{T}$ carried out initially, $\exists r.C \sqsubseteq A_{\exists r.C} \in \mathcal{T}$. Consequently $A_{\exists r.C} \in \mathsf{tp}_{\mathcal{A}'_G,\mathcal{T}}(a_v)$ which by the claim is identical to $\mathsf{tp}_{\mathcal{C},\mathcal{T}}(a_v)$ in some component $\mathcal{C} = \mathcal{A}^+_{u,v}$, and thus $A_{\exists r.C} \in t_1$ since $\mathsf{tp}_{\mathcal{C},\mathcal{T}}(a_v) \subseteq t_1$. By Condition 2 of Definition 11, $A_{\exists r.C} \in \mathsf{tp}_{\mathcal{A},\mathcal{T}}(c)$. We thus find a homomorphism from q_i to $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ that maps x_0 to c, for $1 \leq i \leq n$. Combining all these homomorphisms allows us to extend g to q', thus to q. Again, g is not core close and thus we obtain a contradiction to Condition 5 of Lemma 11. □

Together, Propositions 13 and 14 show if an OMQ $Q \in (\mathcal{EL}, \text{conCQ})$ has unbounded depth, then $\textsc{eval}(Q)$ is NL-hard. This finishes the proof of Theorem 9.

4. NL versus PTime for Connected CQs

We prove a dichotomy between NL and PTime for $(\mathcal{EL}, \text{conCQ})$ and show that for OMQs from this language, evaluation in NL coincides with rewritability into linear Datalog. We also show that the latter two properties coincide with the OMQ having bounded pathwidth, as defined below. We generalize our results to potentially disconnected CQs in Section 5.

Let $Q = (\mathcal{T}, \Sigma, q) \in (\mathcal{EL}, \text{CQ})$. We say that Q *has pathwidth at most k* if for every Σ-ABox $\mathcal{A}$ and tuple $\mathbf{a}$ with $\mathcal{A} \models Q(\mathbf{a})$, there is a Σ-ABox $\mathcal{A}'$ of pathwidth at most k such that $\mathcal{A}' \models Q(\mathbf{a})$ and a homomorphism from $\mathcal{A}'$ to $\mathcal{A}$ that is the identity on $\mathbf{a}$. Now Q has *bounded pathwidth* if it has pathwidth at most k for some k. If this is the case, we use $\mathsf{pw}(Q)$ to denote the smallest k such that Q has pathwidth at most k. The OMQ Q_2 from Example 1 has bounded pathwidth since every ABox from $\mathcal{M}_{Q_2}$ takes the form of a path and every path has pathwidth 1. On the other hand, Q_1 has unbounded pathwidth. To see this, one may consider as witnesses the ABoxes that take the form of full binary trees of increasing depth where the left successor is connected via role name $\mathsf{hasMother}$, the right successor via role name $\mathsf{hasFather}$, and for all leafs a there is a concept assertion $\mathsf{Gene1Carrier}(a)$. It is well known that the pathwidth of the full binary tree of depth k is $\lceil \frac{k}{2} \rceil$, so the pathwidth of these ABoxes is not bounded [49]. The following is the main result of this section.

Theorem 17 (NL/PTIME dichotomy)**.** *Let $Q \in (\mathcal{EL}, \text{conCQ})$. The following are equivalent (assuming $\text{NL} \neq \text{PTIME}$):*

(i) Q has bounded pathwidth.

(ii) Q is rewritable into linear Datalog.

(iii) $\text{EVAL}(Q)$ is in NL.

If these conditions do not hold, then $\text{EVAL}(Q)$ is PTIME-hard under FO reductions.

Remark. Without the assumption $\text{NL} \neq \text{PTIME}$, Conditions (i) and (ii) are still equivalent to each other and they still imply (iii).

The equivalence (i) $\Leftrightarrow$ (ii) is closely related to a result in CSP. In fact, it is proved in [21] that a CSP has an obstruction set of bounded pathwidth if and only if its complement is expressible in linear Datalog. From the viewpoint of the connection between OMQs and CSPs [3], obstructions correspond to homomorphic preimages of ABoxes and thus the result in [21] implies (i) $\Leftrightarrow$ (ii) for OMQs of the form $(\mathcal{T}, \Sigma, \exists x A(x))$, $\mathcal{T}$ formulated in $\mathcal{ELI}$. We give a direct proof of (i) $\Leftrightarrow$ (ii) in Section 4.2 to capture also CQs of more complex shape.

The implication (ii) $\Rightarrow$ (iii) is clear since every linear Datalog program can be evaluated in NL. It thus remains to prove the converse and the last sentence of the theorem. Since we assume $\text{NL} \neq \text{PTIME}$, we may achieve both by showing that unbounded pathwidth implies PTIME-hardness. This is achieved in the subsequent section.

4.1. Unbounded Pathwidth Implies PTIME*-hardness*

We reduce from the well-known PTIME-complete problem *path systems accessibility* (PSA) [48], closely related to alternating reachability on directed graphs and to the evaluation of Boolean circuits. The structure of the proof is similar to the one for the dichotomy between AC^0 and NL in Section 3, but more sophisticated. 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*. A node $v \in V$ is *accessible* if $v \in S$ or there are accessible nodes u, w with $(u, w, v) \in E$. G is a yes-instance if the target node t is accessible. We assume w.l.o.g. that t does not appear in the first or second component of a triple in E, that no $s \in S$ appears in the third component of a triple in E, and that $t \notin S$.

The main difference to the NL-hardness proof in Section 3 is that instead of a gadget $\mathcal{A}_{\mathsf{edge}}$ that transports a selected type t_1 from its input individual to its output individual, we now need a gadget $\mathcal{A}_\wedge$ with two input individuals and one output individual that behaves like a logical AND-gate when type t_1 is interpreted as 'true' and type t_0 as 'false'. Similar to the NL-hardness proof, the reduction translates a tuple $G = (V, E, S, t)$ into a Σ-ABox $\mathcal{A}_G$ and a tuple $\mathbf{a}$ such that $\mathcal{A}_G \models Q(\mathbf{a})$ if and only if t is accessible in G. But instead of replacing binary edges by a copy of $\mathcal{A}_{\mathsf{edge}}$, we replace the ternary tuples from E by a copy of $\mathcal{A}_\wedge$. Below, we formalize the existence of suitable gadgets as the ability to simulate PSA.

Instead of proving directly that unbounded pathwidth of an OMQ Q implies that Q has the ability to simulate PSA, we consider it more convenient to use an intermediate condition, namely that Q has unbounded branching. This means that for any depth bound k, there is a pseudo tree-shaped ABox in $\mathcal{M}_Q$ that contains the full binary tree of depth k as a minor. We show that for any OMQ Q from $(\mathcal{EL}, \text{CQ})$,

1. Q has unbounded pathwidth if and only if Q has unbounded branching;
2. Q has unbounded branching if and only if Q has the ability to simulate PSA;
3. if Q has the ability to simular PSA, then $\mathsf{eval}(Q)$ is PTIME-hard.

While the 'only if' directions of Points 1 and 2 are not required to establish Theorem 17, they are useful for the complexity analysis of the meta problems carried out in Section 7.

We define unbounded branching more formally. Let $\mathcal{A}$ be a tree-shaped ABox. The *full binary tree of depth k* is the directed graph $G = (V, E)$ with $V = \{w \in \{1, 2\}^* \mid 0 \leq |w| \leq k\}$ and $(v, w) \in E$ if $w = v1$ or $w = v2$. *$\mathcal{A}$ has the full binary tree of depth k as a minor* if there is a mapping f from the nodes of the full binary tree of depth k to $\mathsf{ind}(\mathcal{A})$ such that if $(v, w) \in E$, then $f(w)$ is a descendant of $f(v)$. We usually do not make the mapping f explicit but only say which individuals lie in the range of f.[5]

We are mostly interested in the largest k such that $\mathcal{A}$ has the full binary tree of depth k as a minor. This number, which we call the *branching number* of $\mathcal{A}$, denoted by $\mathsf{br}(\mathcal{A})$, can be easily computed by the following algorithm. Label every leaf of $\mathcal{A}$ with 0 and then inductively label the inner nodes as follows: If a is an inner node whose children have already been labeled and m is the maximum label of its children, label a with m if at most one child of a is labeled with m and label a with $m + 1$ if at least two children of a are labeled with m. It can be easily proved by induction on the co-depth of individuals a that the label of a is equal to $\mathsf{br}(\mathcal{A}^a)$. In particular, $\mathsf{br}(\mathcal{A})$ is the label of the root of $\mathcal{A}$.

[5]The standard definition of a minor says that graph H is a minor of graph G if H can be obtained from a subgraph of G by edge contraction. This is equivalent to our definition.

We say that $Q \in (\mathcal{EL}, \mathrm{CQ})$ is *boundedly branching* if there exists a k such that for every pseudo tree-shaped ABox $\mathcal{A} \in \mathcal{M}_Q$ and every tree $\mathcal{A}_i$ in $\mathcal{A}$, we have $\mathsf{br}(\mathcal{A}_i) \leq k$. In that case, we define $\mathsf{br}(Q)$ to be the smallest such k. Otherwise, we call Q *unboundedly branching.*

Proposition 18. *Let $Q \in (\mathcal{EL}, \mathrm{CQ})$. Then Q has unbounded pathwidth iff Q is unboundedly branching.*

The "$\Leftarrow$" direction follows from the facts that the full binary tree of depth k has pathwidth $\lceil \frac{k}{2} \rceil$ and that the pathwidth of a graph cannot be smaller than the pathwidth of its minors. For the "$\Rightarrow$" direction, we show that the branching number of an OMQ gives rise to an upper bound on its pathwidth. This is done by induction on the depth of tree-shaped ABoxes and then lifted to pseudo tree-shaped ABoxes.

Our next goal is to identify suitable gadgets for the reduction from PSA. To achieve this, it is convenient to extend the TBox of the OMQ $Q = (\mathcal{T}, \Sigma, q)$ involved in the reduction. Recall that we have also used such an extension in the NL-hardness proof in Section 3 and that it has helped us to avoid unintended homomorphisms from the CQ to the (universal model of the) reduction ABox $\mathcal{A}_G$, in case that the CQ is Boolean. Avoiding such homomorphisms is more complicated in the reduction of PSA which leads us to a different TBox extension that introduces inverse roles and thus results in an $\mathcal{ELI}$-TBox. This is unproblematic since, as in Section 3, the OMQ based on the extended TBox is equivalent to the original OMQ.

First assume that q is Boolean and treeifiable. A *role path between variables x and y in q* is a sequence of role names $r_1 \cdots r_n$ such that for distinct variables $x = x_0, \ldots, x_n = y$, q contains the atoms $r_1(x_1, x_2), \ldots, r_n(x_{n-1}, x_n)$. If q is treeifiable, then there are only polynomially many role paths in q^{tree}: the paths that occur in q^{tree}, the least constrained treeification of q defined in Section 2. Let $\mathcal{C}_q$ denote the set of $\mathcal{ELI}$-concepts of the form $\exists r_n^-. \cdots .\exists r_1^-.C$ where $r_1 \cdots r_n$ is a (potentially empty) role path in q^{tree} and C is $\top$ or a concept name from q or a CQ from $\mathsf{trees}(q)$ viewed as an $\mathcal{EL}$-concept. Extend $\mathcal{T}$ with $C \sqsubseteq A_C$, A_C a fresh concept name, for all $C \in \mathcal{C}_q$. Finally, normalize again. Clearly, the number of concept inclusions added to $\mathcal{T}$ is polynomial in $|q|$ and the resulting OMQ is equivalent to the original one.

Now assume that q is not Boolean and treeifiable. Then unintended homomorphisms are ruled out automatically. To prepare for the complexity analysis of the meta problems carried out in Section 7, however, we still carry out the same modification that we have also used in Section 3: For every $p \in \mathsf{trees}(q)$, view p as an $\mathcal{EL}$-concept C and extend $\mathcal{T}$ with $C \sqsubseteq A_C$, A_C a fresh concept name.

The following lemma prepares for the use of the concepts $\mathcal{C}_q$ later on and gives an idea of why we use this particular set of concepts.

Lemma 19. *Let $q \in conCQ$ be Boolean and treeifiable, $\mathcal{I}_1, \mathcal{I}_2$ tree-shaped interpretations, and $d_i \in \Delta^{\mathcal{I}_i}$ for $i \in \{1, 2\}$ such that $d_1 \in C^{\mathcal{I}_1}$ implies $d_2 \in C^{\mathcal{I}_2}$ for* *all $C \in \mathcal{C}_q$. If there is a homomorphism from q to $\mathcal{I}_1$ with d_1 in its range, then there is a homomorphism from q to $\mathcal{I}_2$ with d_2 in its range.*

If $\mathcal{A}$ is a pseudo tree-shaped ABox and $b \in \mathsf{ind}(\mathcal{A})$ has distance at least n from the core, we define the *ancestor path of b up to length n* to be the unique sequence $r_1 r_2 \dots r_n$ of role names such that $r_1(b_1, b_2), r_2(b_2, b_3), \dots r_n(b_n, b) \in \mathcal{A}$.

Definition 20. Let $Q = (\mathcal{T}, \Sigma, q) \in (\mathcal{EL}, \text{conCQ})$. We say that Q has *the ability to simulate* PSA if there exist

- a pseudo tree-shaped Σ-ABox $\mathcal{A}$ of core size $|q|$,
- a tuple $\mathbf{a}$ from the core of $\mathcal{A}$ of length $\mathsf{ar}(q)$,
- a tree $\mathcal{A}_i$ in $\mathcal{A}$ with three distinguished non-core individuals b, c and d from $\mathsf{ind}(\mathcal{A}_i)$ where c and d are incomparable descendants of b and such that b has distance at least $|q|$ from the core and the individuals b, c and d have pairwise distance at least $|q|$ from each other and
- $\mathcal{T}$-types $t_0 \subsetneq t_1$,

such that

1. $\mathcal{A}, \mathcal{T} \models q(\mathbf{a})$;
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. $\mathcal{A}_b \cup t_0(b), \mathcal{T} \not\models q(\mathbf{a})$;
4. $\mathsf{tp}_{\mathcal{A}_c \cup t_0(c),\mathcal{T}}(b) = \mathsf{tp}_{\mathcal{A}_d \cup t_0(d),\mathcal{T}}(b) = t_0$,
5. if q is Boolean, then every homomorphism from q to $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ is core close and
6. if q is Boolean, then b, c and d have the same ancestor path up to length $|q|$.

We define $\mathcal{A}_{\mathsf{target}} := \mathcal{A}_b$, $\mathcal{A}_\wedge := \mathcal{A}^b_{cd}$ and $\mathcal{A}_{\mathsf{source}} := \mathcal{A}^c$.

With c and d being 'incomparable' descendants of b, we mean that neither d is a descendant of c nor vice versa. To understand the essence of Definition 20, it is worthwhile to consider the special case where q is an AQ $A(x)$. In this case, Q has the ability to simulate PSA if there is a tree-shaped Σ-ABox $\mathcal{A}$ with root $a = \mathbf{a}$, three distinguished non-root individuals $b, c, d \in \mathsf{ind}(\mathcal{A})$, c and d incomparable descendants of b, and $\mathcal{T}$-types $t_0 \subsetneq t_1$ such that Conditions (1)-(4) of Definition 20 are satisfied. Condition (2) expresses that $\mathcal{A}_{\mathsf{source}}$ produces the type t_1 and also that $\mathcal{A}_\wedge$ propagates this type further to its 'output' (b) if both of its 'inputs' (c and d) receive the type t_1. Condition (4) says that replacing one of the inputs by t_0 will change the output to t_0, just as expected in a logical AND-gate. Condition (3) says that the type t_0 as input for $\mathcal{T}_{\mathsf{target}}$ is not sufficient to make the query true. All remaining parts of Definition 20 should be thought of as technical complications induced by replacing AQs with CQs.

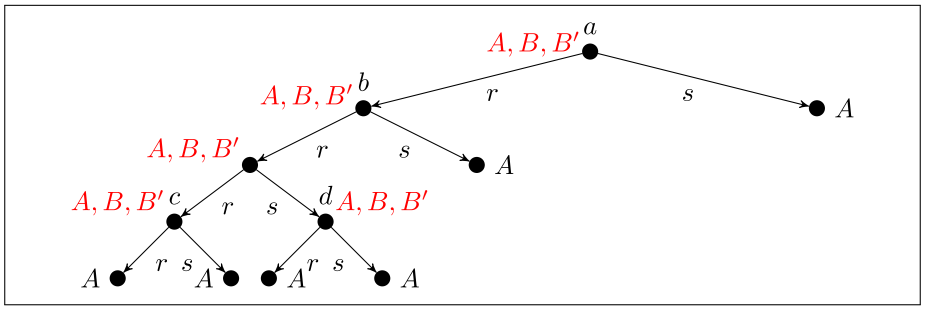

Figure 6: The ABox $\mathcal{A}$ displayed in black witnesses that the OMQ from Example 21 has the ability to simulate PSA. The assertions in red are derived from $\mathcal{A}$ using $\mathcal{T}$, showing that $t_1 = \mathsf{tp}_{\mathcal{A},\mathcal{T}}(b) = \mathsf{tp}_{\mathcal{A},\mathcal{T}}(c) = \mathsf{tp}_{\mathcal{A},\mathcal{T}}(d)$.

Example 21. Consider the OMQ $Q = (\mathcal{T}, \Sigma, A(x))$ where

$$\mathcal{T} = \{\exists r.A \sqsubseteq B, \exists s.A \sqsubseteq B', B \sqcap B' \sqsubseteq A\} \quad \text{and} \quad \Sigma = \{r, s, A\}.$$

Figure 6 shows an ABox $\mathcal{A}$ that witnesses that Q has the ability to simulate PSA, via the types $t_1 = \{A, B, B'\}$ and $t_0 = \{B'\}$, and with $\mathbf{a} = a$.

We next show that the ability to simulate PSA is equivalent to unbounded branching. As a preliminary, we give the following combinatorial lemma.

Lemma 22. *Let T be a full binary tree of depth $n \cdot k \cdot d$ whose nodes are colored with n colors, $k \geq 0$ and $n, d \geq 1$. Then T has as a minor a monochromatic full binary tree of depth k such that any two distinct nodes of the minor have distance at least d from each other in T.*

Now for the announced equivalence.

Proposition 23. *Let $Q \in (\mathcal{EL}, \text{conCQ})$. Then Q has the ability to simulate* PSA *iff Q is unboundedly branching.*

Proof. We present the proof of the "$\Leftarrow$" direction here as this is the important direction for showing that unbounded pathwidth implies PTIME-hardness, the main aim of this section. The proof of the"$\Rightarrow$" direction is in the appendix.

Assume that $Q = (\mathcal{T}, \Sigma, q) \in (\mathcal{EL}, \text{conCQ})$ is not boundedly branching. Let TP denote the set of all $\mathcal{T}$-types and set $m = 2^{|\mathcal{T}|}$. Clearly, $|\mathsf{TP}| \leq m$. Set $k = m \cdot 2^m \cdot |\mathcal{T}|^{|q|} \cdot (m^m + 1) \cdot |q|$. Since Q is not boundedly branching, we find a Σ-ABox $\mathcal{A} \in \mathcal{M}_Q$ and a tuple $\mathbf{a}$ from its core such that $\mathcal{A}, \mathcal{T} \models q(\mathbf{a})$ and one the trees $\mathcal{A}_i$ of $\mathcal{A}$ has the full binary tree of depth k as a minor. We show that $\mathcal{A}$ and $\mathbf{a}$ can serve as the ABox and tuple in Definition 20, that is, as a witness for Q having the ability to simulate PSA.

To identify the distinguished individuals b, c, d, we use a suitable coloring of the individuals of $\mathcal{A}_i$ and Lemma 22. In fact, we color every $a \in \mathsf{ind}(\mathcal{A}_i)$ with the color $(\mathsf{tp}_{\mathcal{A},\mathcal{T}}(a), S_a, r_1^a r_2^a \ldots r_{|q|}^a)$ where $S_a = \{t \in \mathsf{TP} \mid \mathcal{A}_a \cup t(a), \mathcal{T} \models q(\mathbf{a})\}$ and where $r_1^a r_2^a \ldots r_{|q|}^a$ is the ancestor path of a up to length $|q|$. There are no more than $m \cdot 2^m \cdot |\mathcal{T}|^{|q|}$ colors, so from Lemma 22 we know that $\mathcal{A}$ has as a minor a monochromatic full binary tree T of depth $m^m + 1$ whose nodes have distance at least $|q|$ from each other. Let b be a child of the root of T (to make sure that b has depth at least $|q|$ from the core) and $T' \subseteq T$ the subtree of T rooted at b, so T' is a full binary tree of depth m^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 22, there will be the monochromatic binary tree of depth 1 as a minor. In particular, we find two incomparable individuals c and d in T' that are colored with the same function. We show that we can find types t_1 and t_0 such that with the distinguished nodes b, c, d, $\mathcal{A}$ and $\mathbf{a}$ satisfy Conditions 1-6 from Definition 20.

Condition 1 is true by choice of $\mathcal{A}$. Set $t_1 := \mathsf{tp}_{\mathcal{A},\mathcal{T}}(b)$. Then Condition 2 is satisfied because b, c and d were colored with the same color by the first coloring. For the same reason, Conditon 6 is fulfilled. Condition 5 follows from Lemma 10.

To find t_0, we define a sequence $t'_0, t'_1, \ldots$ 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 is clear that $t'_i \subseteq t'_{i+1}$ for all i. Let t_0 be the limit of the sequence. 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 $\mathcal{A}_c \cup t'_i(c), \mathcal{T} \not\models q(\mathbf{a})$ for all $i \geq 0$. It is clear that $\mathcal{A}_c \cup t'_0(c), \mathcal{T} \not\models q(\mathbf{a})$ since $\mathcal{A}$ is minimal with $\mathcal{A}, \mathcal{T} \models q(\mathbf{a})$. Now assume that $\mathcal{A}_c \cup t'_i(c), \mathcal{T} \not\models q(\mathbf{a})$ for some i. Then $\mathcal{A}_c \cup t'_{i+1}(b), \mathcal{T} \not\models q(\mathbf{a})$. Since $S_b = S_c$ has been assured by the first coloring, we obtain $\mathcal{A}_c \cup t'_{i+1}(c), \mathcal{T} \not\models q(\mathbf{a})$ which completes the induction.

Thus $\mathcal{A}_c \cup t_0(c), \mathcal{T} \not\models q(\mathbf{a})$ and using again that $S_b = S_c$, we obtain Condition 3. □

To establish that unbounded pathwidth implies PTime-hardness, it remains to show that the ability to simulate PSA implies PTime-hardness.

Proposition 24. *If $Q \in (\mathcal{EL}, \text{conCQ})$ has the ability to simulate* PSA, *then* EVAL(Q) *is* PTime*-hard under FO reductions.*

To prove Proposition 24, let $Q = (\mathcal{T}, \Sigma, q) \in (\mathcal{EL}, \text{conCQ})$ have the ability to simulate PSA. Then there is a pseudo tree-shaped Σ-ABox $\mathcal{A}$, a tuple $\mathbf{a}$ in its core, distinguished individuals b, c and d, and types $t_0 \subsetneq t_1$ as in Definition 20. Fix such $\mathcal{A}, \mathbf{a}, b, c, d, t_0$ and t_1 for what follows. This gives rise to ABoxes $\mathcal{A}_{\mathsf{source}}$, $\mathcal{A}_\wedge$ and $\mathcal{A}_{\mathsf{target}}$ as in Definition 20. We reduce PSA to EVAL(Q).

Let $G = (V, E, S, t)$ be an input for PSA. We use (copies of) $\mathcal{A}_{\mathsf{source}}$, $\mathcal{A}_\wedge$ and $\mathcal{A}_{\mathsf{target}}$ as building blocks to construct a Σ-ABox $\mathcal{A}_G$ with a designated tuple of individual names $\mathbf{a}$ such that $\mathcal{A}_G \models Q(\mathbf{a})$ if and only if t is accessible in G. We reserve an individual a_v for every node $v \in V$, to be used in $\mathcal{A}_G$ along with additional individuals. Define

$$\mathcal{A}_G \;:=\; \bigcup_{s \in S} \mathcal{A}^s_{\mathsf{source}} \;\cup\; \bigcup_{(u,v,w) \in E} \mathcal{A}^{u,v,w}_\wedge \;\cup\; \mathcal{A}^t_{\mathsf{target}}$$

where

- for every $s \in S$, $\mathcal{A}^s_{\mathsf{source}}$ is a copy of $\mathcal{A}_{\mathsf{source}}$ where c is renamed to a_s and all other individuals are fresh;
- for every $(u, v, w) \in E$, $\mathcal{A}^{u,v,w}_{\wedge}$ is a copy of $\mathcal{A}_{\wedge}$ where c is renamed to a_u, d is renamed to a_v, b is renamed to a_w and all other individuals are fresh;
- $\mathcal{A}^t_{\mathsf{target}}$ is a copy of $\mathcal{A}_{\mathsf{target}}$ where b is renamed to a_t, all individuals from $\mathbf{a}$ retain their original name, and all other individuals are fresh.[6]

It can be verified that $\mathcal{A}_G$ can be constructed from G using an FO-query, see the comments given in Section 3. It thus remains to show the correctness of the reduction.

Lemma 25. *t is accessible in G iff $\mathcal{A}_G \models Q(\mathbf{a})$.*

The proof of Lemma 25 is similar to that of Lemma 15. We only give an overview here and present details in the appendix. As for Lemma 15, the "$\Rightarrow$" direction is relatively simply, relying only on monotonicity and three technical properties of the component ABoxes $\mathcal{A}_{\mathsf{source}}$, $\mathcal{A}_{\wedge}$ and $\mathcal{A}_{\mathsf{target}}$, made precise in Lemma 46 in the appendix.

The heart of the proof is the "$\Leftarrow$" direction. As in the proof of Lemma 15, we assume that t is not accessible in G, consider an ABox $\mathcal{A}'_G \supseteq \mathcal{A}_G$, and prove that $\mathcal{A}'_G \not\models Q(\mathbf{a})$. More precisely, $\mathcal{A}'_G$ is assembled in a step-by-step way by taking isomorphic copies of the ABoxes $\mathcal{A}^+_s := \mathcal{A}^s_{\mathsf{source}} \cup t_1(a_s)$ for all $s \in S$, $\mathcal{A}^+_{\mathsf{target}} := \mathcal{A}^t_{\mathsf{target}} \cup t_0(a_t)$, and $\mathcal{A}^+_{u,v,w} := \mathcal{A}^{u,v,w}_{\wedge} \cup t_{\mathsf{acc}(u)}(a_u) \cup t_{\mathsf{acc}(v)}(a_v) \cup t_{\mathsf{acc}(w)}(a_w)$ for all $(u, v, w) \in E$. We call these the *components* of $\mathcal{A}'_G$. We then prove that each individual a in $\mathcal{A}'_G$ realizes the same type as the individual in the original component that it is a copy of, which essentially enables us to apply the conditions from the ability to simulate PSA to the ABox $\mathcal{A}'_G$. We then assume to the contrary of what is to be shown that $\mathcal{A}'_G \models Q(\bar{a})$ and take a witnessing homomorphism h. Set $\mathsf{ind}_V = \{a_v \mid v \in V\}$. Any two distinct individuals from ind_V have distance at least $|q|$ in $\mathcal{A}'_G$ since the same is true for the individuals b, c, d in $\mathcal{A}_{\wedge}$. Since q is connected, there can thus be at most one such individual in the range of h. If there is no individual in the range of h, than the range of h must fall within a single copy of a component. This can be used to derive a contradiction against Condition 3 and 5 of the ability to simulate PSA. It remains to deal with the case that the range of h contains some $a_v \in \mathsf{ind}_V$. We first show that, then, q is Boolean an treeifiable. We then use h to construct a homomorphism h' from q to $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ with $h(\bar{x}) = \bar{a}$, again contradicting Condition 5 of the ability to simulate PSA. The construction of h' is subtle and relies on the extension of $\mathcal{T}$ with inverse roles and on Condition 6 of the ability to simulate PSA.

4.2. Bounded Pathwidth Implies Linear Datalog Rewritability

We prove the equivalence (i) $\Leftrightarrow$ (ii) from Theorem 17. Our proof works even for OMQs from $(\mathcal{ELI}, \text{CQ})$, that is, when inverse role are admitted in the

[6]As a consequence, all individuals from $\mathbf{a}$ are part of $\mathcal{A}_G$.

TBox and when the conjunctive queries are not necessarily connected. We thus establish our result for this more general class of OMQs right away.

Proposition 26. *Let $Q = (\mathcal{T}, \Sigma, q) \in (\mathcal{ELI}, \mathrm{CQ})$. Then Q has bounded pathwidth if and only if Q is rewritable into linear Datalog. In the positive case, there exists a linear Datalog program of width $\mathsf{pw}(Q) + \mathsf{ar}(q)$.*

Before proving Proposition 26, we give some additional preliminaries about Datalog. Let Π be a Datalog program, $\mathcal{A}$ an ABox and $\mathbf{a}$ a tuple from $\mathsf{ind}(\mathcal{A})$. A *derivation of* $\Pi(\mathbf{a})$ *in* $\mathcal{A}$ is a labelled directed tree (V, E, ℓ) where

1. $\ell(x_0) = \mathsf{goal}(\mathbf{a})$ 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 $S(\mathbf{y}) \leftarrow p(\mathbf{x})$ 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 $p(\sigma\mathbf{x})$;
3. if x is a leaf, then $\ell(x) \in \mathcal{A}$.

It is well known that $\mathcal{A} \models \Pi(\mathbf{a})$ iff there is a derivation of $\Pi(\mathbf{a})$ in $\mathcal{A}$.

We associate with each derivation $D = (V, E, \ell)$ of $\Pi(\mathbf{a})$ in $\mathcal{A}$ an ABox $\mathcal{A}_D$. 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 non-leafs and $y_{\ell+1}, \dots, y_k$ are leafs, then $\mathcal{A}_x$ is obtained by starting with the assertions 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 is well known [40] and easy to verify.

Lemma 27. *Let Π be a linear Datalog program and let D be a derivation of $\Pi(\mathbf{a})$ in $\mathcal{A}$ and Π of diameter d. Then*

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

With this lemma, the "$\Leftarrow$" direction of Proposition 26 is easy to prove. Assume that $Q \in (\mathcal{ELI}, \mathrm{CQ})$ is rewritable into a linear Datalog program Π. Then $\mathsf{pw}(Q)$ is bounded from above by the diameter d of Π. In fact, take any pair $(\mathcal{A}, \mathbf{a})$ such that $\mathcal{A} \models Q(\mathbf{a})$. Since Π is a linear Datalog rewriting of Q, there exists a derivation of $\Pi(\mathbf{a})$ in $\mathcal{A}$. By Lemma 27, there exists an ABox $\mathcal{A}_D$ of pathwidth at most d such that $\mathcal{A}_D \models Q(\mathbf{a})$ and a homomorphism from $\mathcal{A}_D$ to $\mathcal{A}$ that is the identity on $\mathbf{a}$. Hence, Q has pathwidth at most d.

The rest of this section is concerned with proving the "$\Rightarrow$" direction of Proposition 26. Assume that Q has bounded pathwidth, say $\mathsf{pw}(Q) = k$. We obtain a linear Datalog program in the following way. We encode pairs $(\mathcal{A}, \mathbf{a})$ of an ABox $\mathcal{A}$ of pathwidth at most k and a tuple $\mathbf{a}$ from $\mathcal{A}$ as words over a finite alphabet, where one symbol of the word encodes one bag of the path decomposition of $\mathcal{A}$. We then construct an alternating two-way automaton on finite words that accepts precisely those words that encode a pair $(\mathcal{A}, \mathbf{a})$ such that $\mathcal{A} \models Q(\mathbf{a})$. Such an automaton can be transformed into a deterministic one-way automaton that accepts the same language [50]. From the latter automaton, we then construct the linear Datalog program that is equivalent to Q.

Derivation trees for AQs. It is well known that in $\mathcal{ELI}$, entailment of AQs can be characterized in terms of derivation trees [11]. Let $\mathcal{T}$ be an $\mathcal{ELI}$-TBox in normal form, $\mathcal{A}$ an ABox, $a_0 \in \mathsf{ind}(\mathcal{A})$ and $A_0 \in \mathsf{N_C}$. A *derivation tree* for $A_0(a_0)$ is a finite $\mathsf{ind}(\mathcal{A}) \times \mathsf{N_C}$-labeled tree (T, ℓ) such that

- $\ell(\varepsilon) = (a_0, A_0)$;
- if $\ell(x) = (a, A)$ and either $A(a) \in \mathcal{A}$ or $\top \sqsubseteq A \in \mathcal{T}$, then x is a leaf;
- if $\ell(x) = (a, A)$ and neither $A(a) \in \mathcal{A}$ or $\top \sqsubseteq A \in \mathcal{T}$, then one of the following holds:
  - x has successors $y_1, \ldots, y_n$ with $n \geq 1$ and $\ell(y_i) = (a, A_i)$ such that $\mathcal{T} \models A_1 \sqcap \ldots \sqcap A_n \sqsubseteq A$;
  - x has a single successor y with $\ell(y) = (b, B)$ such that $\mathcal{T} \models \exists r.B \sqsubseteq A$ and $r(a, b) \in \mathcal{A}$, where r is a (possibly inverse) role.

For proving the correctness of the constructed automaton later on, we need the following lemma, which is a special case of Lemma 29 in [11].

Lemma 28. *Let $\mathcal{T}$ be an $\mathcal{ELI}$-TBox in normal form, $\mathcal{A}$ an ABox, $a \in \mathsf{ind}(\mathcal{A})$ and $A \in \mathsf{N_C}$. Then $\mathcal{A}, \mathcal{T} \models A(a)$ if and only if there exists a derivation tree for $A(a)$.*

Two-way alternating finite state automata. We introduce *two-way alternating finite state automata (2AFAs).* 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 admitted as well. A 2AFA is a tuple $\mathfrak{A} = (S, \Gamma, \delta, s_0)$, where S is a finite set of *states*, Γ a finite alphabet, $\delta : S \times (\Gamma \cup \{\vdash, \dashv\}) \rightarrow \mathcal{B}^+(\{\mathsf{left}, \mathsf{right}, \mathsf{stay}\} \times S)$ the *transition function* and $s_0 \in S$ the *initial state.* The two symbols $\vdash$ and $\dashv$ are used as the left end marker and right end marker, respectively, and it is required that $\delta(s, \vdash) \in \mathcal{B}^+(\{\mathsf{right}\} \times S)$ and $\delta(s, \dashv) \in \mathcal{B}^+(\{\mathsf{left}\} \times S)$ for all $s \in S$ so that the 2AFA can never leave the space of the input word.

For an input word $w = w_1 \ldots w_n \in \Gamma^n$, define $w_0 = \vdash$ and $w_{n+1} = \dashv$. A *configuration* is a pair $(i, s) \in \{0, \ldots, n+1\} \times S$. An *accepting run* of a 2AFA $\mathfrak{A} = (S, \Gamma, \delta, s_0)$ on w is a pair (T, r) that consists of a finite tree T and a labeling r that assigns a configuration to every node in T such that

1. $r(\varepsilon) = (1, s_0)$, where ε is the root of T and

2. if $m \in T$, $r(m) = (i, s)$, and $\delta(s, w_i) = \varphi$, then there is a (possibly empty) set $V \subseteq \{\mathsf{left}, \mathsf{right}, \mathsf{stay}\} \times S$ such that V (viewed as a propositional valuation) satisfies φ and for every $(\mathsf{left}, s') \in V$ there is a successor of m in T labeled with $(i-1, s')$, for every $(\mathsf{right}, s') \in V$ there is a successor of m in T labeled with $(i+1, s')$ and for every $(\mathsf{stay}, s') \in V$ there is a successor of m in T labeled with (i, s').

The *language accepted by a 2AFA* $\mathfrak{A}$, denoted by $L(\mathfrak{A})$, is the set of all words $w \in \Gamma^*$ such that there is an accepting run of $\mathfrak{A}$ on w. Note that there is no set of final states, acceptance is implicit via the transition function δ by using the formulas true and false. In particular, if there is a leaf labeled (i, s) in an accepting run, then $\delta(s, w_i) = \mathsf{true}$.

Construction of the 2AFA. Let $Q = (\mathcal{T}, \Sigma, q) \in (\mathcal{ELI}, \text{CQ})$ with $\mathsf{pw}(Q) = k$, and let $\mathbf{x} = x_1 \cdots x_{\mathsf{ar}(q)}$ be the answer variables in q. We encode pairs $(\mathcal{A}, \mathbf{a})$ with $\mathcal{A}$ a Σ-ABox of pathwidth at most k and $\mathbf{a} \in \mathsf{ind}(\mathcal{A})^{\mathsf{ar}(q)}$ as words over a suitable finite alphabet Γ. Reserve a set $\mathsf{N} \subseteq \mathsf{N_I}$ of $2k+2$ individual names. Then Γ consists of all tuples $(\mathbf{b}, \mathcal{B}, \mathbf{c}, f)$, where

- $\mathcal{B}$ is a Σ-ABox with $|\mathsf{ind}(\mathcal{B})| \leq k+1$ and $\mathsf{ind}(\mathcal{B}) \subseteq \mathsf{N}$,
- $\mathbf{b}$ and $\mathbf{c}$ are tuples over $\mathsf{ind}(\mathcal{B})$ of arity at most k, and
- f is a partial function from $\mathbf{x}$ to $\mathsf{ind}(\mathcal{B})$.

Let $(\mathcal{A}, \mathbf{a})$ be a pair as described above with $\mathbf{a} = a_1 \cdots a_{\mathsf{ar}(q)}$ and let $V_1, \ldots, V_n$ be a $(j, k+1)$ path decomposition of $\mathcal{A}$. We encode $(\mathcal{A}, \mathbf{a})$ by a word $(\mathbf{b}_1, \mathcal{B}_1, \mathbf{c}_1, f_1) \cdots (\mathbf{b}_n, \mathcal{B}_n, \mathbf{c}_n, f_n)$ from Γ^n, as follows:

- As $\mathcal{B}_1$, we use a copy of $\mathcal{A}|_{V_1}$ that uses only individual names from N.
- For $1 < i \leq n$, $\mathcal{B}_i$ is a copy of $\mathcal{A}|_{V_i}$ that uses the same individual names as $\mathcal{B}_{i-1}$ on $\mathsf{ind}(\mathcal{A}|_{V_{i-1}}) \cap \mathsf{ind}(\mathcal{A}|_{V_i})$ and otherwise only individual names from $\mathsf{N} \setminus \mathsf{ind}(\mathcal{B}_{i-1})$. Since bags have size at most $k+1$, $|\mathsf{N}| = 2k+2$ individual names suffice.
- $\mathbf{b}_1 = \mathbf{c}_n$ is the empty tuple.
- For $1 < i \leq n$, $\mathbf{b}_{i-1} = \mathbf{c}_i$ is the tuple that contains every individual from $\mathsf{ind}(\mathcal{B}_{i-1}) \cap \mathsf{ind}(\mathcal{B}_i)$ exactly once, ascending in some fixed order on N.
- For $1 \leq i \leq n$, f_i is defined as follows. If V_i contains a copy a'_j of a_j, then $f_i(x_j) = a'_j$; otherwise $f_i(x_j)$ is undefined.

It is easy to see that $(\mathcal{A}, \mathbf{a})$ can be recovered from w, and in particular $\mathbf{a}$ from f. Note that different words over Γ might encode the same pair $(\mathcal{A}, \mathbf{a})$, for example because we can choose different path decompositions, and there are words over Γ that do not properly encode a pair $(\mathcal{A}, \mathbf{a})$. Neither of this is problematic for the remaining proof.

We now construct a 2AFA $\mathfrak{A}$ that accepts a word that encodes a pair $(\mathcal{A}, \mathbf{a})$ if and only if $\mathcal{A} \models Q(\mathbf{a})$. The idea is that an accepting run of the automaton has one main path on which it traverses the word from left to right, while guessing a homomorphism h from q to $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ with $h(\mathbf{x}) = \mathbf{a}$ in a stepwise fashion. The truth of all concept memberships in $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ that are necessary to realize this homomorphism is then checked by partial runs that branch off from the main path.

We now describe the set S of states of $\mathfrak{A}$. For the main path, we use states $s^g_{V,W}$ where $V \subseteq \mathsf{var}(q)$, $g : V \to \mathsf{N}$ is a partial function, and W is a subset of the binary atoms in q. Informally, the meaning of the state $s^g_{V,W}$ is that the variables from V have already been mapped to individuals in bags seen before, the binary atoms from W are already satisfied via this mapping, and g describes how variables are mapped to individuals that are in the intersection of the previous and the current bag. The initial state of $\mathfrak{A}$ is $s^g_{\emptyset,\emptyset}$ with g the empty map. We also use states s^a_A that make sure that the concept name A can be derived at $a \in \mathsf{N}$.

We have to take care of the fact that a homomorphism from q to $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ can map existentially quantified variables to anonymous individuals, which are not explicitly represented in the input word. Let $\mathcal{B}$ be a Σ-ABox. A *partial q-match* in $\mathcal{B}$ is a partial function $h : \mathsf{var}(q) \to \mathsf{ind}(\mathcal{B}) \times \{\mathsf{named}, \mathsf{anon}\}$ such that if $x, y \in \mathsf{dom}(h)$, $r(x, y) \in q$ and $h_2(x) = h_2(y) = \mathsf{named}$, then $r(h_1(x), h_1(y)) \in \mathcal{B}$, where h_1 and h_2 are the projections of h to the first and second component, respectively. Informally, a partial q-match h partially describes a homomorphism g from q to $\mathcal{U}_{\mathcal{B},\mathcal{T}}$ where $h(x) = (a, \mathsf{named})$ means that $g(x) = a$ and $h(x) = (a, \mathsf{anon})$ means that $g(x)$ is some element in the subtree below a generated by the chase. Whether a part of the query can map into the anonymous part below some individual a only depends on the type realized at a. Define a relation $\mathcal{R} \subseteq \mathsf{TP} \times 2^{\mathsf{var}(q)} \times 2^{\mathsf{var}(q)}$ by putting $(t, V_1, V_2) \in \mathcal{R}$ if and only if $V_1 \subseteq V_2$, $V_2 \cap \mathbf{x} \subseteq V_1$ and there is a homomorphism from $q|_{V_2}$ to the universal model of the ABox $\{A(a) \mid A \in t\}$ and $\mathcal{T}$ that maps precisely the variables from V_1 to the root of the (tree-shaped) universal model.

An *explanation set* for a partial q-match $h : \mathsf{var}(q) \to \mathsf{ind}(\mathcal{B}) \times \{\mathsf{named}, \mathsf{anon}\}$ is a set Z of concept assertions that uses only individuals from $\mathsf{ind}(\mathcal{B})$ and satisfies the following conditions:

1. if $h(x) = (a, \mathsf{named})$, then $\mathcal{B} \cup Z, \mathcal{T} \models A(h(x))$ for all $A(x) \in q$ and
2. if $h(x) = (a, \mathsf{anon})$, then $(\{A \mid A(a) \in Z\}, h^{-1}(a, \mathsf{named}), h_1^{-1}(a)) \in \mathcal{R}$.

Next, we describe the transition function δ. The following transitions are used for the main branch of automata runs:

$$\delta(s^g_{V,W}, (\mathbf{b}, \mathcal{B}, \mathbf{c}, f)) = \bigvee_{h \in H} \left((\mathsf{right}, s^{g_h}_{V_h,W_h}) \wedge \bigvee_{Z \in \mathcal{Z}_h} \left(\bigwedge_{A(a) \in Z} (\mathsf{stay}, s^a_A) \right) \right)$$

where H is the set of all partial q-matches h for $\mathcal{B}$ such that $\mathsf{dom}(g) \subseteq \mathsf{dom}(h)$, h_1 and g agree on the intersection of their domains, and so do h_1 and f, and where

- g_h is h_1 restricted to answer variables x_i with $h_1(x_i)$ in $\mathbf{c}$,
- $V_h = V \cup \mathsf{dom}(h)$,
- $V \cap \mathsf{dom}(h) = \mathsf{dom}(g)$,
- W_h is the union of W and all binary atoms from q that only use variables from h, and
- $\mathcal{Z}_h$ is the set of all explanation sets for h.

When the automaton reads the right end marker $\dashv$ and is in a state signifying that a complete homomorphism from q to $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ has been found, then we accept the input using the transition $\delta(s^g_{V,W}, \dashv) = \mathsf{true}$ where $V = \mathsf{var}(q)$, W is the set of all binary atoms of q, and g is the empty map.

The following transitions are used to verify the required concept memberships by checking for the existence of a suitable derivation tree (Lemma 28). Consider a state s^a_A and a symbol $(\mathbf{b}, \mathcal{B}, \mathbf{c}, f)$ such that $a \in \mathsf{ind}(\mathcal{B})$. If $A(a) \in \mathcal{B}$, we set $\delta(s^a_A, (\mathbf{b}, \mathcal{B}, \mathbf{c}, f)) = \mathsf{true}$. If a appears neither in $\mathbf{b}$ nor in $\mathbf{c}$:

$$\delta(s^a_A, (\mathbf{b}, \mathcal{B}, \mathbf{c}, f)) = \bigvee_{\substack{Z\\ \mathcal{B}\cup Z, \mathcal{T} \models A(a)}} \bigwedge_{B(b)\in Z} (\mathsf{stay}, s^b_B)$$

If a appears in $\mathbf{b}$ but not in $\mathbf{c}$:

$$\delta(s^a_A, (\mathbf{b}, \mathcal{B}, \mathbf{c}, f)) = (\mathsf{left}, s^a_A) \vee \bigvee_{\substack{Z\\ \mathcal{B}\cup Z, \mathcal{T} \models A(a)}} \bigwedge_{B(b)\in Z} (\mathsf{stay}, s^b_B)$$

If a appears in $\mathbf{c}$ but not in $\mathbf{b}$:

$$\delta(s^a_A, (\mathbf{b}, \mathcal{B}, \mathbf{c}, f)) = (\mathsf{right}, s^a_A) \vee \bigvee_{\substack{Z\\ \mathcal{B}\cup Z, \mathcal{T} \models A(a)}} \bigwedge_{B(b)\in Z} (\mathsf{stay}, s^b_B)$$

If i appears in both $\mathbf{b}$ and $\mathbf{c}$:

$$\delta(s^a_A, (\mathbf{b}, \mathcal{B}, \mathbf{c}, f)) = (\mathsf{left}, s^a_A) \vee (\mathsf{right}, s^a_A) \vee \bigvee_{\substack{Z\\ \mathcal{B}\cup Z, \mathcal{T} \models A(a)}} \bigwedge_{B(b)\in Z} (\mathsf{stay}, s^b_B)$$

Set $\delta(\cdot, \cdot) = \mathsf{false}$ for all pairs from $S \times \Gamma$ that were not mentioned.

The automaton is now defined as $\mathfrak{A} = (S, \Gamma, \delta, s_0)$.

Lemma 29. *Let $\mathcal{A}$ be an ABox of pathwidth at most k, $\mathbf{a} \in \mathsf{ind}(\mathcal{A})^{\mathsf{ar}(q)}$, and $w \in \Gamma^*$ a word that encodes $(\mathcal{A}, \mathbf{a})$. Then $\mathcal{A} \models Q(\mathbf{a})$ if and only if $w \in L(\mathfrak{A})$.*

Construction of the linear Datalog program. Since every 2AFA can be transformed into an equivalent deterministic finite automaton (DFA) [50], Lemma 29 also ensures the existence of a DFA $\mathfrak{A} = (Q, \Sigma, \delta, s_0, F)$ that accepts a word that encodes a pair $(\mathcal{A}, \mathbf{a})$ if and only if $\mathcal{A} \models Q(\mathbf{a})$. We use $\mathfrak{A}$ to construct the desired linear Datalog rewriting of Q.

The idea for the program is to guess a tuple $\mathbf{a} \in \mathsf{ind}(\mathcal{A})^{\mathsf{ar}(q)}$ up front and then verify that $\mathcal{A} \models Q(\mathbf{a})$ by simulating $\mathfrak{A}$. The program uses the states of $\mathfrak{A}$ as IDBs. Each of these IDBs can appear in any arity between $\mathsf{ar}(q)$ and $\mathsf{ar}(q)+k$ with k the pathwidth of Q—technically, this means that we have $k+1$ different IDBs for every state, but we use the same symbol for all of them since the arity will always be clear from the context. The first $\mathsf{ar}(q)$ components of each IDB are used to store the tuple $\mathbf{a}$ while the other components are used to store the individuals that occur in both of two consecutive bags of a path decomposition of $\mathcal{A}$.

Start rules: Given the ABox $\mathcal{A}$, the program starts by guessing a tuple $\mathbf{a} \in \mathsf{ind}(\mathcal{A})^{\mathsf{ar}(q)}$ using the following rule:

$$s_0(x_1, \dots, x_n) \leftarrow \top(x_1) \land \top(x_2) \land \cdots \land \top(x_n).$$

Transition rules: Consider any transition $\delta(s_1, (\mathbf{b}, \mathcal{B}, \mathbf{c}, f)) = s_2$. Let $\varphi_\mathcal{B}$ be $\mathcal{B}$ viewed as a conjunction of atoms with individual names viewed as variables and let $\mathbf{x}'$ be obtained from $\mathbf{x}$ by replacing every variable $x_i \in \mathsf{dom}(f)$ by the individual name $f(x_i) \in \mathsf{ind}(\mathcal{B})$, also here viewed as a variable. We then include the following rule:

$$s_2(\mathbf{x}', \mathbf{c}) \leftarrow s_1(\mathbf{x}', \mathbf{b}) \land \varphi_\mathcal{B}.$$

This rule says that if the DFA is in state s_1, the intersection between the last bag and the current bag is $\mathbf{b}$, and we see a homomorphic image of $\mathcal{B}$, then the DFA can transition into state s_2 and remember the tuple $\mathbf{c}$. Applying such a rule leaves the tuple $\mathbf{a}$ stored in the first $\mathsf{ar}(q)$ components unchanged, but some of the variables in $\mathbf{x}'$ can appear in $\varphi_\mathcal{B}$ to enforce that $\mathbf{a}$ is compatible with the mapping of the answer variables that is prescribed by f and used in the simulated run of $\mathfrak{A}$.

Goal rules: If $s \in F$, then include the following rule:

$$\mathsf{goal}(\mathbf{x}) \leftarrow s(\mathbf{x}).$$

Lemma 30. Π *is a rewriting of* Q.

Proof. Let $\mathcal{A}$ be a Σ-ABox and let $\mathbf{a} \in \mathsf{ind}(\mathcal{A})^{\mathsf{ar}(q)}$. First assume that $\mathcal{A} \models Q(\mathbf{a})$. Since Q is of pathwidth k, there must be a Σ-ABox $\mathcal{A}'$ of pathwidth at most k such that $\mathcal{A}' \models Q(\mathbf{a})$ and there is a homomorphism from $\mathcal{A}'$ to $\mathcal{A}$ that is the identity on $\mathbf{a}$. Let $w = w_1 \dots w_n \in \Gamma^*$ encode the pair $(\mathcal{A}', \mathbf{a})$ where $w_i = (\mathbf{b}_i, \mathcal{B}_i, \mathbf{c}_i, f_i)$. By Lemma 29, $w \in L(\mathfrak{A})$, so there is an accepting run of $\mathfrak{A}$ on w and thus we find states $s_0, \dots, s_n$ of $\mathfrak{A}$ such that $\delta(s_i, w_i) = s_{i+1}$ for $0 \leq i < n$ and s_n is an accepting state. This yields a derivation of $\Pi(\mathbf{a})$ in $\mathcal{A}'$, as follows. First, use the start rule to derive $s_0(\mathbf{a})$. For the next n steps, use the rule introduced for the transitions $\delta(s_i, w_i) = s_{i+1}$. In this way, we derive $s_n(\mathbf{a})$ since the individuals in the first $\mathsf{ar}(q)$ components do not change when using the transition rules. Because $s_n \in F$, a goal rule can be applied to derive

$\mathsf{goal}(\mathbf{a})$ and thus $\mathcal{A}' \models \Pi(\mathbf{a})$. It is well known that answers to Datalog programs are preserved under ABox homomorphisms [40] and there is a homomorphism from $\mathcal{A}'$ to $\mathcal{A}$ that is the identity on $\mathbf{a}$, we obtain $\mathcal{A} \models \Pi(\mathbf{a})$ as desired.

For the converse direction, assume that $\mathcal{A} \models \Pi(\mathbf{a})$. Then there is a derivation D of $\Pi(\mathbf{a})$ in $\mathcal{A}$. Since Π is of diameter at most k, $\mathcal{A}_D$ has pathwidth at most k. Consider the encoding of $(\mathcal{A}_D, \mathbf{a})$ as a word $w \in \Gamma^*$, based on the path decomposition induced by D in the obvious way. By construction of Π, D must use a start rule for $\mathbf{a}$, then a number of transition rules, and then a goal rule. Using the way in which these rules are constructed, it can be verified that this yields an accepting run of $\mathfrak{A}$ on w. Thus $\mathcal{A}_D \models Q(\mathbf{a})$. It remains to recall that there is a homomorphism from $\mathcal{A}_D$ to $\mathcal{A}$ that is the identity on $\mathbf{a}$, and that answers to OMQs from $(\mathcal{ELI}, \text{CQ})$ are preserved under ABox homomorphisms [3]. $\square$

5. The Trichotomy for Disconnected CQs

We now lift the trichotomy result that is provided by Theorems 9 and 17 from connected CQs to unrestricted CQs. To achieve this, we show that the complexity of an OMQ $Q = (\mathcal{T}, \Sigma, q)$ with q a disconnected CQ is precisely the complexity of the hardest OMQ $(\mathcal{T}, \Sigma, q')$ with q' a maximal connected component (MCC) of q, provided that we first have removed redundant MCCs from q.

Let $Q = (\mathcal{T}, \Sigma, q) \in (\mathcal{EL}, \text{CQ})$. We say that Q is *empty* if $\mathcal{A} \not\models Q(\mathbf{a})$ for all Σ-ABoxes $\mathcal{A}$ and tuples $\mathbf{a}$. Every empty OMQ is trivially FO-rewritable. An MCC of q is *Boolean* if it contains no answer variables. We call Q *redundant* if there is a Boolean MCC of q such that the OMQ obtained from Q by dropping that MCC from q is equivalent to Q. For proving the intended trichotomy result, it is clearly sufficient to consider OMQs that are non-empty and non-redundant. We first lift the dichotomy between AC^0 and NL.

Theorem 31. *Let $Q \in (\mathcal{EL}, \text{CQ})$. Then either*

1. *Q is FO-rewritable and thus* EVAL(Q) *is in* AC^0 *or*
2. *Q is not FO-rewritable and* EVAL(Q) *is* NL*-hard under FO reductions.*

Proof. Let $Q = (\mathcal{T}, \Sigma, q) \in (\mathcal{EL}, \text{CQ})$ be non-redundant and non-empty and let $q_1(\mathbf{x}_1), \dots, q_n(\mathbf{x}_n)$ be the MCCs of $q(\mathbf{x})$.

If every OMQ $Q_i = (\mathcal{T}, \Sigma, q_i)$ is FO-rewritable, then the conjunction of all these FO-rewritings is an FO-rewriting of Q. It thus suffices to show that otherwise, Q is NL-hard. Thus assume that some Q_i is not FO-rewritable. Since q_i is connected, EVAL(Q_i) is NL-hard under FO reductions by Theorem 9. We prove that EVAL(Q) is NL-hard under FO reductions by giving an FO reduction from EVAL(Q_i). Let $\mathcal{A}_i$ be a Σ-ABox and $\mathbf{a}_i$ a tuple from $\mathsf{ind}(\mathcal{A}_i)^{\mathsf{ar}(q_i)}$. Since Q is non-empty and non-redundant, for every $j \neq i$ we find a Σ-ABox $\mathcal{A}_j$ and a tuple $\mathbf{a}_j$ such that

1. $\mathcal{A}_j, \mathcal{T} \models q_j(\mathbf{a}_j)$ and
2. if q_i is Boolean, then $\mathcal{A}_j, \mathcal{T} \not\models q_i$.

Define $\mathcal{A}$ to be the disjoint union of $\mathcal{A}_1, \dots, \mathcal{A}_n$ and $\mathbf{a} = \mathbf{a}_1 \cdots \mathbf{a}_n$. Clearly, $\mathcal{A}$ and $\mathbf{a}$ can be defined by an FO-query, so this is an FO reduction.

We have to show that $\mathcal{A}_i \models Q_i(\mathbf{a}_i)$ iff $\mathcal{A} \models Q(\mathbf{a})$. The "$\Rightarrow$" direction is clear by construction of $\mathcal{A}$ and $\mathbf{a}$. For "$\Leftarrow$", assume that $\mathcal{A} \models Q(\mathbf{a})$. This implies $\mathcal{A} \models Q_i(\mathbf{a}_i)$, so there is a homomorphism h from q_i to $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ with $h(\mathbf{x}_i) = \mathbf{a}_i$. The universal model $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ is the disjoint union of the universal models $\mathcal{U}_{\mathcal{A}_j,\mathcal{T}}$, $1 \leq j \leq n$. Since q_i is connected, the range of h lies completely inside one of the $\mathcal{U}_{\mathcal{A}_j,\mathcal{T}}$. In fact, it must lie in $\mathcal{U}_{\mathcal{A}_i,\mathcal{T}}$. If q_i is not Boolean, this is the case because $h(\mathbf{x}_i) = \mathbf{a}_i$ is a tuple from $\mathcal{A}_i$. If q_i is Boolean, then this follows from $\mathcal{A}_j, \mathcal{T} \not\models q_i$ which implies $\mathcal{U}_{\mathcal{A}_j,\mathcal{T}} \not\models q_i$. We have thus shown that $\mathcal{U}_{\mathcal{A}_i,\mathcal{T}} \models q_i(\mathbf{a}_i)$, implying $\mathcal{A}_i \models Q_i(\mathbf{a}_i)$ by Lemma 2, as desired. We have shown that $(\mathcal{T}, \Sigma, q)$ is NL-hard. It follows that $(\mathcal{T}, \Sigma, q)$ is not FO-rewritable [46]. $\Box$

To lift the dichotomy between NL and PTime including the equivalence of NL and linear Datalog rewritability, we first give a helpful lemma about linear Datalog programs.

Lemma 32. *Let $\Pi_1, \dots, \Pi_n$ be linear Datalog programs. Then there exists a linear Datalog program Π of arity $\Sigma_{i=1}^n \mathsf{ar}(\Pi_i)$ such that for all ABoxes $\mathcal{A}$ and tuples $\mathbf{a}_1, \dots, \mathbf{a}_n$,*

$$\mathcal{A} \models \Pi_i(\mathbf{a}_i) \ \textit{ for } 1 \leq i \leq n \textit{ iff } \ \mathcal{A} \models \Pi(\mathbf{a}_1, \dots, \mathbf{a}_n)\ .$$

Proof. It suffices to give a construction for the case $n = 2$ as the general case then follows by repeatedly applying the construction. So let Π_1, Π_2 be linear Datalog programs. We assume w.l.o.g. that Π_1 and Π_2 use disjoint sets of variables. The idea to define Π is to use IDB relations (S_1, S_2), with S_i an IDB relation from Π_i for $i \in \{1, 2\}$, where the arity of (S_1, S_2) is the sum of the arities of S_1 and S_2. Π is then built so that a derivation of Π simulates derivations of both Π_1 and Π_2 by doing the first derivation step simultaneously and then alternating between the two derivations. While doing a step of one derivation, the 'state' of the other derivation is stored in the extended IDB relations. More precisely, we define program Π to contain the following rules:

- for all rule $S_i(\mathbf{x}_i) \leftarrow \varphi_i(\mathbf{x}_i, \mathbf{y}_i) \in \Pi_i$, $i \in \{1, 2\}$, such that neither φ_1 not φ_2 contains an IDB relation, the rule

$$(S_1, S_2)(\mathbf{x}_1, \mathbf{x}_2) \leftarrow \varphi_1(\mathbf{x}_1, \mathbf{y}_1) \wedge \varphi_2(\mathbf{x}_2, \mathbf{y}_2);$$

- for each rule $S_1(\mathbf{x}_1) \leftarrow \varphi(\mathbf{x}_1, \mathbf{y}_1) \in \Pi_1$ where φ contains IDB atom $S_1'(\mathbf{z}_1)$ and each IDB relation S_2 from Π_2, the rule

$$(S_1, S_2)(\mathbf{x}_1, \mathbf{z}_2) \leftarrow \varphi'(\mathbf{x}_1, \mathbf{y}_1, \mathbf{z}_2)$$

  where $\mathbf{x}_2$ is a tuple of fresh variables and φ' can be obtained from φ by replacing atom $S_1'(\mathbf{z}_1)$ with atom $(S_1', S_2)(\mathbf{z}_1, \mathbf{z}_2)$, $\mathbf{z}_2$ a tuple of fresh variables;

- the same rules as in the previous item, with the roles of Π_1 and Π_2 swapped;
- the rule
$$\mathsf{goal}(\mathbf{x}_1, \mathbf{x}_2) \leftarrow (\mathsf{goal}, \mathsf{goal})(\mathbf{x}_1, \mathbf{x}_2).$$

It can be verified that $\mathcal{A} \models \Pi(\mathbf{a}_1, \mathbf{a}_2)$ iff both $\mathcal{A} \models \Pi_1(\mathbf{a}_1)$ and $\mathcal{A} \models \Pi_2(\mathbf{a}_2)$, for all Σ-ABoxes $\mathcal{A}$ and tuples $\mathbf{a}_1$, $\mathbf{a}_2$. □

Theorem 33. *Let $Q \in (\mathcal{EL}, \text{CQ})$. The following are equivalent (assuming $\text{NL} \neq \text{PTime}$):*

1. *Q has bounded pathwidth;*
2. *Q is linear Datalog rewritable;*
3. *$\text{EVAL}(Q)$ is in NL.*

If these conditions do not hold, then $\text{EVAL}(Q)$ is PTime-hard under FO reductions.

Proof. The equivalence of (1) and (2) has been shown in Proposition 26 and the implication (2) $\Rightarrow$ (3) is clear. To finish the proof, we show that if (2) does not hold, then $\text{EVAL}(Q)$ is PTime-hard, proving the implication (3) $\Rightarrow$ (2) as well as the last sentence of the theorem.

Let $Q = (\mathcal{T}, \Sigma, q) \in (\mathcal{EL}, \text{CQ})$ and assume that Q is not rewritable into linear Datalog. As before, we can assume Q to be non-empty non-redundant. Let $q_1(\mathbf{x}_1), \dots, q_n(\mathbf{x}_n)$ be the connected components of $q(\mathbf{x})$. By Theorem 17, every OMQ $Q_i = (\mathcal{T}, \Sigma, q_i)$ is either rewritable into linear Datalog or PTime-hard. By Lemma 32, every $(\mathcal{T}, \Sigma, q_i)$ being rewritable into linear Datalog implies that also $(\mathcal{T}, \Sigma, q)$ is linear Datalog rewritable. Since this is not the case, there must be some Q_i that is not rewritable into linear Datalog and thus PTime-hard.

It is now possible to show PTime-hardness of $\text{EVAL}(Q)$ by an FO reduction from $\text{EVAL}(Q_i)$, exactly as in the proof of Theorem 31. □

Remark. Recall that we are working with CQs that do not admit equality throughout this paper. However, the trichotomy result established in this section can easily be generalized to $(\mathcal{EL}, \text{CQ}^=)$ where $\text{CQ}^=$ denotes the class of CQs with equality. This is due to the observation that for every OMQ $Q \in (\mathcal{EL}, \text{CQ}^=)$, there is an OMQ $Q' \in (\mathcal{EL}, \text{CQ})$ such that there is an FO reduction from $\text{EVAL}(Q)$ to $\text{EVAL}(Q')$ and vice versa. Let $Q = (\mathcal{T}, \Sigma, q) \in (\mathcal{EL}, \text{CQ}^=)$. To construct Q', we simply eliminate the equality atoms in q by identifying variables, ending up with a CQ q' without equality atoms that might have lower arity than q. It is easy to see that the required FO reductions exist, which essentially consist of dropping resp. duplicating components from answer tuples.

6. Width Hierarchy for Linear Datalog Rewritability

The width of the linear Datalog rewritings constructed in Section 4 depends on $\mathsf{pw}(Q)$, so if Q has high pathwidth, then we end up with a linear Datalog rewriting of high width. We aim to show that this is unavoidable, that is, there is no constant bound on the width of linear Datalog rewritings of OMQs from $(\mathcal{EL}, \text{CQ})$ and in fact not even from $(\mathcal{EL}, \text{AQ})$. It is interesting to contrast this with the fact that every OMQ from $(\mathcal{EL}, \text{CQ})$ can be rewritten into a *monadic* (non-linear) Datalog program [5]. Our result strengthens a result by [22] who establish an analogous statement for CSPs. This does not imply our result: While every OMQ from $(\mathcal{EL}, \text{AQ})$ is equivalent to a CSP up to complementation [3], the converse is false and indeed the CSPs used by Dalmau and Krokhin are not equivalent to an OMQ from $(\mathcal{EL}, \text{AQ})$. Our main aim is to prove the following.

Theorem 34. *For every $k > 0$, there is an OMQ $Q_k \in (\mathcal{EL}, \text{AQ})$ that is rewritable into linear Datalog, but not into a linear Datalog program of width k.*

When constructing the OMQs $Q_1, Q_2, \ldots$ for Theorem 34, we would like the ABoxes in $\mathcal{M}_{Q_k}$ to contain *more and more branching* for increasing values of k. Note that since we only work with $(\mathcal{EL}, \text{AQ})$, $\mathcal{M}_{Q_k}$ consists of tree-shaped ABoxes rather than of pseudo tree-shaped ones. Intuitively, if the ABoxes from $\mathcal{M}_{Q_k}$ branch a lot, then a linear Datalog rewriting of Q_k needs large width to simultaneously collect information about many different branches. However, we want Q_k to be linear Datalog rewritable so by Proposition 18 there must be a constant upper bound on the branching of the ABoxes from $\mathcal{M}_{Q_k}$, for each $k \geq 1$. We thus construct Q_k such that the branching number $\mathsf{br}(\mathcal{A})$ of $\mathcal{A}$ is at least k for all $\mathcal{A} \in \mathcal{M}_{Q_k}$ while for every $n \geq k$, $\mathcal{M}_{Q_k}$ contains an ABox $\mathcal{A}_k^n$ that takes the form of a tree of outdegree 2 and of depth n such that $\mathsf{br}(\mathcal{A}_k^n) = k$ and $\mathcal{A}_k^n$ has the maximum number of leaves that any such ABox can have. To make the latter more precise, let $\ell_d^k(n)$ denote the maximum number of leaves in any tree that has degree d, depth n, and does not have the full binary tree of depth $k+1$ as a minor, $d, k, n \geq 0$. We then want $\mathcal{A}_k^n$ to have exactly $\ell_2^k(n)$ leaves, which ensures that it is maximally branching.

We now construct Q_k. For every $k \geq 1$, let $Q_k = (\mathcal{T}_k, \Sigma, A_k(x))$ where $\Sigma = \{r, s, t, u\}$ and

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

Each concept name A_i 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. The concept inclusions $\exists x.B_{x,i} \sqsubseteq B_{x,i}$

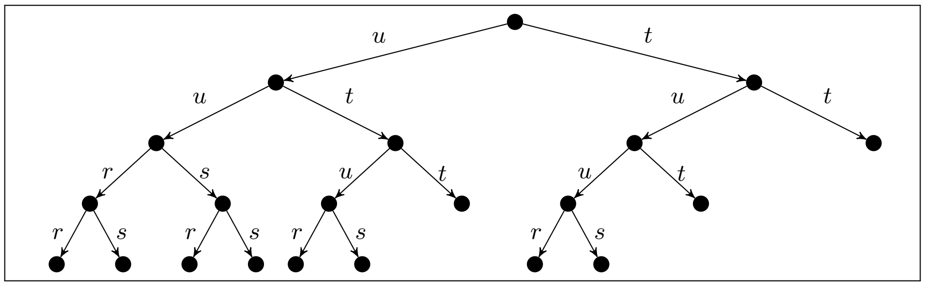

Figure 7: The ABox $\mathcal{A}_2^4$, which has depth 4, branching number 2 and lies in $\mathcal{M}_{Q_2}$. Since $4 > 2$, $\mathcal{A}_2^4$ is composed of one copy of $\mathcal{A}_2^3$ and one copy of $\mathcal{A}_1^3$ and a new tu-individual as root. This ABox has 11 leaves, which is the largest number of leaves that a binary tree of depth 4 can have, unless it contains the full binary tree of depth 3 as a minor.

in $\mathcal{T}_k$ ensure that Q_k is closed under subdivisions of ABoxes, that is, if $\mathcal{A} \in \mathcal{M}_{Q_k}$ 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 $\mathcal{A} \models Q_k(a)$ if and only if $\mathcal{A}' \models Q_k(a)$ for all $a \in \mathsf{ind}(\mathcal{A})$. Informally spoken, subdivision will allow us to assume that every (connected) rule body in a linear Datalog rewriting can only 'see' a single branching.

To provide a better understanding of the four role names used, we now explicitly define the ABoxes $\mathcal{A}_k^n$ mentioned above. We refer to non-leaf individuals by the combination of role names of their outgoing edges, e.g. an rs-individual is an individual that has one outgoing r-edge, one outgoing s-edge and no other outgoing edges. Let $1 \leq k \leq n$.

- If $k = n$, then $\mathcal{A}_k^n$ is the full binary tree of depth n, where every non-leaf is an rs-individual.
- If $k < n$, then $\mathcal{A}_k^n$ consists of a root that is a tu-individual where the t-successor is the root of a copy of $\mathcal{A}_{k-1}^{n-1}$ and the u-successor is the root of a copy of $\mathcal{A}_k^{n-1}$. Here, for consistency, we define $\mathcal{A}_0^n := \emptyset$.

As an example, Figure 7 shows $\mathcal{A}_2^4$.

The following lemma states some basic properties of the ABoxes $\mathcal{A}_k^n$ and the OMQs Q_k. All three points are proved by induction on n.

Lemma 35. *For all* $1 \leq k \leq n$,

1. $\mathcal{A}_k^n \in \mathcal{M}_{Q_k}$;
2. $\mathsf{br}(\mathcal{A}_k^n) = k$;
3. $\mathcal{A}_k^n$ *has exactly* $\ell_2^k(n)$ *leaves.*

The following lemma establishes the first part of Theorem 34.

Lemma 36. *For every* $k \geq 1$, Q_k *is rewritable into linear Datalog.*

To prove Lemma 36, we show that $\mathsf{br}(Q_k) = k$, which by Proposition 18 implies that Q_k has bounded pathwidth which by Theorem 17 implies that Q_k is rewritable into linear Datalog. The proof that $\mathsf{br}(Q_k) = k$ relies on a careful analysis of the types $\mathsf{tp}_{\mathcal{A},\mathcal{T}_k}(a)$ for ABoxes $\mathcal{A} \in \mathcal{M}_{Q_k}$. Note 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 all $x \in \{r, s, t, u\}$. An important obervation is that for all $a \in \mathsf{ind}(\mathcal{A})$, $\mathsf{br}(\mathcal{A}^a) = i$ if and only if i is the largest integer such that $A_i \in \mathsf{tp}_{\mathcal{A},\mathcal{T}_k}(a)$.

The proof of the second part of Theorem 34 relies on an estimate of $\ell_d^k(n)$, namely on the fact that $\ell_d^k(n)$ as a function of n grows like a polynomial of degree k. This is established by the following lemma.

Lemma 37. *$(d-1)^k(n-k)^k \leq \ell_d^k(n) \leq (k+1)(d-1)^k n^k$ for all $d, k \geq 0$ and $n \geq 2k$.*

To show that linear Datalog rewritings of the family of OMQs Q_k require unbounded width, we first show that they require unbounded diameter and then proceed by proving that the width of rewritings cannot be significantly smaller than the required diameter. More precisely, we show the former on an infinite family $\mathfrak{C}_0, \mathfrak{C}_1, \dots$ of classes of ABoxes of restricted shape. Trivially, if linear Datalog rewritings require unbounded diameter on any restricted class of ABoxes, then they also require unbounded diameter on the class of all ABoxes. For all $m \geq 0$, we choose $\mathfrak{C}_m$ to be the class of all forest-shaped Σ-ABoxes in which the distance between any two branching individuals exceeds m, 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 subdivisions of edges in ABoxes and by Point 1 of Lemma 35, for all $m, n, k \geq 1$ with $k \leq n$ we can find an ABox in $\mathcal{M}_{Q_k} \cap \mathfrak{C}_m$ that can be obtained from $\mathcal{A}_k^n$ by subdividing edges. The concrete choice of $\mathfrak{C}_0, \mathfrak{C}_1, \dots$ helps to achieve the second step that relates the width and diameter of linear Datalog rewritings. We give more details later on.

The idea for proving that any linear Datalog rewriting of Q_k requires high diameter is then as follows. We assume to the contrary that there is a constant c such that for every OMQ Q_k, there is a linear Datalog rewriting Π_k of Q_k of diameter c. Then consider the derivation D of $\Pi_k(a)$ in some ABox $\mathcal{A}_k^n$ with root a, where n and $k > c$ are chosen to be sufficently large. Recall from Section 4.2 that each derivation D gives rise to an ABox $\mathcal{A}_D$, and that the pathwidth of $\mathcal{A}_D$ is bounded by the diameter c of Π. We use a careful analysis to show that then $\mathcal{A}_D$ contains as a subset a tree-shaped ABox of depth n that has as many leaves as $\mathcal{A}_k^n$ and thus by Lemma 37 contains as a minor a full binary tree of depth k when n is chosen sufficiently large. When k is chosen large enough, this implies that $\mathcal{A}_D$ has pathwidth larger then c, which is a contradiction.

Proposition 38. *Let $m \geq 0$. For all $k \geq 1$, Q_{2k+3} is not rewritable into a linear Datalog program of diameter k on the class of ABoxes $\mathfrak{C}_m$.*

We are now ready to show that the width of linear Datalog rewritings of the OMQs Q_k must increase with k. The idea is as follows. Assume to the contrary that there is a constant c such that for every OMQ Q_k, there is a linear Datalog rewriting Π_k of Q_k of width c. Choose k large enough and consider Π_k as a rewriting of Q_k on the class of ABoxes $\mathfrak{C}_m$, m the diameter of Π_k. Since the distance between any two branching individuals in $\mathfrak{C}_m$ exceeds the diameter of Π_k, every connected component of a rule body in Π_k can 'see' only a single branching individual in every ABox from $\mathfrak{C}_m$. This allows us to modify Π_k into a linear Datalog rewriting Π'_k of Q_k on $\mathfrak{C}_m$ (but not on the class of all Σ-ABoxes) that has a small diameter and thus contradicts Proposition 38. Each rule body being able to see only a single branching individual is crucial here since branching forces us to increase the width when making the diameter of rule bodies smaller. The modification is in three steps. In the first step, we normalize the shape of rule bodies so that they take the form of a forest with at most one branching point (exploiting that we work on $\mathfrak{C}_m$). In the second step, we bound the number of disconnected components in the rule body, and in the third step we actually bound the diameter by replacing each rule with a collection of rules. Note that the class $\mathfrak{C}_m$ consists of ABoxes that take the shape of forests rather than trees to deal with disconnected rule bodies in Π_k. Clearly, the following implies the second part of Theorem 34.

Proposition 39. *Let $\ell \geq 1$. $Q_{8\ell+13}$ is not rewritable into a linear Datalog program of width ℓ.*

7. Decidability and Complexity

We study the meta problems that emerge from the results in the previous sections such as deciding whether a given OMQ is in NL, PTime-hard, or rewritable into linear Datalog. We show that all these problems are ExpTime-complete. Apart from applying and adapting known lower and upper bounds, the central ingredient is giving a single exponential time decision procedure for deciding whether an OMQ from $(\mathcal{EL}, \text{conCQ})$ has the ability to simulate PSA. We start with lower bounds, which hold already for $(\mathcal{EL}, \text{AQ})$.

Theorem 40. *The following problems are* ExpTime*-hard: given an OMQ $Q \in (\mathcal{EL}, \text{AQ})$,*

(1) is Q FO-rewritable?

(2) is Q rewritable into linear Datalog?

(3) is EVAL$(Q) \in \text{AC}^0$*?*

(4) is EVAL$(Q) \in \text{NL}$*? (unless* NL = PTime*)*

(5) is EVAL(Q) NL*-hard?*

(6) is EVAL(Q) PTime*-hard?*

Proof. ExpTime-hardness of (1) is proved in (the appendix of) [11]. By our Theorem 31, (1) and (3) are equivalent, so (3) is also ExpTime-hard.

For (2), (4), (5) and (6), we analyse the mentioned hardness proof from [11] a little closer. The proof is by a reduction from 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 if and only if M accepts w. A careful inspection of the construction of Q and of the "$\Leftarrow$" part of the proof reveals that

($*$) if M accepts w, then Q is unboundedly branching, thus (by Proposition 18 and Theorem 33) not linear Datalog rewritable, PTime-hard, NL-hard and not in NL (unless NL = PTime).

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

- in AC^0 and thus in NL and neither NL-hard nor PTime-hard;
- linear Datalog rewritable (since every FO-rewritable OMQ from $(\mathcal{EL}, \text{AQ})$ is rewritable into a UCQ [11]).

The stated hardness results for (2), (4), (5) and (6) follow. □

The following theorem summarizes the corresponding upper bounds.

Theorem 41. *The following problems can be decided in* ExpTime*: given an OMQ $Q \in (\mathcal{EL}, \text{CQ})$,*

(1) is Q FO-rewritable?

(2) is Q rewritable into linear Datalog?

(3) is eval$(Q) \in \text{AC}^0$*?*

(4) is eval$(Q) \in \text{NL}$*? (unless* NL = PTime*)*

(5) is eval(Q) NL*-hard?*

(6) is eval(Q) PTime*-hard? (unless* NL = PTime*)*

In [18], it was shown that (1) is in ExpTime. By Theorem 31, the same algorithm decides (3) and (5). By Theorem 33, the remaining (2), (4) and (6) come down to a single decision problem. We may thus choose to concentrate on deciding (6). We first argue that it suffices to decide (6) for OMQs from $(\mathcal{EL}, \text{conCQ})$, that is, to restrict our attention to connected CQs.

Let $Q = (\mathcal{T}, \Sigma, q) \in (\mathcal{EL}, \text{CQ})$. To decide whether eval$(Q)$ is PTime-hard (unless NL = PTime), we can first check whether Q is empty. This can be done in ExpTime [18] and an empty OMQ is clearly not PTime-hard. Otherwise, we make Q non-redundant (see Section 5) by exhaustively removing Boolean MCCs that cause non-redundancy. This can also be done in exponential time since containment of OMQs from $(\mathcal{EL}, \text{CQ})$ is in ExpTime [18]. The resulting

OMQ $Q' = (\mathcal{T}, \Sigma, q')$ is equivalent to Q and as seen in the proof of Theorem 33, $\text{EVAL}(Q')$ is PTIME-hard if and only if there is an MCC q'_i of q' such that $(\mathcal{T}, \Sigma, q'_i) \in (\mathcal{EL}, \text{conCQ})$ is PTIME-hard.

It thus remains to show how (6) can be decided in EXPTIME for OMQs $Q \in (\mathcal{EL}, \text{conCQ})$. For such Q, it follows from Propositions 18, 23, 24, and 26 and Theorem 17 that (6) is equivalent to deciding whether Q has the ability to simulate PSA.

In the remainder of this section, we reduce the question whether a given OMQ $Q \in (\mathcal{EL}, \text{conCQ})$ has the ability to simulate PSA to the (non-)emptiness problem of two-way alternating parity tree automata (TWAPA), which is EXPTIME-complete. In fact, we construct a TWAPA that accepts precisely those (encodings of) pseudo tree-shaped ABoxes that witness the ability to simulate PSA and then check non-emptiness.

Two-way alternating parity tree automata (TWAPA). A *tree* is a non-empty (and potentially infinite) set $T \subseteq \mathbb{N}^*$ closed under prefixes. We say that T is *m-ary* if $T \subseteq \{1, \dots, m\}^*$. For an alphabet Γ, a *Γ-labeled 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 $x \in P$ with $|x| = i$.

Definition 42 (TWAPA)**.** A *two-way alternating parity automaton (TWAPA) 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 $\mathsf{tran}(\mathfrak{A}) = \{\langle i \rangle s,\ [i]s \mid -1 \leq i \leq m \text{ and } s \in S\}$ 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*.

A TWAPA with alphabet Γ accepts a set of Γ-labeled trees. 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, and $\langle -1 \rangle s$ indicates moving to the predecessor of the current node, which is then required to exist. Transitions $[i]s$ mean that a copy of the automaton in state s is sent on the relevant successor if that successor exists (which is not required).

Definition 43 (Run, Acceptance)**.** Let $\mathfrak{A} = (S, \Gamma, \delta, s_0, c)$ be a TWAPA and (T, L) a finite Γ-labeled tree. A *configuration* is a pair from $T \times S$. A *run* of $\mathfrak{A}$ on (T, L) from the configuration γ is a $T \times S$-labeled tree (T_r, r) such that the following conditions are satisfied:

1. $r(\varepsilon) = \gamma$;
2. if $y \in T_r$, $r(y) = (x, s)$, and $\delta(s, L(x)) = \varphi$, 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 $x \cdot i \in T$ and there is a node $y \cdot j \in T_r$ such that $r(y \cdot j) = (x \cdot i, s')$;

(b) if $[i]s' \in S$ and $x \cdot i \in T$, then there is a node $y \cdot j \in T_r$ such that $r(y \cdot j) = (x \cdot i, s')$.

We say that (T_r, r) is *accepting* if on all infinite paths of T_r, the maximum priority that appears infinitely often on this path is even. A finite Γ-labeled tree (T, L) is *accepted* by $\mathfrak{A}$ if there is an accepting run of $\mathfrak{A}$ on (T, L) from the configuration (ε, s_0). We use $L(\mathfrak{A})$ to denote the set of all finite Γ-labeled tree accepted by $\mathfrak{A}$.

It is known (and easy to see) that TWAPAs are closed under complementation and intersection, and that these constructions involve only a polynomial blowup [51]. In particular, complementation boils down to dualizing the transitions and increasing all priorities by one. It is also known that their emptiness problem can be solved in time single exponential in the number of states and highest occurring priority, and polynomial in all other components of the automaton [52]. In what follows, we shall generally only explicitly analyze the number of states of a TWAPA, but only implicitly take care that all other components are of the allowed size for the complexity result that we aim to obtain.

Encoding pseudo tree-shaped ABoxes. To check the ability to simulate PSA using TWAPAs, we build one TWAPA $\mathfrak{A}_{t_0,t_1}$ for every pair (t_0, t_1) of $\mathcal{T}$-types. An input tree for the TWAPA encodes a tuple $(\mathcal{A}, \mathbf{a}, b, c, d)$ of a pseudo tree-shaped ABox $\mathcal{A}$ of core size at most $|q|$, a tuple $\mathbf{a}$ from the core and three distinguished individuals b, c and d. The TWAPA $\mathfrak{A}_{t_0,t_1}$ should accept a tree that encodes $(\mathcal{A}, \mathbf{a}, b, c, d)$ if and only if $t_0, t_1, \mathcal{A}, \mathbf{a}, b, c$ and d witness the ability to simulate PSA according to Definition 20. The ExpTime decision procedure is obtained by checking whether at least one of the (exponentially many) $\mathfrak{A}_{t_0,t_1}$ accepts a non-empty language.

We encode tuples $(\mathcal{A}, \mathbf{a}, b, c, d)$ as finite $(|\mathcal{T}| \cdot |q|)$-ary $\Gamma_\varepsilon \cup \Gamma_N$-labeled trees, where Γ_ε is the alphabet used for labeling the root node and Γ_N is for non-root nodes. These alphabets are different because the root of a tree encodes the entire core of a pseudo tree-shaped ABox whereas each non-root node represents a single non-core individual.

Let $\mathsf{C_{core}} \subseteq \mathsf{N_I}$ be a fixed set of size $|q|$. Define Γ_ε to be the set of all tuples $(\mathcal{B}, \mathbf{a})$, where $\mathcal{B}$ is a Σ-ABox that only uses individual names from $\mathsf{C_{core}}$ and $\mathbf{a}$ a tuple of length $\mathsf{ar}(q)$ from $\mathsf{ind}(\mathcal{B})$. Let ROL be the set of roles that appear in $\mathcal{T}$ or Σ and let CN be the set of all concept names that appear in $\mathcal{T}$ or Σ. Let $S = \mathsf{ROL} \cup \mathsf{CN} \cup \mathsf{C_{core}} \cup \{b, c, d\}$. The alphabet Γ_N is defined to be the set of all subsets of S that contain exactly one element from ROL, at most one element from $\mathsf{C_{core}}$ and at most one element of $\{b, c, d\}$. We call a $(\Gamma_\varepsilon \cup \Gamma_N)$-labeled tree (T, L) *proper* if

- $L(\varepsilon) \in \Gamma_\varepsilon$ and $L(x) \in \Gamma_N$ for all $x \neq \varepsilon$,
- $L(x)$ contains an element of $\mathsf{C_{core}}$ if and only if x is a child of ε; moreover, any such element must occur in $\mathsf{ind}(L(\varepsilon))$;

- there is exactly one node $x_b \in T$ with $b \in L(x_b)$, exactly one node $x_c \in T$ with $c \in L(x_c)$ and exactly one node $x_d \in T$ with $d \in L(x_d)$,
- the nodes x_c and x_d are incomparable descendants of x_b,
- the nodes ε, x_b, x_c and x_d have pairwise distance more than $|q|$ from each other.

A proper tree (T, L) encodes a tuple $(\mathcal{A}, \mathbf{a}, b, c, d)$ in the following way. If $L(\varepsilon) = (\mathcal{B}, \mathbf{a})$, then

$$\begin{aligned} \mathcal{A} \;\; = \;\; & \mathcal{B} \cup \{A(x) \mid A \in L(x), x \neq \varepsilon\} \\ & \cup \{r(a, x) \mid \{a, r\} \subseteq L(x) \text{ with } a \in \mathsf{C}_{\mathsf{core}}\} \\ & \cup \{r(x, y) \mid r \in L(y), y \text{ is a child of } x, x \neq \varepsilon\} \end{aligned}$$

with x_b replaced with b, x_c with c, and x_d with d. Note that $a \in L(x) \in \Gamma_N$ indicates that x is a successor of core individual $a \in \mathsf{ind}(L(\varepsilon))$. It is easy to see that there is a TWAPA $\mathfrak{A}_{\mathsf{proper}}$ that accepts a $(\Gamma_\varepsilon \cup \Gamma_N)$-labeled tree if and only if it is proper. Details are omitted.

From now on, let t_0 and t_1 be fixed. We construct the TWAPA $\mathfrak{A}_{t_0,t_1}$ as the intersection of $\mathfrak{A}_{\mathsf{proper}}$ and TWAPAs $\mathfrak{A}_1, \dots, \mathfrak{A}_6$ where each $\mathfrak{A}_k$ accepts a proper input tree (T, L) if and only if the tuple $(\mathcal{A}, \mathbf{a}, b, c, d)$ encoded by (T, L) satisfies Condition (k) from Definition 20. We make sure that all $\mathfrak{A}_k$ can be constructed in exponential time and have only polynomially many states in the size of Q.

Derivation of concept names. We start with describing a capability of TWAPAs that most of the $\mathfrak{A}_k$ will make use of, namely to check whether a concept name is derived by $\mathcal{T}$ at an individual of the ABox encoded by the input tree. We construct a TWAPA $\mathfrak{A}_{\mathsf{derive}}$ that is capable of performing such a check and that we will use as a building block for defining the TWAPAs $\mathfrak{A}_1, \dots, \mathfrak{A}_6$.

Recall from Section 4.1 that in the context of the ability to simulate PSA, we had generally assumed that the $\mathcal{EL}$-TBox $\mathcal{T}$ has been extended by certain $\mathcal{ELI}$-concept inclusions. We thus consider $\mathcal{ELI}$-TBoxes in the construction of $\mathfrak{A}_{\mathsf{derive}}$. The set of states of $\mathfrak{A}_{\mathsf{derive}}$ includes among others

$$\{d_A \mid A \in \mathsf{CN}\} \cup \{d^a_A \mid A \in \mathsf{CN} \wedge a \in \mathsf{C}_{\mathsf{core}}\}$$

and $\mathfrak{A}_{\mathsf{derive}}$ is built such that the following holds:

- if $\mathfrak{A}_{\mathsf{derive}}$ is started on a proper input tree encoding $(\mathcal{A}, \mathbf{a}, b, c, d)$ from a configuration (a, d_A), then it accepts if and only if $\mathcal{A}, \mathcal{T} \models A(a)$;
- if $\mathfrak{A}_{\mathsf{derive}}$ is started on a proper input tree encoding $(\mathcal{A}, \mathbf{a}, b, c, d)$ from a configuration (ε, d^a_A), then it accepts if and only if $\mathcal{A}, \mathcal{T} \models A(a)$.

Note that the two cases correspond to the treatment of non-core individuals and core individuals, respectively. The exact construction of $\mathfrak{A}_{\mathsf{derive}}$ is given in the appendix. It makes use of Lemma 28, which states that $\mathcal{A}, \mathcal{T} \models A(a)$ if and only if there is a derivation tree for $A(a)$. In fact, we construct the transitions of $\mathfrak{A}_{\mathsf{derive}}$ in a straightforward way so as to check the existence of a derivation tree.

Construction of $\mathfrak{A}_1$. This TWAPA checks whether $\mathcal{A} \models Q(\mathbf{a})$. Due to Condition 5 of the ability to simulate PSA, we only need to check whether $\mathcal{A} \models Q(\mathbf{a})$ via a core close homomorphism. The existence of such a homomorphism, in turn, depends only on the core of $\mathcal{A}$ and on the concept names that are derived at core individuals. We next make this precise. Recall that $\mathcal{T}$ contains the CI $C \sqsubseteq A_C$ for every CQ $C \in \mathsf{trees}(q)$ viewed as an $\mathcal{EL}$-concept. Let $(\mathcal{B}, \mathbf{a}) \in \Sigma_\epsilon$. A *completion* of $\mathcal{B}$ is a set M of assertions $A(a)$ with $A \in \mathsf{CN}$ and $a \in \mathsf{C}_{\mathsf{core}}$. Every completion M of $\mathcal{B}$ gives rise to an extension $\mathcal{B}_M$ of $\mathcal{B}$ obtained as follows:

- add every concept assertion from M;
- if $A_C(a) \in M$ with $C \in \mathsf{trees}(q)$, then add a disjoint copy of $\mathcal{A}_C$, glueing the root to a.

We say that M is *matching* if $\mathcal{B}_M \models q(\mathbf{a})$ and use $\mathsf{MComp}(\mathcal{B})$ to denote the set of all matching completions of $\mathcal{B}$. Clearly, there are exponentially many completions for every $\mathcal{B}$ and given $\mathcal{B}$ and a completion M of $\mathcal{B}$, it can be decided in exponential time (actually in NP) whether M is matching. We next observe the following.

Lemma 44. *Let $\mathcal{A}$ be a pseudo tree-shaped Σ-ABox with core $\mathcal{B}$, $\mathsf{ind}(\mathcal{B}) \subseteq \mathsf{C}_{\mathsf{core}}$, and $\mathbf{a}$ a tuple from $\mathsf{ind}(\mathcal{B})$. Then*

1. *if $\mathcal{A} \models Q(\mathbf{a})$ via a core close homomorphism, then the following is a matching completion of $\mathcal{B}$: $\{A(a) \mid a \in \mathsf{ind}(\mathcal{B})$ and $\mathcal{A}, \mathcal{T} \models A(a)\}$;*
2. *if there exists a matching completion M of $\mathcal{B}$ such that $\mathcal{A}, \mathcal{T} \models A(a)$ for all $A(a) \in M$, then $\mathcal{A} \models Q(\mathbf{a})$.*

The strategy of TWAPA $\mathfrak{A}_1 = (S_{\mathsf{derive}} \cup \{s_0\}, \Gamma_\varepsilon \cup \Gamma_N, \delta, s_0, c)$ is now as follows. If $(\mathcal{B}, \mathbf{a}) \in \Gamma_\varepsilon$ is the label of the root of the input tree, then guess a matching completion M for $\mathcal{B}$ and use $\mathfrak{A}_{\mathsf{derive}}$ to check whether all $A(a) \in M$ can be derived in $\mathcal{A}$. The transition function δ for states in S_{derive} is thus defined as before, and additionally for all $(\mathcal{B}, \mathbf{a}) \in \Sigma_\epsilon$ we set

$$\delta(s_0, (\mathcal{B}, \mathbf{a})) = \bigvee_{M \in \mathsf{MComp}(\mathcal{B})} \bigwedge_{A(a) \in M} d_A^a.$$

Construction of $\mathfrak{A}_2$. This TWAPA checks that $t_1 = \mathsf{tp}_{\mathcal{A},\mathcal{T}}(b) = \mathsf{tp}_{\mathcal{A},\mathcal{T}}(c) = \mathsf{tp}_{\mathcal{A},\mathcal{T}}(d)$. Using $\mathfrak{A}_{\mathsf{derive}}$ and its dualization, this is straightforward: send a copy to the nodes in the input tree that are marked with b, c, and d, and then use $\mathfrak{A}_{\mathsf{derive}}$ to make sure that all $A \in t_1$ are derived there and the dual of $\mathfrak{A}_{\mathsf{derive}}$ to make sure that no $A \notin t_1$ is derived there.

Construction of $\mathfrak{A}_3$. This TWAPA checks that $\mathcal{A}_b \cup t_0(b), \mathcal{T} \not\models q(\mathbf{a})$. We have to prevent homomorphisms from q to $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ that are core close as well as homomorphisms that are not core close. The latter is only possible if q is Boolean and treeifiable, as every homomorphism from a CQ that is not treeifiable into $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ hits the core. $\mathfrak{A}_3$ thus contains two sub-TWAPAs with disjoint sets of states, and the initial state of $\mathfrak{A}_3$ is associated with a transition $s_1 \wedge s_2$, s_i the starting state of the i-th sub-TWAPA.

The first sub-TWAPA is constructed in the same way as $\mathfrak{A}_1$ except that it uses a modified version of $\mathfrak{A}_{\mathsf{derive}}$ that checks derivations on $\mathcal{A}_b \cup t_0(b)$ instead of $\mathcal{A}$, by making sure that all concept names from t_0 are assumed to be true at the node of the input tree marked with b and disregarding the subtree below. In addition, we dualize the constructed sub-TWAPA at the end.

The second sub-TWAPA works as follows. If q is not Boolean or not treeifiable, then it accepts every input. Otherwise, q viewed as an $\mathcal{EL}$-concept C_q is in $\mathsf{trees}(q)$ and $C_q \sqsubseteq A_{C_q} \in \mathcal{T}$. Thus it suffices for $\mathfrak{A}_5$ to check that A_{C_q} is neither derived at any non-core individual a nor at an anonymous individual below a non-core individual. Let M_Q be the set of all $\mathcal{T}$-types t with $t(a), \mathcal{T} \models \exists x\, C_q(x)$, which can be computed in exponential time using the fact that CQ evaluation in $(\mathcal{ELI}, \text{CQ})$ is in ExpTime [53]. The TWAPA makes sure that no type from M_Q is realized at a non-core individual, using $\mathfrak{A}_{\mathsf{derive}}$ and its dualization. Here, we again mean the modified version of $\mathfrak{A}_{\mathsf{derive}}$ also used by the first sub-TWAPA.

Construction of $\mathfrak{A}_4$. $\mathfrak{A}_4$ checks that $\mathsf{tp}_{\mathcal{A}_c \cup t_0(c),\mathcal{T}}(b) = \mathsf{tp}_{\mathcal{A}_d \cup t_0(d),\mathcal{T}}(b) = t_0$. It is constructed similarly to $\mathfrak{A}_2$.

Construction of $\mathfrak{A}_5$. This TWAPA checks that every homomorphism from q to $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ is core close. It is only required if q is Boolean. The construction is the same as for the second sub-TWAPA of $\mathfrak{A}_3$, except that we work with the original version of $\mathfrak{A}_{\mathsf{derive}}$.

Construction of $\mathfrak{A}_6$. This TWAPA checks that b, c and d all have the same ancestor path up to length $|q|$. It is only required if q is Boolean. The idea is to guess an ancestor path $r_1 r_2 \ldots r_{|q|}$ up front and then verify that b, c and d all have this ancestor path. To achieve this using only polynomially many states, the guessed path is not stored in a single state. Instead, we use $|q|$ copies of the automaton, the i-th copy guessing states of the form $s_{i,r}$ which stands for $r_i = r$. This copy then further spawns into three copies that visit the nodes labeled b, c, and d, travels upwards from there $n - i$ steps, and checks that the node label there contains r.

8. Conclusion

We have established a complexity trichotomy between AC^0, NL, and PTime for OMQs from $(\mathcal{EL}, \text{CQ})$. We have also proved that linear Datalog rewritability coincides with OMQ evaluation in NL and that deciding all these (and related) properties is ExpTime complete with the lower bounds applying already to $(\mathcal{EL}, \text{AQ})$.

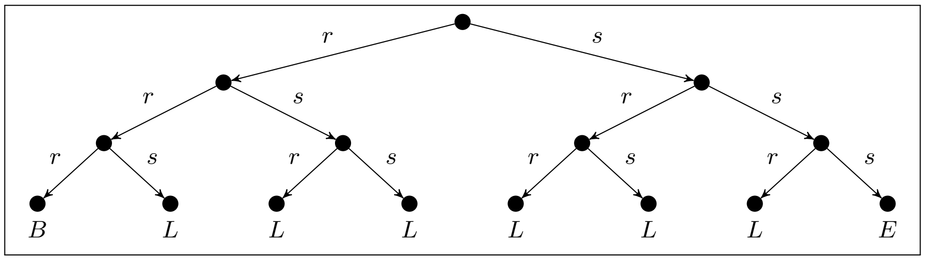

Figure 8: An ABox $\mathcal{A}$ with $\mathcal{A} \models Q(a)$, a the root of $\mathcal{A}$ and Q the OMQ from Example 45.

There are several natural directions in which our results can be generalized. One direction is to transition from CQs to unions of CQs (UCQs), that is, to consider the OMQ language $(\mathcal{EL}, \text{UCQ})$. We conjecture that the complexity trichotomy and the established equivalences between complexity classes and rewritability generalize to this case without significant additional technical difficulties. We refrained from studying this generalization in detail since it makes the proofs even more technical and distracts from the main ideas.

An important direction for future work is to extend our analysis to $\mathcal{ELI}$, that is, to add inverse roles. Even the case of $(\mathcal{ELI}, \text{AQ})$ appears to be challenging. In fact, it can be seen that a complexity classification of $(\mathcal{ELI}, \text{AQ})$ is equivalent to a complexity classification of all CSPs that have tree obstructions. In the following, we elaborate on this extension.

With inverse roles, there are OMQs $(\mathcal{T}, \Sigma, A(x))$ that are L-complete, obtained for example by setting $\mathcal{T} = \{\exists r.A \sqsubseteq A,\ \exists r^-.A \sqsubseteq A\}$ and $\Sigma = \{r, A\}$. Using a variation of the techniques from Section 3 and the technique of *transfer sequences* from [18], it should not be too hard to establish a dichotomy between AC^0 and L in $(\mathcal{ELI}, \text{CQ})$. We conjecture that AC^0, L, NL, and PTime are the only complexities that occur. We also conjecture that L-completeness coincides with rewritability into symmetric Datalog [54].

Lifting our dichotomy between NL and PTime to $(\mathcal{ELI}, \text{AQ})$ is non-trivial. In fact, we give below an example which shows that unbounded branching no longer coincides with unbounded pathwidth and thus our proof strategy, which uses unbounded branching in central places, has to be revised. Moreover, it seems difficult to approach the dichotomy between NL and PTime without first solving the L versus NL case. In this context, it is interesting to point out that for CSPs, the following conditional result is known [55]: if rewritability into linear Datalog coincides with NL, then rewritability into symmetric Datalog coincides with L.

Example 45. Consider the OMQ $Q = (\mathcal{T}, \Sigma, A(x)) \in (\mathcal{ELI}, \text{AQ})$ with $\Sigma = \{r, s, B, E, L\}$ and

$$\mathcal{T} = \{B \sqsubseteq M_1, \exists s.M_1 \sqsubseteq M_1, \exists r M_1 \sqsubseteq M_1', \exists s^- M_1' \sqsubseteq M_2,$$
$$\exists r^- M_2 \sqsubseteq M_2, M_2 \sqcap L \sqsubseteq M_1, M_2 \sqcap E \sqsubseteq A, \exists s.A \sqsubseteq A\}.$$

Q is unboundedly branching, as witnessed by the ABox in Figure 8 and generalizations thereof to arbitrary depth. A derivation of the query starts at B, the beginning marker, then it uses markers M_1 and M_2 to visit all the leafs from left to right in sequence, until it reaches E, the end marker, to derive the queried concept name A.

At the same time, Q is rewritable into linear Datalog and thus in NL, showing that unbounded branching and PTime-hardness no longer coincide.

Acknowledgements. This research was supported by ERC consolidator grant 647289 CODA. We express our gratitude to the reviewers of a previous version of this article for their careful comments.

References

References

[1] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, M. Rodriguez-Muro, R. Rosati, Ontologies and databases: The DL-Lite approach, in: Proc. of Reasoning Web, 2009, pp. 255–356.

[2] R. Kontchakov, M. Rodriguez-Muro, M. Zakharyaschev, Ontology-based data access with databases: A short course, in: Proc. of Reasoning Web, 2013, pp. 194–229.

[3] M. Bienvenu, B. ten Cate, C. Lutz, F. Wolter, Ontology-based data access: A study through disjunctive datalog, CSP, and MMSNP, ACM Trans. Database Syst. 39 (4) (2014) 33:1–33:44.

[4] M. Bienvenu, M. Ortiz, Ontology-mediated query answering with data-tractable description logics, in: Proc. of Reasoning Web, Vol. 9203 of LNCS, Springer, 2015, pp. 218–307.

[5] F. Baader, I. Horrocks, C. Lutz, U. Sattler, An Introduction to Description Logics, Cambride University Press, 2017.

[6] U. Hustadt, B. Motik, U. Sattler, Data complexity of reasoning in very expressive description logics, in: Proc. of IJCAI, Professional Book Center, 2005, pp. 466–471.

[7] A. Krisnadhi, C. Lutz, Data complexity in the $\mathcal{EL}$ family of description logics, in: Proc. of LPAR, Vol. 4790 of LNAI, Springer, 2007, pp. 333–347.

[8] R. Rosati, The limits of querying ontologies, in: Proc. of ICDT, Vol. 4353 of LNCS, Springer, 2007, pp. 164–178.

[9] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, R. Rosati, Data complexity of query answering in description logics, Artif. Intell. 195 (2013) 335–360.

[10] T. Eiter, M. Ortiz, M. Simkus, T. Tran, G. Xiao, Query rewriting for Horn-$\mathcal{SHIQ}$ plus rules, in: Proc. of AAAI, AAAI Press, 2012.

[11] M. Bienvenu, C. Lutz, F. Wolter, First order-rewritability of atomic queries in horn description logics, in: Proc. of IJCAI, IJCAI/AAAI, 2013, pp. 754–760.

[12] M. Kaminski, Y. Nenov, B. C. Grau, Datalog rewritability of disjunctive datalog programs and its applications to ontology reasoning, in: Proc. of AAAI, AAAI Press, 2014, pp. 1077–1083.

[13] S. Ahmetaj, M. Ortiz, M. Simkus, Polynomial datalog rewritings for expressive description logics with closed predicates, in: Proc. of IJCAI, IJCAI/AAAI Press, 2016, pp. 878–885.

[14] C. Feier, A. Kuusisto, C. Lutz, Rewritability in monadic disjunctive datalog, MMSNP, and expressive description logics, Logical Methods in Computer Science To appear (2019).

[15] P. Hansen, C. Lutz, İnanç Seylan, F. Wolter, Efficient query rewriting in the description logic $\mathcal{EL}$ and beyond, in: Proc. of IJCAI, AAAI Press, 2015, pp. 3034–3040.

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

[17] D. Trivela, G. Stoilos, A. Chortaras, G. B. Stamou, Optimising resolution-based rewriting algorithms for OWL ontologies, J. Web Sem. 33 (2015) 30–49.

[18] M. Bienvenu, P. Hansen, C. Lutz, F. Wolter, First order-rewritability and containment of conjunctive queries in Horn description logics, in: Proc. of IJCAI, IJCAI/AAAI Press, 2016, pp. 965–971.

[19] C. Lutz, F. Wolter, Non-uniform data complexity of query answering in description logics, in: Proc. of KR, AAAI Press, 2012.

[20] B. Motik, B. C. Grau, I. Horrocks, Z. Wu, A. Fokoue, C. Lutz (Eds.), OWL 2 Web Ontology Language Profiles (Second Edition), W3C Recommendation, 2009, available at `https://www.w3.org/TR/owl2-profiles/`.

[21] V. Dalmau, Linear datalog and bounded path duality of relational structures, Logical Methods in Computer Science 1 (1) (2005).

[22] V. Dalmau, A. A. Krokhin, Majority constraints have bounded pathwidth duality, Eur. J. Comb. 29 (4) (2008) 821–837.

[23] C. Carvalho, V. Dalmau, A. A. Krokhin, CSP duality and trees of bounded pathwidth, Theor. Comput. Sci. 411 (34-36) (2010) 3188–3208.

[24] A. Schaerf, On the complexity of the instance checking problem in concept languages with existential quantification, J. Intell. Inf. Syst. 2 (3) (1993) 265–278.

[25] D. Calvanese, G. D. Giacomo, D. Lembo, M. Lenzerini, R. Rosati, Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family, J. Autom. Reason. 39 (3) (2007) 385–429.

[26] P. Hansen, C. Lutz, Computing fo-rewritings in *EL* in practice: From atomic to conjunctive queries, in: Proc. of ISWC, Vol. 10587 of LNCS, Springer, 2017, pp. 347–363.

[27] A. Mottet, T. Nagy, M. Pinsker, M. Wrona, Smooth approximations and relational width collapses, in: Proc. of ICALP, Vol. 198 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, pp. 138:1–138:20.

[28] S. S. Cosmadakis, H. Gaifman, P. C. Kanellakis, M. Y. Vardi, Decidable optimization problems for database logic programs (preliminary report), in: Proc. of STOC, ACM, 1988, pp. 477–490.

[29] R. van der Meyden, Predicate boundedness of linear monadic datalog is in PSPACE, Int. J. Found. Comput. Sci. 11 (4) (2000) 591–612.

[30] M. Benedikt, B. ten Cate, T. Colcombet, M. Vanden Boom, The complexity of boundedness for guarded logics, in: Proc. of LICS, IEEE Computer Society, 2015, pp. 293–304.

[31] T. Feder, M. Y. Vardi, The computational structure of monotone monadic SNP and constraint satisfaction: A study through datalog and group theory, SIAM J. Comput. 28 (1) (1998) 57–104.

[32] A. A. Bulatov, A dichotomy theorem for nonuniform CSPs, in: Proc. of FOCS, 2017, pp. 319–330.

[33] D. Zhuk, A proof of CSP dichotomy conjecture, in: Proc. of FOCS, 2017, pp. 331–342.

[34] O. Gerasimova, S. Kikot, A. Kurucz, V. V. Podolskii, M. Zakharyaschev, A data complexity and rewritability tetrachotomy of ontology-mediated queries with a covering axiom, in: Proc. of KR, 2020, pp. 403–413.

[35] S. Kikot, A. Kurucz, V. V. Podolskii, M. Zakharyaschev, Deciding boundedness of monadic sirups, in: Proc. of PODS, ACM, 2021, pp. 370–387.

[36] C. Lutz, I. Seylan, F. Wolter, Ontology-based data access with closed predicates is inherently intractable(sometimes), in: Proc. of IJCAI, IJCAI/AAAI, 2013, pp. 1024–1030.

[37] C. Lutz, I. Seylan, F. Wolter, Ontology-mediated queries with closed predicates, in: Proc. of IJCAI, AAAI Press, 2015, pp. 3120–3126.

[38] C. Lutz, I. Seylan, F. Wolter, The data complexity of ontology-mediated queries with closed predicates, Log. Methods Comput. Sci. 15 (3) (2019).

[39] C. Lutz, L. Sabellek, Ontology-mediated querying with the description logic el: Trichotomy and linear datalog rewritability, in: Proc. of IJCAI, ijcai.org, 2017, pp. 1181–1187.

[40] S. Abiteboul, R. Hull, V. Vianu, Foundations of Databases, Addison-Wesley, 1995.

[41] F. Baader, S. Brandt, C. Lutz, Pushing the $\mathcal{EL}$ envelope, in: Proc. of IJCAI, 2005, pp. 364–369.

[42] C. Lutz, F. Wolter, Deciding inseparability and conservative extensions in the description logic $\mathcal{EL}$, J. Symb. Comput. 45 (2) (2010) 194–228.

[43] A. Calì, G. Gottlob, M. Kifer, Taming the infinite chase: Query answering under expressive relational constraints, J. Artif. Intell. Res. 48 (2013) 115–174.

[44] M. Bienvenu, M. Ortiz, Ontology-mediated query answering with data-tractable description logics, in: Proc. of Reasoning Web, 2015, pp. 218–307.

[45] C. Lutz, The complexity of conjunctive query answering in expressive description logics, in: Proc. of IJCAR, Vol. 5195 of LNCS, Springer, 2008, pp. 179–193.

[46] M. L. Furst, J. B. Saxe, M. Sipser, Parity, circuits, and the polynomial-time hierarchy, in: Proc. of FOCS, 1981, pp. 260–270.

[47] A. A. Bulatov, A. A. Krokhin, B. Larose, Dualities for constraint satisfaction problems, in: Complexity of Constraints - An Overview of Current Research Themes [Result of a Dagstuhl Seminar]., Vol. 5250 of LNCS, Springer, 2008, pp. 93–124.

[48] N. Immerman, Descriptive complexity, Graduate texts in computer science, Springer, 1999.

[49] P. 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.

[50] V. Geffert, A. Okhotin, Transforming two-way alternating finite automata to one-way nondeterministic automata, in: Proc. of MFCS, 2014, pp. 291–302.

[51] D. E. Muller, P. E. Schupp, Alternating automata on infinite trees, Theor. Comput. Sci. 54 (1987) 267–276.

[52] M. Y. Vardi, Reasoning about the past with two-way automata, in: Proc. of ICALP, Vol. 1443 of LNCS, Springer, 1998, pp. 628–641.

[53] T. Eiter, G. Gottlob, M. Ortiz, M. Simkus, Query answering in the description logic horn-, in: Proc. of JELIA, Vol. 5293 of LNCS, Springer, 2008, pp. 166–179.

[54] L. Egri, B. Larose, P. Tesson, Symmetric datalog and constraint satisfaction problems in LogSpace, Electronic Colloquium on Computational Complexity (ECCC) 14 (024) (2007) 1.

[55] A. Kazda, n-permutability and linear datalog implies symmetric datalog, Logical Methods in Computer Science 14 (2) (2018) 1–24.

Appendix A. Proofs for Section 2

Lemma 4. *Let $Q = (\mathcal{T}, \Sigma, q) \in (\mathcal{EL}, \text{conCQ})$, $\mathcal{A}$ a Σ-ABox and $\mathbf{a} \in \mathsf{ind}(\mathcal{A})^{\mathsf{ar}(q)}$ such that $\mathcal{A} \models Q(\mathbf{a})$. Then there is a pseudo tree-shaped Σ-ABox $\mathcal{A}'$ of core size at most $|q|$ and with $\mathbf{a}$ in its core that satisfies the following conditions:*

1. *there is a homomorphism from $\mathcal{A}'$ to $\mathcal{A}$ that is the identity on $\mathbf{a}$;*
2. *$\mathcal{A}' \models Q(\mathbf{a})$, witnessed by a homomorphism from q to $\mathcal{U}_{\mathcal{A}',\mathcal{T}}$ whose range consists solely of core individuals and of anonymous elements in a tree rooted in a core individual.*

Proof. (sketch) Assume that $\mathcal{A} \models Q(\mathbf{a})$ and let h be a homomorphism from $q(\mathbf{x})$ to $\mathcal{U}_{\mathcal{A},\mathcal{T}}$. Let $I \subseteq \mathsf{ind}(\mathcal{A})$ be the set of all individuals b that are either in the range of h or such that an anonymous element in the chase-generated tree below b is in the range of h. We unravel $\mathcal{A}$ into a potentially infinite pseudo tree-shaped ABox $\mathcal{A}_0$ with core I, see also [18]. A *trace* is a sequence $t = a_0 r_0 a_1 r_1 \cdots r_{n-1} a_n$, $n \geq 0$, such that $a_0 \in I$ and $r_i(a_i, a_{i+1}) \in \mathcal{A}$ for $0 \leq i < n$. We use $\mathsf{tail}(t)$ to denote a_n. Then $\mathcal{A}_0$ consists of the following assertions:

- all assertions $r(a, b) \in \mathcal{A}$ with $a, b \in I$;
- $A(t)$ for all traces t such that $A(\mathsf{tail}(t)) \in \mathcal{A}$;
- $r(t, t')$ for all traces t, t' such that $t' = tra$ for some a.

It is not hard to prove that

(∗) for every trace t, the tree in $\mathcal{U}_{\mathcal{A}_0,\mathcal{T}}$ below t is isomorphic to the tree in $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ below $\mathsf{tail}(t)$.

In fact, this can be shown by observing that a chase sequence $\mathcal{A}_0, \mathcal{A}_1, \ldots$ for constructing $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ can be converted into a chase sequence $\mathcal{A}'_0, \mathcal{A}'_1, \ldots$ for constructing $\mathcal{U}_{\mathcal{A}_0,\mathcal{T}}$ by replacing in any rule application the individuals $a \in \mathsf{ind}(\mathcal{A})$ with all possible traces t with $\mathsf{tail}(t) = a$.

Since $\mathcal{A} \models Q(\mathbf{a})$, there is a homomorphism h from q to $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ with $h(\mathbf{x}) = \mathbf{a}$. Due to the choice of I in the construction of $\mathcal{A}_0$ and (∗), it is clear that h can be converted into a homomorphism g from q to $\mathcal{U}_{\mathcal{A}_0,\mathcal{T}}$ such that $h(x) = g(x)$ if $h(x) \in \mathsf{ind}(\mathcal{A})$. Thus, $\mathcal{A}_0 \models Q(a)$ and the homomorphism g is as required by Condition (2) of Lemma 4.

It remains to deal with the fact that $\mathcal{A}_0$ needs not be finite. By the compactness theorem of first order logic, there exists a finite subset $\mathcal{A}_1 \subseteq \mathcal{A}_0$ such that $\mathcal{A}_1 \models Q(\mathbf{a})$. Let $\mathcal{A}'$ be the restriction of $\mathcal{A}_1$ to those individuals that are reachable in $G_{\mathcal{A}_1}$ from an individual in I. It can be verified that $\mathcal{A}'$ is as required. □

Lemma 5. *Let $\mathcal{T}$ be an $\mathcal{EL}$-TBox, $\mathcal{A}$ an pseudo tree-shaped ABox, and $a \in \mathsf{ind}(\mathcal{A})$ a non-core individual. Then $\mathcal{A}, \mathcal{T} \models B(a)$ if and only if $\mathcal{A}^a, \mathcal{T} \models B(a)$ for every concept name $B \in \mathsf{N_C}$.*

Proof. Let $\mathcal{A}_0, \mathcal{A}_1, \ldots$ be the sequence of ABoxes from the construction of $\mathcal{U}_{\mathcal{A},\mathcal{T}}$, and likewise for $\mathcal{B}_0, \mathcal{B}_1, \ldots$ and $\mathcal{U}_{\mathcal{A}^a,\mathcal{T}}$. It can be shown by induction on i that $B(b) \in \mathcal{A}_i$ if and only if $B(b) \in \mathcal{B}_i$ for all elements $b \in \Delta^{\mathcal{U}_{\mathcal{A}^a,\mathcal{T}}}$ and all concept names $B \in \mathsf{N_C}$ (assuming a canonical naming scheme for introducing fresh individual names). For the induction step, notice that rules (ii) and (vi) will never be applied, since $\mathcal{T}$ is an $\mathcal{EL}$-TBox. Overall, this yields an isomorphism between $\mathcal{U}_{\mathcal{A}^a,\mathcal{T}}$ and $(\mathcal{U}_{\mathcal{A},\mathcal{T}})^a$ that is the identity on $\mathsf{ind}(\mathcal{A}^a)$. Here, $(\mathcal{U}_{\mathcal{A},\mathcal{T}})^a$ refers to $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ restricted to elements b such that there is a path from a to b. The statement then follows from Point 3 of Lemma 2. $\square$

Lemma 6. *Let $Q = (\mathcal{T}, \Sigma, A(x)) \in (\mathcal{EL}, \mathrm{AQ})$ be an OMQ and $\mathcal{A}$ a Σ-ABox such that $\mathcal{A} \models Q(a)$. Then there exists $\mathcal{A}' \subseteq \mathcal{A}$ of degree at most $|\mathcal{T}|$ such that $\mathcal{A}' \models Q(a)$.*

Proof. (sketch) Assume that $\mathcal{A} \models Q(a)$ and let $\mathcal{A}_0, \mathcal{A}_1, \ldots$ be the sequence of ABoxes produced by chasing $\mathcal{A}$ with $\mathcal{T}$ as described in the construction of the universal model $\mathcal{U}_{\mathcal{A},\mathcal{T}}$. Let $\mathcal{A}_\omega$ be the ABox obtained in the limit. Since $\mathcal{A} \models Q(a)$, by Lemma 2, $A(a) \in \mathcal{A}_\omega$. Let $\mathcal{A}'$ be obtained from $\mathcal{A}$ as follows: For each $a \in \mathcal{A}$ and $\exists r.A \sqsubseteq B \in \mathcal{T}$ such that there is some $r(a,b) \in \mathcal{A}$ with $A(b) \in \mathcal{A}_\omega$, choose some such $r(a,b)$ such that for i minimal with $A(b) \in \mathcal{A}_i$, there is no $j < i$ and $r(a,b') \in \mathcal{A}$ with $A(b') \in \mathcal{A}_j$. Now let $\mathcal{A}'$ be obtained from $\mathcal{A}$ by removing all role assertions that have not been chosen. It is clear that the degree of $\mathcal{A}'$ is at most $|\mathcal{T}|$. Moreover, it is easy to verify that the role applications that led to the sequence $\mathcal{A}_0, \mathcal{A}_1, \ldots$ can be reproduced in $\mathcal{A}'$. Thus, the result $\mathcal{A}'_\omega$ of chasing $\mathcal{A}'$ with $\mathcal{T}$ is such that $A(a) \in \mathcal{A}'_\omega$. $\square$

Lemma 7. *Let $\mathcal{A}_1, \mathcal{A}_2$ be Σ-ABoxes and $\mathcal{T}$ an $\mathcal{ELI}$-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 $a \in \mathsf{ind}(\mathcal{A}_i)$, $i \in \{1,2\}$. Assume that $\mathcal{A}_i, \mathcal{T} \not\models A(a)$ for some $i \in \{1,2\}$, concept name A, and $a \in \mathsf{ind}(\mathcal{A}_i)$. We have to show that $\mathcal{A}_1 \cup \mathcal{A}_2, \mathcal{T} \not\models A(a)$. For each $j \in \{1,2\}$, let $\mathcal{I}_j$ be the universal model of $\mathcal{T}$ 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 $\mathcal{I}_2$, that is, $\Delta^{\mathcal{I}} = \Delta^{\mathcal{I}_1} \cup \Delta^{\mathcal{I}_2}$, $A^{\mathcal{I}} = A^{\mathcal{I}_1} \cup A^{\mathcal{I}_2}$ for all concept names A, and $r^{\mathcal{I}} = r^{\mathcal{I}_1} \cup r^{\mathcal{I}_2}$ for all role names r. 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, $\mathsf{tp}_{\mathcal{A}_1,\mathcal{T}}(a) = \mathsf{tp}_{\mathcal{A}_2,\mathcal{T}}(a)$. We must have $A_1, A_2 \in \mathsf{tp}_{\mathcal{A}_1,\mathcal{T}}(a)$, so $a \in A_1^{\mathcal{I}_1} \cap A_2^{\mathcal{I}_1}$ and since $\mathcal{I}_1$ is a model of $\mathcal{T}$, we conclude $a \in A_3^{\mathcal{I}_1} \subseteq A_3^{\mathcal{I}}$. Thus 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 $\mathcal{I}$.

□

Corollary 8. *Let $\mathcal{T}$ be an $\mathcal{ELI}$-TBox, $\mathcal{A}$ a pseudo tree-shaped ABox, $b \in \mathsf{ind}(\mathcal{A})$ not in the core of $\mathcal{A}$, and $\mathcal{A}'$ a tree-shaped ABox with root b such that $\mathsf{ind}(\mathcal{A}) \cap \mathsf{ind}(\mathcal{A}') = \{b\}$ and $\mathsf{tp}_{\mathcal{A}',\mathcal{T}}(b) = \mathsf{tp}_{\mathcal{A},\mathcal{T}}(b)$. If $\mathcal{A}''$ is the ABox obtained from $\mathcal{A}$ by replacing the subtree rooted at b by $\mathcal{A}'$, then $\mathsf{tp}_{\mathcal{A},\mathcal{T}}(a) = \mathsf{tp}_{\mathcal{A}'',\mathcal{T}}(a)$ for all $a \in \mathsf{ind}(\mathcal{A}) \cap \mathsf{ind}(\mathcal{A}'')$.*

Proof. The procedure of replacing the subtree rooted at b by $\mathcal{A}'$ can be implemented by the following four steps, none of which changes the types derived at individuals from $\mathsf{ind}(\mathcal{A}) \cap \mathsf{ind}(\mathcal{A}'')$.

- Let $t = \mathsf{tp}_{\mathcal{A},\mathcal{T}}(b)$ and define $\mathcal{B}_1 := \mathcal{A} \cup t(b)$. By Lemma 7, $\mathsf{tp}_{\mathcal{A},\mathcal{T}}(a) = \mathsf{tp}_{\mathcal{B}_1,\mathcal{T}}(a)$ for all $a \in \mathsf{ind}(\mathcal{A})$.

- Let $\mathcal{B}_2 := \mathcal{A}_b \cup t(b)$, so $\mathcal{B}_1 = \mathcal{B}_2 \cup (\mathcal{A}^b \cup t(b))$. Lemma 7 yields $\mathsf{tp}_{\mathcal{B}_1,\mathcal{T}}(a) = \mathsf{tp}_{\mathcal{B}_2,\mathcal{T}}(a)$ for all $a \in \mathsf{ind}(\mathcal{B}_1)$.

- Let $\mathcal{B}_3 := \mathcal{B}_2 \cup \mathcal{A}'$. By Lemma 7, $\mathsf{tp}_{\mathcal{B}_2,\mathcal{T}}(a) = \mathsf{tp}_{\mathcal{B}_3,\mathcal{T}}(a)$ for all $a \in \mathsf{ind}(\mathcal{B}_2)$.

- Observe that $\mathcal{B}_3 = \mathcal{A}'' \cup t(b)$, so Lemma 7 yields $\mathsf{tp}_{\mathcal{B}_3,\mathcal{T}}(a) = \mathsf{tp}_{\mathcal{A}'',\mathcal{T}}(a)$ for all $a \in \mathsf{ind}(\mathcal{A}'')$.

Overall, we have $\mathsf{tp}_{\mathcal{A},\mathcal{T}}(a) = \mathsf{tp}_{\mathcal{B}_1,\mathcal{T}}(a) = \mathsf{tp}_{\mathcal{B}_2,\mathcal{T}}(a) = \mathsf{tp}_{\mathcal{B}_3,\mathcal{T}}(a) = \mathsf{tp}_{\mathcal{A}'',\mathcal{T}}(a)$ for all $a \in \mathsf{ind}(\mathcal{A}) \cap \mathsf{ind}(\mathcal{A}'')$. □

Appendix B. Proofs for Section 3

Lemma 16. *Let $Q = (\mathcal{T}, \Sigma, q) \in (\mathcal{EL}, \text{conCQ})$ have the ability to simulate* REACH *and let $\mathcal{A}$, $\mathbf{a}$, b, c, t_0, t_1, $\mathcal{A}_{\mathsf{source}}$, $\mathcal{A}_{\mathsf{edge}}$ and $\mathcal{A}_{\mathsf{target}}$ be as in Definition 11. Then*

1. $\mathsf{tp}_{\mathcal{A}_{\mathsf{source}},\mathcal{T}}(c) = t_1$;
2. $\mathsf{tp}_{\mathcal{A}_{\mathsf{edge}} \cup t_1(c),\mathcal{T}}(b) = t_1$;
3. $\mathcal{A}_{\mathsf{target}} \cup t_1(b), \mathcal{T} \models Q(\mathbf{a})$.
4. $\mathsf{tp}_{\mathcal{A}_{\mathsf{source}} \cup t_1(c),\mathcal{T}}(c) = t_1$;
5. $\mathsf{tp}_{\mathcal{A}_{\mathsf{edge}} \cup t_i(b) \cup t_j(c),\mathcal{T}}(c) = t_j$ *for all* $i, j \in \{0, 1\}$;
6. $\mathsf{tp}_{\mathcal{A}_{\mathsf{edge}} \cup t_i(b) \cup t_j(c),\mathcal{T}}(b) = t_{\max\{i,j\}}$ *for all* $i, j \in \{0, 1\}$;
7. $\mathsf{tp}_{\mathcal{A}_{\mathsf{target}} \cup t_0(b),\mathcal{T}}(b) = t_0$;
8. $\mathsf{tp}_{\mathcal{A}_{\mathsf{source}} \cup t_1(c),\mathcal{T}}(a) = \mathsf{tp}_{\mathcal{A},\mathcal{T}}(a)$ *for all* $a \in \mathsf{ind}(\mathcal{A}_{\mathsf{source}})$;
9. $\mathsf{tp}_{\mathcal{A}_{\mathsf{edge}} \cup t_1(b) \cup t_1(c),\mathcal{T}}(a) = \mathsf{tp}_{\mathcal{A},\mathcal{T}}(a)$ *for all* $a \in \mathsf{ind}(\mathcal{A}_{\mathsf{edge}})$;
10. $\mathsf{tp}_{\mathcal{A}_{\mathsf{target}} \cup t_1(b),\mathcal{T}}(a) = \mathsf{tp}_{\mathcal{A},\mathcal{T}}(a)$ *for all* $a \in \mathsf{ind}(\mathcal{A}_{\mathsf{target}})$.

Proof. Point 1: $\mathsf{tp}_{\mathcal{A},\mathcal{T}}(c) = t_1$, but by Lemma 5, $\mathcal{A}$ can be replaced by $\mathcal{A}_{\mathsf{source}}$.

Point 2: $\mathsf{tp}_{\mathcal{A},\mathcal{T}}(b) = t_1$, but by Lemma 5, $\mathcal{A}$ can be replaced by $\mathcal{A}^b$. Furthermore, by Corollary 8, $\mathcal{A}^b$ can be replaced by $\mathcal{A}_{\mathsf{edge}} \cup t_1(c)$.

Point 3: $\mathcal{A} = \mathcal{A}_{\mathsf{target}} \cup \mathcal{A}^b$. Since $\mathsf{tp}_{\mathcal{A}^b,\mathcal{T}}(b) = t_1$, Corollary 8 yields that $\mathsf{tp}_{\mathcal{A}_{\mathsf{target}} \cup \mathcal{A}^b,\mathcal{T}}(a) = \mathsf{tp}_{\mathcal{A}_{\mathsf{target}} \cup t_1(b),\mathcal{T}}(a)$ for all $a \in \mathcal{A}_{\mathsf{target}}$. As a consequence and due to Lemma 3, $\mathcal{U}_{\mathcal{A}_{\mathsf{target}} \cup t_1(b),\mathcal{T}}$ and the restriction of $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ to the individuals in $\mathsf{ind}(\mathcal{A}_{\mathsf{target}})$ and the anonymous elements below them are homomorphically equivalent via homomorphisms that are the identity on $\mathsf{ind}(\mathcal{A}_{\mathsf{target}})$. Since $\mathcal{A} \models Q(\mathbf{a})$, q is connected, and b has distance at least $|q|$ from the core, there is a homomorphism h from q to the restriction of $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ to the individuals in $\mathsf{ind}(\mathcal{A}_{\mathsf{target}})$ and the anonymous elements below them such that $h(\mathbf{x}) = \mathbf{a}$. It follows that $\mathcal{A}_{\mathsf{target}} \cup t_1(b), \mathcal{T} \models Q(\mathbf{a})$.

Point 4 follows immediately from Point 1 and Lemma 7.

Point 5 is similar to Point 2, since c is a leaf in $\mathcal{A}_{\mathsf{edge}}$.

Point 6: For the cases $(i, j) \in \{(1, 0), (1, 1), (0, 1)\}$, the statement follows from Point 2 and monotonicity. If $i = j = 0$, then it follows from Point 3 of Definition 11 and Lemma 5.

Point 7: Since b is a leaf in $\mathcal{A}_{\mathsf{target}}$ and by Lemma 5, $\mathsf{tp}_{\mathcal{A}_{\mathsf{target}} \cup t_0(b),\mathcal{T}}(b) = \mathsf{tp}_{(\mathcal{A}_{\mathsf{target}})^b \cup t_0(b),\mathcal{T}}(b) = t_0$.

Point 8: If $a = c$, this follows from Point 1. If $a \neq c$, then a is a descendant of c, since c is the root of $\mathcal{A}_{\mathsf{source}}$. Applying Lemma 5 twice yields $\mathsf{tp}_{\mathcal{A}_{\mathsf{source}} \cup t_1(c),\mathcal{T}}(a) = \mathsf{tp}_{(\mathcal{A}_{\mathsf{source}})^a,\mathcal{T}}(a) = \mathsf{tp}_{\mathcal{A},\mathcal{T}}(a)$.

Point 9: For $a \in \{b, c\}$, this follows from Points 5 and 6. If $a \notin \{b, c\}$, then a is a descendant of b, since b is the root of $\mathcal{A}_{\mathsf{edge}}$. By Lemma 5, $\mathsf{tp}_{\mathcal{A}_{\mathsf{edge}} \cup t_1(b) \cup t_1(c),\mathcal{T}}(a) = \mathsf{tp}_{(\mathcal{A}_{\mathsf{edge}})^a \cup t_1(c),\mathcal{T}}(a)$. By Corollary 8 and Point 1, the latter is equal to $\mathsf{tp}_{(\mathcal{A}_{\mathsf{edge}})^a \cup \mathcal{A}_{\mathsf{source}},\mathcal{T}}(a)$. Again by Lemma 5, this is equal to $\mathsf{tp}_{\mathcal{A},\mathcal{T}}(a)$.

Point 10: By Point 2 of Definition 11 and Lemma 5, $\mathsf{tp}_{\mathcal{A}^b,\mathcal{T}}(b) = t_1$. Thus, we can replace $\mathcal{A}_{\mathsf{target}} \cup t_1(b)$ by $\mathcal{A}_{\mathsf{target}} \cup \mathcal{A}^b$ using Corollary 8. Since $\mathcal{A} = \mathcal{A}_{\mathsf{target}} \cup \mathcal{A}^b$, the statement follows. □

Appendix C. Proofs for Section 4.1

Proposition 18. *Let $Q \in (\mathcal{EL}, \mathrm{CQ})$. Then Q has unbounded pathwidth iff Q is unboundedly branching.*

Proof. The "$\Leftarrow$" direction is clear since the full binary tree of depth k has pathwidth $\lceil \frac{k}{2} \rceil$ and pathwidth of a graph cannot be smaller than that of its minors. For the "$\Rightarrow$" direction, we start by showing that for tree-shaped ABoxes, the branching number gives an upper bound on the pathwidth.

Claim. Let $\mathcal{A}$ be a tree-shaped ABox. Then there exists a (j, k)-path decomposition $V_1, \ldots, V_n$ of $\mathcal{A}$ with $k \leq \mathsf{br}(\mathcal{A}) + 2$ and $j \leq k - 1$ such that the root of $\mathcal{A}$ is an element of V_n.

We prove the claim by induction on the depth of $\mathcal{A}$. If $\mathcal{A}$ has depth 0, then $\mathcal{A}$ has only one individual, $\mathsf{br}(\mathcal{A}) = 0$, and there is a trivial $(0, 1)$-path decomposition. If $\mathcal{A}$ has depth 1, then the root a of $\mathcal{A}$ has children $a_1, \ldots, a_n$ with $n \geq 1$. We have $\mathsf{br}(\mathcal{A}) \leq 1$ and there is a $(1, 2)$-path decomposition $V_1, \ldots, V_n$, where $V_i = \{a, a_i\}$.

If $\mathcal{A}$ has depth at least 2, let the root of $\mathcal{A}$ be called a and its children $a_1, \ldots, a_m$. Let $V_1^i, \ldots, V_{n_i}^i$ be the path decomposition of $\mathcal{A}^{a_i}$ that exists by induction hypothesis, for $1 \leq i \leq m$. We distinguish two cases:

- If $\mathsf{br}(\mathcal{A}) = \mathsf{max}\{\mathsf{br}(\mathcal{A}^{a_i}) \mid 1 \leq i \leq m\}$, then by definition of br, there is precisely one child a_i of a with $\mathsf{br}(\mathcal{A}^{a_i}) = \mathsf{br}(\mathcal{A})$. W.l.o.g. assume that $a_i = a_1$. Then $V_1^1, \ldots, V_{n_1}^1, \{a, a_1\}, \{a\} \cup V_1^2, \ldots, \{a\} \cup V_{n_2}^2, \{a\} \cup V_1^3, \ldots, \{a\} \cup V_{n_3}^3, \ldots, \{a\} \cup V_1^m, \ldots, \{a\} \cup V_{n_m}^m$ is a path decomposition of $\mathcal{A}$ that fulfils the condition from the claim.
- If $\mathsf{br}(\mathcal{A}) = 1 + \mathsf{max}\{\mathsf{br}(\mathcal{A}^{a_i}) \mid 1 \leq i \leq m\}$, then $\{a\} \cup V_1^1, \ldots, \{a\} \cup V_{n_1}^1, \{a\} \cup V_1^2, \ldots, \{a\} \cup V_{n_2}^2, \ldots, \{a\} \cup V_1^m, \ldots, \{a\} \cup V_{n_m}^m$ is a path decomposition of $\mathcal{A}$ that fulfils the condition from the claim.

This finishes the proof of the claim.

We next show that for every OMQ $Q = (\mathcal{T}, \Sigma, q) \in (\mathcal{EL}, \mathrm{CQ})$, $\mathsf{br}(Q) = k$ implies $\mathsf{pw}(Q) \leq k + 2 + |q|$. Let Q be such an OMQ. Take a Σ-ABox $\mathcal{A}$ and $\mathbf{a} \in \mathsf{ind}(\mathcal{A})$ with $\mathcal{A} \models Q(\mathbf{a})$. We have to show that there is a Σ-ABox $\mathcal{A}'$ of pathwidth at most $k+2+|q|$ such that $\mathcal{A}' \models Q(\mathbf{a})$ and there is a homomorphism from $\mathcal{A}'$ to $\mathcal{A}$ that is the identity on $\mathbf{a}$. By Lemma 4, we obtain from $\mathcal{A}$ a pseudo tree-shaped Σ-ABox $\mathcal{A}'$ such that there is a homomorphism from $\mathcal{A}'$ to $\mathcal{A}$ that is the identity on $\mathbf{a}$. Clearly, $\mathcal{M}_Q$ contains a subset $\mathcal{A}''$ of $\mathcal{A}'$. We show that $\mathcal{A}''$ is as required, that is, the pathwidth of $\mathcal{A}''$ is at most k. From $\mathsf{br}(Q) = k$, $\mathcal{A}'' \models Q(\mathbf{a})$, and $\mathcal{A}'' \in \mathcal{M}_Q$, it follows that $\mathsf{br}(\mathcal{A}'') \leq k$. Let $\mathcal{A}'$ have core $\mathcal{C}$ and trees $\mathcal{A}_1, \ldots, \mathcal{A}_m$. By the claim, every $\mathcal{A}_i$ has a $(j, k+2)$-path decomposition $V_1^i, \ldots, V_{n_i}^i$. Then we find a $(j + |q|, k + 2 + |q|)$-path decomposition of $\mathcal{A}'$: $\mathsf{ind}(\mathcal{C}) \cup V_1^1, \ldots, \mathsf{ind}(\mathcal{C}) \cup V_{n_1}^1, \ldots, \mathsf{ind}(\mathcal{C}) \cup V_1^m, \ldots, \mathsf{ind}(\mathcal{C}) \cup V_{n_m}^m$. □

Lemma 19. *Let $q \in conCQ$ be Boolean and treeifiable, $\mathcal{I}_1, \mathcal{I}_2$ tree-shaped interpretations, and $d_i \in \Delta^{\mathcal{I}_i}$ for $i \in \{1, 2\}$ such that $d_1 \in C^{\mathcal{I}_1}$ implies $d_2 \in C^{\mathcal{I}_2}$ for* *all $C \in \mathcal{C}_q$. If there is a homomorphism from q to $\mathcal{I}_1$ with d_1 in its range, then there is a homomorphism from q to $\mathcal{I}_2$ with d_2 in its range.*

Proof. Assume that there is a homomorphism h_1 from q to $\mathcal{I}_1$ with d_1 in its range. Since $\mathcal{I}_1$ is tree-shaped, this homomorphism factors into $h_1 = g_1 \circ h_q$, where h_q is the obvious homomorphism from q to q^{tree} and g_1 is a homomorphism from q^{tree} to $\mathcal{I}_1$. It clearly suffices to show that there is a homomorphism g_2 from q^{tree} to $\mathcal{I}_2$ with d_2 in its range.

Let $X_0 = g_1^{-1}(d_1)$. In a first step, we set $g_2(x) = d_2$ for all $x \in X_0$ and extend g_2 upwards as follows. Whenever $g_2(y)$ is already defined and there is an atom $r(x, y)$ in q^{tree}, define $g_2(x)$ to be the (unique) predecessor of $g_2(y)$ in $\mathcal{I}_2$. We show that g_2 is a homomorphism from $q^{\mathsf{tree}}|_{\mathsf{dom}(g_2)}$ to $\mathcal{I}_2$, $\mathsf{dom}(g_2)$ the domain of g_2. If $r(x, y) \in q^{\mathsf{tree}}$ with $g_2(x), g_2(y)$ defined, then q^{tree} contains a role path $r_1 \cdots r_n$ from x_1 to $x_n \in X_0$ with $r_1 = r$. Thus, there is a concept $C = \exists r_n^- \ldots \exists r_1^-.\top \in \mathcal{C}_q$ such that $d_1 \in C^{\mathcal{I}_1}$. It follows that $d_2 \in C^{\mathcal{I}_2}$ and since $\mathcal{I}_2$ is tree-shaped and by construction of g_2, this yields $(g_2(x), g_2(y)) \in r^{\mathcal{I}_2}$. The argument for atoms $A(x) \in q^{\mathsf{tree}}$ where g_2 has been defined on x is similar, using concepts of the form $C = \exists r_n^- \ldots \exists r_1^-.A \in \mathcal{C}_q$.

In a second step, we define g_2 on all the remaining variables. Whenever $g_2(z)$ is still undefined for some $z \in \mathsf{var}(q^{\mathsf{tree}})$, there must be some $r(x, y) \in q^{\mathsf{tree}}$ such that $g_2(x)$ is already defined, $g_2(y)$ is not yet defined, and z is in $q^{\mathsf{tree}}|_{\mathsf{reach}(x,y)}$. Since q is connected, there must be a (potentially empty) role path $r_1 \cdots r_n$ in q^{tree} from x to a variable $x_0 \in X_0$. Thus, $\mathcal{C}_q$ contains $C = \exists r_n^-. \cdots \exists r_1^-.D \in \mathcal{C}_q$ where D is the $\mathcal{EL}$ concept that corresponds to $q^{\mathsf{tree}}|_{\mathsf{reach}(x,y)}$ and since $g_1(x_0) = d_1$, we have $d_1 \in C^{\mathcal{I}_1}$. Consequently, $d_2 \in C^{\mathcal{I}_2}$. Since $\mathcal{I}_2$ is tree-shaped, this implies $g_2(x) \in D^{\mathcal{I}_2}$ and thus there is a homomorphism from $q|_{\mathsf{reach}(x,y)}$ to $\mathcal{I}_2$ that maps x to $g_2(x)$. We use this homomorphism to extend g_2 to all variables in $\mathsf{reach}(x, y)$. $\square$

Lemma 22. *Let T be a full binary tree of depth $n \cdot k \cdot d$ whose nodes are colored with n colors, $k \geq 0$ and $n, d \geq 1$. Then T has as a minor a monochromatic full binary tree of depth k such that any two distinct nodes of the minor have distance at least d from each other in T.*

Proof. Let T be a full binary tree of depth k whose nodes are colored with n colors. We associate T with a tuple $(m_1, \ldots, m_n)$ by letting, for $1 \leq i \leq m$, m_i be the minimum integer such that T does not have the color i monochromatic full binary tree of depth m_i as a minor. We prove the following.

Claim. $\sum_{i=1}^{n} m_i \geq k + 1$.

We prove the claim by induction on k. For $k = 0$, there is only one node, say of color i. Then clearly $\sum_{i=1}^{n} m_i = 1 \geq 1 = k + 1$.

Now assume that the claim holds for k and consider a tree T of depth $k+1$, with associated tuple $(m_1, \ldots, m_n)$. Let a be the root of T and let the children of a root the subtrees T_1 and T_2, $(m_1^j, \ldots, m_n^j)$ the tuple associated with T_j for $j \in \{1, 2\}$. We distinguish two cases.

First assume that 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 $m_j = \max\{m_j^1, m_j^2\} > m_j^1$ and $m_i \geq m_i^1$ for all $i \neq j$. By the claim, $\sum_{i=1}^n m_i^1 \geq k+1$. It follows that $\sum_{i=1}^n m_i \geq k+2$, as required.

Now assume that there is no such color j. Let i_0 be the color of a. From $m_{i_0}^1 = m_{i_0}^2$, it follows that $m_{i_0} > m_{i_0}^1$ and thus we can proceed as before with i_0 in place of j. This finishes the proof of the claim.

The statement of the lemma now follows easily for $d = 1$: Let T be a full binary tree of depth $n \cdot k$ whose nodes are colored with n different colors. If there is no full monochromatic binary tree of depth k as a minor in T, then $m_i \leq k$ for all colors i, in contradiction to $\sum_{i=1}^n m_i \geq n \cdot k + 1$.

Now consider the case where $d > 1$. From the case $d = 1$, T contains as a minor a full monochromatic binary tree T' of depth $d \cdot k$. To obtain the desired full monochromatic binary tree T'' of depth k whose nodes have distance at least d from each other, we choose appropriate nodes from T'. Recall that the nodes of T' are $V = \{1,2\}^k$. Then T'' can be constructed by choosing the nodes $V \cap \{1^d, 2^d\}^*$. Clearly, T'' is as required. □

Proposition 23. *Let $Q \in (\mathcal{EL}, \text{conCQ})$. Then Q has the ability to simulate* PSA *iff Q is unboundedly branching.*

Proof. We prove the missing "⇒" direction. Assume that $Q = (\mathcal{T}, \Sigma, q) \in (\mathcal{EL}, \text{conCQ})$ has the ability to simulate PSA. Then there are $\mathcal{A}, \mathbf{a}, b, c, d, t_0$, and t_1 as in Definition 20. Let $k \geq 1$. We have to show that there is a pseudo tree-shaped Σ-ABox $\mathcal{A} \in \mathcal{M}_Q$ that has a tree $\mathcal{A}_i$ that has the full binary tree of depth k as a minor. We start with constructing an ABox $\mathcal{A}_0$ built up from the following set of ABoxes:

- one copy of $\mathcal{A}_{\mathsf{target}}$, where b is renamed to b_ε and all other individuals retain their original name;
- for every $w \in \bigcup_{i=0}^{k-1} \{0,1\}^i$, one copy $\mathcal{A}_{\wedge,w}$ of $\mathcal{A}_\wedge$, where b is renamed to b_w, c is renamed to b_{w0} and d is renamed to b_{w1}, and all other individuals are fresh;
- for every $w \in \{0,1\}^k$, one copy $\mathcal{A}_{\mathsf{source},w}$ of $\mathcal{A}_{\mathsf{source}}$, where b is renamed to b_w and all other individuals are fresh.

Since all $\mathcal{A}_\wedge$ and $\mathcal{A}_{\mathsf{source}}$ are tree-shaped, the resulting ABox is $\mathcal{A}_0$ pseudo tree-shaped with the same core as $\mathcal{A}_{\mathsf{target}}$.

It is clear that $\mathcal{A}_0$ has the full binary tree of depth k as a minor, formed by the individuals b_w, $w \in \{0,1\}^{\leq k}$.

Claim: $\mathcal{A}_0 \models Q(\mathbf{a})$.

First, we show by induction on $k - |w|$ that $\mathsf{tp}_{(\mathcal{A}_0)^{b_w},\mathcal{T}}(b_w) = t_1$. If $k - |w| = 0$, then $|w| = k$, so $(\mathcal{A}^0)_w^b$ is a copy of $\mathcal{A}_{\mathsf{source}}$. Thus, the statement to show is $\mathsf{tp}_{\mathcal{A}_{\mathsf{source}},\mathcal{T}}(c) = t_1$, which is Point 1 of Lemma 46. Now let $k - |w| = \ell > 0$ and assume the statement has been shown for $k - |w| = \ell - 1$. The ABox $(\mathcal{A}_0)^{b_w}$ can be regarded as the union of three ABoxes, which are first, a copy of $\mathcal{A}_\wedge$ where b is renamed to b_w, c is renamed to b_{w0} and d is renamed to b_{w1}, secondly, $(\mathcal{A}_0)^{b_{w0}}$, and third, $(\mathcal{A}_0)^{b_{w1}}$. By induction hypothesis and Corollary 8, in $(\mathcal{A}_0)^{b_w}$, we can replace $(\mathcal{A}_0)^{b_{w0}}$ by $t_1(b_{w0})$ and $(\mathcal{A}_0)^{b_{w1}}$ by $t_1(b_{w1})$. Thus, $\mathsf{tp}_{(\mathcal{A}_0)^{b_w},\mathcal{T}}(b_w) = \mathsf{tp}_{\mathcal{A}_\wedge \cup t_1(c) \cup t_1(d),\mathcal{T}}(b)$. By Point 2 of Lemma 46, the latter is equal to t_1. This finishes the induction and thus, $\mathsf{tp}_{(\mathcal{A}_0)^{b_\varepsilon},\mathcal{T}}(b_w) = t_1$. By Corollary 8, $\mathsf{tp}_{\mathcal{A}_0,\mathcal{T}}(a) = \mathsf{tp}_{(\mathcal{A}_0)^{b_\varepsilon} \cup t_1(b_\varepsilon),\mathcal{T}}(a)$ for every individual a in the copy of $\mathcal{A}_{\mathsf{target}}$. Now it follows from Point 3 of Lemma 46 that $\mathcal{A}_0 \models Q(\mathbf{a})$, which finishes the proof of the claim.

The claim yields $\mathcal{A}_0 \models Q(\mathbf{a})$, but there is no guarantee that $\mathcal{A}_0$ is minimal with this property, thus $\mathcal{A}_0$ need not be from $\mathcal{M}_Q$. Let $\mathcal{A}_0, \dots \mathcal{A}_\ell$ be the sequence of ABoxes obtained by starting with $\mathcal{A}_0$ and exhaustively removing assertions such that $\mathcal{A}_i \models Q(\mathbf{a})$ still holds. We argue that the resulting ABox still has the full binary tree of depth k as a minor.

It suffices to show that role assertions connecting two individuals that lie on the same path from the core to a b_w, $w \in \{0,1\}^k$ are never removed. Assume to the contrary that such a role assertion is removed when transitioning from $\mathcal{A}_i$ to $\mathcal{A}_{i+1}$. We distinguish three cases:

- The removed role assertion lies in $\mathcal{A}_{\wedge,w}$, for some $w \in \{0,1\}^i$, $0 \leq i < k$, on the path from b_w to b_{w0}. This disconnects b_w from b_{w0} (and from the whole subtree rooted at b_{w0}). Since the type derived at an individual only depends on its connected component, and by monotonicity, $\mathsf{tp}_{\mathcal{A}_{i+1},\mathcal{T}}(b_w) \subseteq \mathsf{tp}_{(\mathcal{A}_0)_{b_{w0}},\mathcal{T}}(b_w)$. By Lemma 5, $\mathsf{tp}_{(\mathcal{A}_0)_{b_{w0}},\mathcal{T}}(b_w) = \mathsf{tp}_{(\mathcal{A}_0)^{b_w}_{b_{w0}},\mathcal{T}}(b_w)$. By the claim above, monotonicity, and Corollary 8,
$$\mathsf{tp}_{(\mathcal{A}_0)^{b_w}_{b_{w0}},\mathcal{T}}(b_w) \subseteq \mathsf{tp}_{(\mathcal{A}_0)^{b_w}_{b_{w0}b_{w1}} \cup t_1(b_{w1}),\mathcal{T}}(b_w).$$
By construction of $\mathcal{A}_0$, the latter is equal to $\mathsf{tp}_{\mathcal{A}_\wedge \cup t_1(d),\mathcal{T}}(b)$. By Point 5 of Lemma 46 and monotonicity, $\mathsf{tp}_{\mathcal{A}_\wedge \cup t_1(d),\mathcal{T}}(b) \subseteq t_0$. Overall, this yields $\mathsf{tp}_{\mathcal{A}_{i+1},\mathcal{T}}(b_w) \subseteq t_0$. A similar induction as in the claim yields $\mathsf{tp}_{\mathcal{A}_{i+1},\mathcal{T}}(b_\varepsilon) \subseteq t_0$. Now it follows from Conditions 3 and 5 from Definition 20 and from Corollary 8 that $\mathcal{A}_{i+1} \not\models Q(\mathbf{a})$, a contradiction.
- The removed role assertion lies in $\mathcal{A}_{\wedge,w}$, for some $w \in \{0,1\}^k$, on the path from b_w to b_{w1}. This case is analogous.
- The removed role assertion lies in $\mathcal{A}_{\mathsf{target}}$ on the path from the core to b_ε. This disconnects a superset of $(\mathcal{A}_0)^{b_0}$ from the core, so by the first case and monotonicity, $\mathcal{A}_{i+1} \not\models Q(\mathbf{a})$, a contradiction.

□

The following lemma lists some properties that follow from the ability to simulate PSA. It will be important for the proof of Lemma 25 below.

Lemma 46. *Let $Q = (\mathcal{T}, \Sigma, q) \in (\mathcal{EL}, conCQ)$ have the ability to simulate* PSA *and let $\mathcal{A}, \mathbf{a}, b, c, d, t_0, t_1, \mathcal{A}_{\mathsf{source}}, \mathcal{A}_\wedge$ and $\mathcal{A}_{\mathsf{target}}$ be as in Definition 20. Then*

1. $\mathsf{tp}_{\mathcal{A}_{\mathsf{source}},\mathcal{T}}(c) = t_1$;

2. $\mathsf{tp}_{\mathcal{A}_\wedge \cup t_1(c) \cup t_1(d),\mathcal{T}}(b) = t_1$;

3. $\mathcal{A}_{\mathsf{target}} \cup t_1(b), \mathcal{T} \models Q(\mathbf{a})$;

4. $\mathsf{tp}_{\mathcal{A}_{\mathsf{source}} \cup t_1(c),\mathcal{T}}(c) = t_1$;

5. *for* $i, j, k \in \{0, 1\}$,
$$\mathsf{tp}_{\mathcal{A}_\wedge \cup t_i(c) \cup t_j(d) \cup t_k(b),\mathcal{T}}(b) = \begin{cases} t_1 & \textit{if } k = 1 \textit{ or } i = j = 1 \\ t_0 & \textit{otherwise} \end{cases}$$

6. *for* $i, j, k \in \{0, 1\}$, $\mathsf{tp}_{\mathcal{A}_\wedge \cup t_i(c) \cup t_j(d) \cup t_k(b),\mathcal{T}}(c) = t_i$;

7. *for* $i, j, k \in \{0, 1\}$, $\mathsf{tp}_{\mathcal{A}_\wedge \cup t_i(c) \cup t_j(d) \cup t_k(b),\mathcal{T}}(d) = t_j$;

8. $\mathsf{tp}_{\mathcal{A}_{\mathsf{target}} \cup t_0(b),\mathcal{T}}(b) = t_0$;

9. $\mathsf{tp}_{\mathcal{A}_{\mathsf{source}} \cup t_1(c),\mathcal{T}}(a) = \mathsf{tp}_{\mathcal{A},\mathcal{T}}(a)$ *for all* $a \in \mathsf{ind}(\mathcal{A}_{\mathsf{source}})$;

10. $\mathsf{tp}_{\mathcal{A}_\wedge \cup t_1(b) \cup t_1(c) \cup t_1(d),\mathcal{T}}(a) = \mathsf{tp}_{\mathcal{A},\mathcal{T}}(a)$ *for all* $a \in \mathsf{ind}(\mathcal{A}_\wedge)$.

Proof. The proof is very similar to the proof of Lemma 16. We only prove some points.

Point 2. By Point 2 of Definition 20, $\mathsf{tp}_{\mathcal{A},\mathcal{T}}(b) = t_1$. We manipulate the ABox without changing the type derived at b. By Lemma 5, $\mathcal{A}$ can be replaced by $\mathcal{A}^b$. By Point 2 of Definition 20 and Corollary 8, we can replace $\mathcal{A}^b$ by $\mathcal{A}_\wedge \cup t_1(c) \cup t_1(d)$, and the statement follows.

Point 5. If $k = 1$ or $i = j = 1$, the statement follows from Point 2 and monotonicity. Otherwise, we have $k = 0$ and not $i = j = 1$. We only consider the case $i = 0$ and $j = 1$, the other cases are similar. We have to show that $\mathsf{tp}_{\mathcal{A}_\wedge \cup t_0(c) \cup t_1(d) \cup t_0(b),\mathcal{T}}(b) = t_0$. By Point 4 of Definition 20, $\mathsf{tp}_{\mathcal{A}_c \cup t_0(c),\mathcal{T}}(b) = t_0$. We manipulate the ABox without changing the type derived at b. By Lemma 5, we can replace $\mathcal{A}_c \cup t_0(c)$ by $\mathcal{A}^b_c \cup t_0(c)$. By Point 2 of Definition 20 and Lemma 5, $\mathsf{tp}_{\mathcal{A}^d,\mathcal{T}}(d) = t_1$. Thus, by Corollary 8, we can replace $\mathcal{A}^b_c \cup t_0(c)$ by $\mathcal{A}^b_{cd} \cup t_0(c) \cup t_1(d)$, the statement follows.

Point 10. For $a \in \{b, c, d\}$, the statement follows from Points 5 to 7. Otherwise, consider $\mathsf{tp}_{\mathcal{A}_\wedge \cup t_1(b) \cup t_1(c) \cup t_1(d),\mathcal{T}}(a)$. We manipulate the ABox so that the type derived at a does not change. By Lemma 7, we can replace the ABox by $\mathcal{A}_\wedge \cup t_1(c) \cup t_1(d)$. By Point 1 and by Corollary 8, we can replace $\mathcal{A}_\wedge \cup t_1(c) \cup t_1(d)$ by $\mathcal{A}^\wedge \cup \mathcal{A}^c \cup t_1(d)$. From Point 2 of Definition 20 and Lemma 5, $\mathsf{tp}_{\mathcal{A}^d,\mathcal{T}}(d) = t_1$. Thus, by Corollary 8, we can replace $\mathcal{A}^\wedge \cup \mathcal{A}^c \cup t_1(d)$ by $\mathcal{A}^\wedge \cup \mathcal{A}^c \cup \mathcal{A}^d$. Again by Lemma 5, we can replace the ABox by $\mathcal{A}$ and the statement follows. □

Lemma 25. *t is accessible in G iff $\mathcal{A}_G \models Q(\mathbf{a})$.*

Proof. For the "⇒" direction, assume that t is accessible in G. Define a sequence $S = S_0 \subseteq S_1 \subseteq \cdots \subseteq V$ by setting

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

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 $t_1 \subseteq \mathsf{tp}_{\mathcal{A}_G,\mathcal{T}}(a_v)$. In fact, the induction start follows from Point 1 of Lemma 46 and monotonicity, and the induction step follows from Point 2 and monotonicity. Since t is accessible, this yields $t_1 \subseteq \mathsf{tp}_{\mathcal{A}_G,\mathcal{T}}(a_t)$. It follows then from Point 3 of Lemma 46 and monotonicity that $\mathcal{A}_G \models Q(\mathbf{a})$, as required.

The "⇐" direction is more laborious. Assume that t is not accessible in G. We define an extension $\mathcal{A}'_G \supseteq \mathcal{A}_G$ and then show that $\mathcal{A}'_G \not\models Q(\mathbf{a})$, which implies $\mathcal{A}_G \not\models Q(\mathbf{a})$ as desired.

For all $v \in V$, let $\mathsf{acc}(v) = 1$ if v is accessible in G and $\mathsf{acc}(v) = 0$ otherwise. Further let $(u_1, v_1, w_1), \dots, (u_m, v_m, w_m)$ be an enumeration of all triples in E. We inductively define a sequence of ABoxes $\mathcal{A}_0, \mathcal{A}_1, \dots, \mathcal{A}_m$ as follows:

$$\mathcal{A}_0 \;=\; \bigcup_{s\in S} (\mathcal{A}^s_{\mathsf{source}} \cup t_1(a_s)) \;\cup\; \mathcal{A}^t_{\mathsf{target}} \cup t_0(a_t)$$

and if $\mathcal{A}_{i-1}$ has already been defined, let

$$\mathcal{A}_i \;=\; \mathcal{A}_{i-1} \;\cup\; \mathcal{A}^{u_i,v_i,w_i}_{\wedge} \;\cup\; t_{\mathsf{acc}(u_i)}(a_{u_i}) \;\cup\; t_{\mathsf{acc}(v_i)}(a_{v_i}) \;\cup\; t_{\mathsf{acc}(w_i)}(a_{w_i}).$$

Finally, define $\mathcal{A}'_G = \mathcal{A}_m$. It is straightforward to verify that

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

Set $\mathsf{ind}_V = \{a_v \mid v \in V\}$. With the *components* of $\mathcal{A}'_G$, we mean the ABoxes $\mathcal{A}^+_s := \mathcal{A}^s_{\mathsf{source}} \cup t_1(a_s)$ for all $s \in S$, $\mathcal{A}^+_{\mathsf{target}} := \mathcal{A}^t_{\mathsf{target}} \cup t_0(a_t)$, and $\mathcal{A}^+_{u,v,w} := \mathcal{A}^{u,v,w}_{\wedge} \cup t_{\mathsf{acc}(u)}(a_u) \cup t_{\mathsf{acc}(v)}(a_v) \cup t_{\mathsf{acc}(w)}(a_w)$ for all $(u, v, w) \in E$. By Points 4 to 8 of Lemma 46, the following types are realized in components:

1. $\mathsf{tp}_{\mathcal{A}^+_s,\mathcal{T}}(a_s) = t_1 = t_{\mathsf{acc}(s)}$ for every $s \in S$,
2. $\mathsf{tp}_{\mathcal{A}^+_{\mathsf{target}},\mathcal{T}}(a_t) = t_0 = t_{\mathsf{acc}(t)}$,
3. $\mathsf{tp}_{\mathcal{A}^+_{u,v,w},\mathcal{T}}(a_u) = t_{\mathsf{acc}(u)}$, $\mathsf{tp}_{\mathcal{A}^+_{u,v,w},\mathcal{T}}(a_v) = t_{\mathsf{acc}(v)}$ and $\mathsf{tp}_{\mathcal{A}^+_{u,v,w},\mathcal{T}}(a_w) = t_{\mathsf{acc}(w)}$.

Claim 1. Let $0 \le i \le m$. Then $\mathsf{tp}_{\mathcal{A}_i,\mathcal{T}}(a) = \mathsf{tp}_{\mathcal{C},\mathcal{T}}(a)$ for every component $\mathcal{C}$ of $\mathcal{A}'_G$ with $\mathcal{C} \subseteq \mathcal{A}_i$ and all $a \in \mathsf{ind}(\mathcal{C})$.

We prove Claim 1 by induction on i. For $i = 0$, the only relevant components are $\mathcal{A}^+_{\mathsf{target}}$ and the $\mathcal{A}^+_s$ for $s \in S$. Since $t \notin S$, all such components are disjoint from each other, so the statement is true.

For the induction step, assume that the statement has already been proven for some $i \geq 0$. There is a component $\mathcal{C}_i = \mathcal{A}^+_{u,v,w}$ such that $\mathcal{A}_{i+1} = \mathcal{A} \cup \mathcal{C}_i$. Note that $\mathcal{A}_i$ and $\mathcal{C}_i$ might share only individuals a_u, a_v and a_w. We have $\mathsf{tp}_{\mathcal{C}_i,\mathcal{T}}(a_u) = t_{\mathsf{acc}(u)}$, $\mathsf{tp}_{\mathcal{C}_i,\mathcal{T}}(a_v) = t_{\mathsf{acc}(v)}$ and $\mathsf{tp}_{\mathcal{C}_i,\mathcal{T}}(a_w) = t_{\mathsf{acc}(w)}$. By induction hypothesis, $\mathsf{tp}_{\mathcal{A}_i,\mathcal{T}}(a_u) = t_{\mathsf{acc}(u)}$ if a_u is shared, $\mathsf{tp}_{\mathcal{A}_i,\mathcal{T}}(a_v) = t_{\mathsf{acc}(v)}$ if a_v is shared, and $\mathsf{tp}_{\mathcal{A}_i,\mathcal{T}}(a_w) = t_{\mathsf{acc}(w)}$ if a_w is shared. The statement thus follows from Lemma 7 for $\mathcal{C} = \mathcal{C}_i$, and from Lemma 7 and the induction hypothesis for all $\mathcal{C} \neq \mathcal{C}_i$. This finishes the proof of Claim 1. Since $\mathcal{A}'_G = \mathcal{A}_m$, we may use Claim 1 with $\mathcal{A}'_G$ in place of $\mathcal{A}_i$.

We now use Claim 1 to show that $\mathcal{A}'_G, \mathcal{T} \not\models q(\mathbf{a})$. Assume to the contrary that $\mathcal{A}'_G, \mathcal{T} \models q(\mathbf{a})$, that is, there is a homomorphism h from $q(\mathbf{x})$ to $\mathcal{U}_{\mathcal{A}'_G,\mathcal{T}}$ such that $h(\mathbf{x}) = \mathbf{a}$. Note that any two distinct individuals from ind_V have distance at least $|q|$ in $\mathcal{A}'_G$, since the same is true for the individuals b, c, d in $\mathcal{A}_\wedge$. Since q is connected, there can thus be at most one such individual in the range of h.

If no individual from ind_V is in the range of h, then q being connected implies that there is a single component $\mathcal{C}$ of $\mathcal{A}'_G$ such that the range of h contains only individuals from $\mathcal{C}$ and anonymous elements below them.

First assume that $\mathcal{C}$ is $\mathcal{A}^+_{\mathsf{target}}$. We can construct a homomorphism h' from q to $\mathcal{U}_{\mathcal{A}^+_{\mathsf{target}},\mathcal{T}}$ by setting $h'(x) = h(x)$ if $h(x) \in \mathsf{ind}(\mathcal{A}'_G)$ and using Claim 1 and Lemma 3 if $h(x)$ is an anonymous element. Thus, $\mathcal{A}^+_{\mathsf{target}}, \mathcal{T} \models q(\mathbf{a})$, contradicting Condition 3 of Definition 20.

Now, assume that $\mathcal{C}$ is $\mathcal{A}^+_s$ or $\mathcal{A}^+_{u,v,w}$ for some $s \in S$ or $(u, v, w) \in E$. We only treat the former, since the latter is completely analogous. Since the constants from $\mathbf{a}$ are not in $\mathcal{A}^+_s$, q is Boolean. We can construct a homomorphism h' from q to $\mathcal{U}_{\mathcal{A}^+_s,\mathcal{T}}$ as in the previous case. We can further construct a homomorphism h'' from $\mathcal{U}_{\mathcal{A}^+_s,\mathcal{T}}$ to $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ by mapping every $a \in \mathsf{ind}(\mathcal{A}^+_s)$ to the individual in $\mathcal{A}$ that a is a copy of and using Points 9 and 10 from Lemma 46 and Lemma 3 to define $h''(d)$ for anonymous elements d. The composition g of h' and h'' is a homomorphism from q to $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ that is not core close since its range falls in the subtree of $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ rooted at c and the distance of c from the core is at least $|q|$. This contradicts Condition 5 of Definition 20.

Next, assume that the range of h contains a_v for some $v \in \mathsf{ind}_V$. We argue that q is Boolean and treeifiable. Since q is connected and the distance between the core in $\mathcal{A}^t_{\mathsf{target}}$ and a_v is at least $|q|$, q is Boolean. To show that q is treeifiable, consider the restriction $\mathcal{U}$ of $\mathcal{U}_{\mathcal{A}'_G,\mathcal{T}}$ to all elements within distance less than $|q|$ from a_v. Then $\mathcal{U}$ is almost tree-shaped in the sense that a_v is the only element that can have multiple predecessors, namely one predecessor for every triple in E where v appears in the first two components, and potentially one more predecessor if $v = t$, the target node. By Condition 6 of Definition 20, there is a unique sequence of roles $r_n \cdots r_1$ such that in each path $d_m s_m \cdots d_2 s_2 d_1 s_1 a_v$ in $\mathcal{U}$, $s_m \cdots s_1$ is a postfix of $r_n \cdots r_1$. We can thus obtain a tree-shaped interpretation $\mathcal{U}'$ from $\mathcal{U}$ by exhaustively identifying elements d_1, d_2 whenever $(d_1, e), (d_2, e) \in r^\mathcal{I}$ for some e and r. The construction of $\mathcal{U}'$ yields a homomorphism g from $\mathcal{U}$ to $\mathcal{U}'$ in an obvious way. The composition $g \circ h$ is a homomorphism from q to $\mathcal{U}'$. Consequently, q is treeifiable and the TBox has been extended with $C \sqsubseteq A_C$ for all $C \in \mathcal{C}_q$.

Claim 2. $a_v \in C^{\mathcal{U}'}$ iff $a_v \in C^{\mathcal{U}}$ for all $C \in \mathcal{C}_q$.

The "$\Leftarrow$" direction is immediate, since there is a homomorphism from $\mathcal{U}$ to $\mathcal{U}'$. For the "$\Rightarrow$" direction let $C = \exists r_n^-.\cdots\exists r_1^-.D$ where $r_1 \cdots r_n$ is a (potentially empty) role path in q^{tree} and D is $\top$ or a concept name from q or a CQ from $\mathsf{trees}(q)$ viewed as an $\mathcal{EL}$-concept. We prove the statement by induction on n, the length of the role path in C. If $n = 0$, then $C = D$ and C is an $\mathcal{EL}$-concept. By construction of $\mathcal{U}'$, it is clear that $a_v \in C^{\mathcal{U}'}$ implies $a_v \in C^{\mathcal{U}}$. Now assume the statement has been proven for $n-1$ and consider $C = \exists r_n^-.\cdots\exists r_1^-.D$. If D is $\top$ or a concept name, the statement again follows from the construction of $\mathcal{U}'$. What remains is the case where $D = \exists r.E$ for some $\mathcal{EL}$-concept E. Since $a_v \in C^{\mathcal{U}'}$, there is a path $d_1 r_1 d_2 \cdots r_{n-1} d_{n-1} r_n a_v$ in $\mathcal{U}'$ and $d_1 \in (\exists r.E)^{\mathcal{U}'}$. If there is an $e \neq d_2$ with $(d_1, e) \in r^{\mathcal{U}'}$ and $e \in E^{\mathcal{U}'}$, then again $a_v \in C^{\mathcal{U}}$ by construction of $\mathcal{U}'$. Assume that this is not the case, that is, $d_1 \in (\exists r.E)^{\mathcal{U}'}$ is true only because $r_1 = r$ and $d_2 \in E^{\mathcal{U}'}$. Then $a_v \in (\exists r_n^-.\cdots\exists r_2^-.E)^{\mathcal{U}'}$. Let $E = A_1 \sqcap \cdots \sqcap A_{n_1} \sqcap \exists s_1.E_1 \cdots \sqcap \exists s_{n_2}.E_{n_2}$ and define Γ to be the set of all concepts of the form $\exists r_n^-.\cdots\exists r_2^-.A_i$ or $\exists r_n^-.\cdots\exists r_2^-.E_j$ for any relevant i and j. Since $\Gamma \subseteq \mathcal{C}_q$ and since the role path in every concept from Γ has length $n-1$, we can apply the induction hypothesis, so $a_v \in G^{\mathcal{U}}$ for every $G \in \Gamma$. By Claim 1, we have $\mathsf{tp}_{\mathcal{A}'_G,\mathcal{T}}(a_v) \in \{t_0, t_1\}$. Let

$$\mathcal{B} = \begin{cases} \mathcal{A} & \text{if } \mathsf{tp}_{\mathcal{A}'_G,\mathcal{T}}(a_v) = t_1 \\ \mathcal{A}_c \cup t_0(c) & \text{if } \mathsf{tp}_{\mathcal{A}'_G,\mathcal{T}}(a_v) = t_0, \end{cases}$$

so it follows from $\mathsf{tp}_{\mathcal{B},\mathcal{T}}(b) = \mathsf{tp}_{\mathcal{A}'_G,\mathcal{T}}(a_v)$ that $a_v \in A_{C'}^{\mathcal{U}_{\mathcal{A}'_G,\mathcal{T}}}$ iff $b \in A_{C'}^{\mathcal{U}_{\mathcal{B},\mathcal{T}}}$ for all $C' \in \mathcal{C}_q$. Since both $\mathcal{U}_{\mathcal{A}'_G,\mathcal{T}}$ and $\mathcal{U}_{\mathcal{B},\mathcal{T}}$ are universal models and by construction of $\mathcal{T}$, $a_v \in C'^{\mathcal{U}_{\mathcal{A}'_G,\mathcal{T}}}$ iff $b \in C'^{\mathcal{U}_{\mathcal{B},\mathcal{T}}}$ for all $C' \in \mathcal{C}_q$. By choice of $\mathcal{U}$, the same is true when $\mathcal{U}_{\mathcal{A}'_G,\mathcal{T}}$ is replaced with $\mathcal{U}$. Thus, $b \in G^{\mathcal{U}_{\mathcal{B},\mathcal{T}}}$ for all $G \in \Gamma$, as well as for all G of the form $\exists r_n^-.\cdots\exists r_1^-.\top$. Since in $\mathcal{U}_{\mathcal{B},\mathcal{T}}$, b has a unique ancestor path up to length n, $b \in C^{\mathcal{U}_{\mathcal{B},\mathcal{T}}}$. Consequently, $a_v \in C^{\mathcal{U}}$ as desired. This finishes the proof of Claim 2.

To reach a contradiction, we now show that there is a homomorphism from q to $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ with b in its range. This is sufficient since b has distance at least $|q|$ from the core and q is connected, so such a homomorphism is not core close, contradicting Condition 5 of Definition 20. Recall that a_v is in the range of homomorphism h. By Claim 1, we have $\mathsf{tp}_{\mathcal{A}'_G,\mathcal{T}}(a_v) \in \{t_0, t_1\}$, so $\mathsf{tp}_{\mathcal{A}'_G,\mathcal{T}}(a_v) \subseteq t_1 = \mathsf{tp}_{\mathcal{A},\mathcal{T}}(b)$. It thus follows from Claim 2 that $a_v \in C^{\mathcal{U}'}$ implies $b \in C^{\mathcal{U}_{\mathcal{A},\mathcal{T}}}$ for all $C \in \mathcal{C}_q$. The same is true if we replace $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ with its restriction $\mathcal{U}''$ to all elements that have distance at most $|q|$ from b. Note that $\mathcal{U}''$ is tree-shaped and recall that $g \circ h$ is a homomorphism from q to $\mathcal{U}'$. We can apply Lemma 19 to this homomorphism, with $\mathcal{U}', a_v$ in place of $\mathcal{I}_1, d_1$ and $\mathcal{U}'', b$ in place of $\mathcal{I}_2, d_2$, obtaining a homomorphism from q to $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ with b in its range, as desired. $\square$

Appendix D. Proofs for Section 4.2

Lemma 29. *Let $\mathcal{A}$ be an ABox of pathwidth at most k, $\mathbf{a} \in \mathsf{ind}(\mathcal{A})^{\mathsf{ar}(q)}$, and $w \in \Gamma^*$ a word that encodes $(\mathcal{A}, \mathbf{a})$. Then $\mathcal{A} \models Q(\mathbf{a})$ if and only if $w \in L(\mathfrak{A})$.*

Proof. We start by proving that the states of the form s_A^a, used for checking the existence of a derivation for $A(a)$, work as intended.

Claim. Let $a \in V_i$ and a' the name of its copy in the $\mathcal{B}_i$. Then there exists a successful run starting from the configuration $(s_A^{a'}, i)$ if and only if $\mathcal{A}, \mathcal{T} \models A(a)$.

First, assume that $\mathcal{A}, \mathcal{T} \models A(a)$. We have to construct a successful run starting from the configuration $(i, s_A^{a'})$. By Lemma 28, there exists a derivation tree for $A(a)$. The statement can be proved by induction on the minimal number k such that $A(a)$ has a derivation tree of depth k. If $k = 0$, then $A(a) \in \mathcal{A}$, so $A(a') \in \mathcal{B}_i$, and in this case we have $\delta(s_A^a, (\mathbf{b}_i, \mathcal{B}_i, \mathbf{c}_i, f_i)) = \mathsf{true}$, which means the run is successful. Now let $k > 0$ and consider a derivation tree for $A(a)$ of depth k.

- If the children of the root are of the form $B_1(a), \ldots, B_n(a)$ such that $\mathcal{T} \models B_1 \sqcap \ldots \sqcap B_n \sqsubseteq A$, then choose the set $Z = \{B_1(a'), \ldots, B_n(a')\}$ in the transition, so in the run, we add the children labeled with $(i, s_{B_j}^{a'})$ for all $1 \leq j \leq n$. By induction hypothesis, from all these configurations there exists a successful run, so these runs can be combined to obtain a successful run for $(i, s_A^{a'})$.
- If the root has one child labeled $B(b)$ and we have $\mathcal{T} \models \exists r.B \sqsubseteq A$, then there exist $b \in \mathsf{ind}(\mathcal{A})$ and $r(a, b) \in \mathcal{A}$. This individual b does not necessarily lie in $\mathcal{B}_i$, but by the properties of a path decomposition, there exists a bag V_j such that $a, b \in V_j$ and, since $a \in V_i$, we also have $a \in V_k$ for all k between i and j. We extend the run as follows: If $j < i$, then use the transition $(\mathsf{left}, s_A^{a'})$ for $i - j$ times. If $j > i$, then use the transition $(\mathsf{right}, s_A^{a'})$ for $j-i$ times. Then we are in the configuration $(j, s_A^{a'})$ and if we choose $Z = \{B(b')\}$, we can extend the run successfully by the induction hypothesis.

For the other direction, assume that there is a successful run starting from the configuration $(i, s_A^{a'})$. We have to argue that $\mathcal{A}, \mathcal{T} \models A(a)$. The proof is by induction on the depth of the run. If the run has depth 0, i.e. the configuration $(i, s_A^{a'})$ does not have any successors, then we must have $\delta(s_A^{a'}, (\mathbf{b}_i, \mathcal{B}_i, \mathbf{c}_i, f_i)) = \mathsf{true}$. This is only the case if $A(a') \in \mathcal{B}_i$, so $A(a) \in \mathcal{A}$ and clearly, $\mathcal{A}, \mathcal{T} \models A(a)$. Now assume the run has depth $k > 0$. If the root node has a successor labeled $(i - 1, s_A^{a'})$ or $(i + 1, s_A^{a'})$, by induction hypothesis we have $\mathcal{A}, \mathcal{T} \models A(a)$. If the root node does not have a successor of this kind, then there exists a set Z and successors $(i, s_B^{b'})$ for all $B(b') \in Z$ such that $\mathcal{B} \cup Z \models A(a')$. By induction hypothesis, we have $\mathcal{A}, \mathcal{T} \models B(b)$ for all $B(b') \in Z$. Together, this gives $\mathcal{A}, \mathcal{T} \models A(a)$. This finishes the proof of the claim.

Now we are ready to prove the lemma.

"$\Rightarrow$". Let $\mathcal{A} \models Q(\mathbf{a})$ and let $(\mathbf{b}_1, \mathcal{B}_1, \mathbf{c}_1, f_1) \dots (\mathbf{b}_n, \mathcal{B}_n, \mathbf{c}_n, f_n)$ be an encoding of $(\mathcal{A}, \mathbf{a})$ based on some $(j, k+1)$ path decomposition $V_1, \dots, V_n$ of $\mathcal{A}$. There exists a homomorphism h_0 from q to $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ that maps the answer variables to $\mathbf{a}$. We use h_0 to guide the accepting run of $\mathfrak{A}$ on the word encoding $(\mathcal{A}, \mathbf{a})$. In the i-th step of the main branch of the run, always choose the partial q-match $h \in H$ according to h_0, i.e. if $h_0(x) = a \in \mathsf{ind}(\mathcal{A}) \cap V_i$ then $h(x) = (a', \mathsf{named})$, and if $h_0(x) = b$ for some anonymous individual b that lies in the subtree below some $c \in \mathsf{ind}(\mathcal{A})$ then $h(x) = (c', \mathsf{anon})$. As the explanation set for h we can just choose $Z_h = \{A(a') \mid a \in V_i \text{ and } \mathcal{A}, \mathcal{T} \models A(a)\}$.

We argue that following these choices, the main path will be successful, i.e. the leaf of the main branch is labeled with $(s^g_{V,W}, \dashv)$ such that $V = \mathsf{var}(q)$, W is the set of all binary atoms of q and g the empty map. Let $x \in \mathsf{var}(q)$. Then either $h_0(x) \in \mathsf{ind}(\mathcal{A})$ or $h_0(x)$ is an anonymous individual below some $b \in \mathsf{ind}(\mathcal{A})$. If $h_0(x) \in \mathsf{ind}(\mathcal{A})$, then let V_i be the first bag such that $h_0(x) \in V_i$ and thus there is a copy of $h_0(x)$ in $\mathsf{ind}(\mathcal{B}_i)$. Thus, in the i-th step of the main branch, x is added to V. Similarly, if $h_0(x)$ is an anonymous individual below some $b \in \mathsf{ind}(\mathcal{A})$, then let V_i be the first bag such that $b \in V_i$. Again, one can conclude that x is added to V in the i-th step of the main branch. Overall, it follows that $V = \mathsf{var}(q)$. Now, let $r(x, y)$ be a binary atom from q. If both $h_0(x)$ and $h_0(y)$ are in $\mathsf{ind}(\mathcal{A})$, then, since $V_1, \dots, V_n$ is a path decomposition of $\mathcal{A}$, there exists a bag V_i such that $h_0(x), h_0(y) \in V_i$, so there exists a copy of $r(h_0(x), h_0(y))$ in $\mathcal{B}_i$ and in the i-th step of the main branch, $r(x, y)$ is added to W. If at least one of $h_0(x)$ and $h_0(y)$ is not in $\mathsf{ind}(\mathcal{A})$, but is an anonymous individual below some $b \in \mathsf{ind}(\mathcal{A})$, then either both $h_0(x)$ and $h_0(y)$ are mapped to anonymous individuals below b or one of them is mapped to b. In any case, $r(x, y)$ is added to W in the i-th step of the main branch. Overall, it follows that W is the set of all binary atoms of q. Finally, g must be the empty map, since $\mathbf{c}_n = \emptyset$.

It follows immediately from the claim above that the other paths will be successful as well, i.e. whenever $\mathcal{A}, \mathcal{T} \models A(a)$ for some $a \in V_i$, then there is a successful run that starts at $(s^{a'}_A, i)$. This concludes the proof of the first direction.

"$\Leftarrow$". Assume that there is a successful run of $\mathfrak{A}$ on the input word $w = (\mathbf{b}_1, \mathcal{B}_1, \mathbf{c}_1, f_1) \dots (\mathbf{b}_n, \mathcal{B}_n, \mathbf{c}_n, f_n)$. The run must have one main path with states of the form $s^g_{V,W}$. In every step of the main path, one partial q-match h together with an explanation set Z_h is chosen. From these partial q-matches we can construct a map h_0 from $\mathsf{var}(q)$ to the universal model of $\mathcal{A}$ and $\mathcal{T}$ in the following way: Whenever a partial q-match h maps a variable x to (a, named), we set $h_0(x) = a$. Whenever a partial q-match h maps a variable x to (a, anon), then consider the explanation set Z_h. By the definition of the explanation set, we have $(\{B \mid B(a) \in Z_h\}, h^{-1}(a, \mathsf{named}), h_1^{-1}(a)) \in \mathcal{R}$, so there exists a homomorphism from $h_1^{-1}(a)$ to the canonical model of $\{B \mid B(a) \in Z_h\}$ that maps precisely the variables from $h^{-1}(a, \mathsf{named})$ to the root, which is the homomorphism we use to build h_0. From the condition $V \cap \mathsf{dom}(h) = \mathsf{dom}(g)$ it follows that a partial q-match chosen later in the run will not assign a different image to a variable that has appeared earlier in the domain of a partial q-match, so h_0 is well defined.

We show that h_0 is indeed a homomorphism from q to $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ with $q(\mathbf{x}) = \mathbf{a}$. Since the main branch ends in a configuration $(s^g_{V,W}, \dashv)$, where $V = \mathsf{var}(q)$, we know that $\mathsf{dom}(h_0) = \mathsf{var}(q)$. We argue that every atom of q is satisfied by h_0.

- Let $A(x)$ be a unary atom from q such that $h_0(x) \in \mathsf{ind}(\mathcal{A})$. Let h be the partial q-match that determined $h_0(x)$, so $h(x) = (a', \mathsf{named})$ for some $a' \in \mathsf{ind}(\mathcal{B})$, and let Z_h be the explanation set chosen in the run, so the configuration has children labeled (i, s^b_B) for every $B(b) \in Z_h$. Since the run is successful and we know from the claim above that a partial run starting from the configuration $(i, s^{b'}_B)$ is successful if and only if $\mathcal{A}, \mathcal{T} \models B(b)$, we have $\mathcal{A}, \mathcal{T} \models B(b)$ for all $B(b') \in Z_h$ and thus, $\mathcal{A}, \mathcal{T} \models A(a)$.

- Let $r(x, y)$ be a binary atom from q such that both $h_0(x)$ and $h_0(y)$ are in $\mathsf{ind}(\mathcal{A})$. Since the main branch ends in the state $s^g_{V,W}$, where W is the set of all binary atoms from q, there must be one step in the main branch, where $r(x, y)$ has been added to W, say the i-th step. This means that both x and y lie in $\mathsf{dom}(h)$, where h is the partial q-match chosen in the i-th step. Since h respects role atoms, we have $r(h_0(x), h_0(y)) \in \mathcal{A}$.

- Let $A(x)$ be a unary atom from q such that $h_0(x)$ is an anonymous individual below some $a \in \mathsf{ind}(\mathcal{A})$. Let h be the partial q-match that determined $h_0(x)$, so $h(x) = (a', \mathsf{anon})$, and let Z_h be the explanation set chosen in the run. By definition of an explanation set, there is a partial homomorphism from q with x in its domain to the universal model of $\{B(a) \mid B(a) \in Z_h\}$, which was used to define h_0 on x, so we have $\mathcal{A}, \mathcal{T} \models A(h_0(x))$.

- Let $r(x, y)$ be a binary atom from q such that at least one of $h_0(x)$ and $h_0(y)$ is an anonymous individual. Then the argument is similar to the previous case.

□

Appendix E. Proofs for Section 6

Lemma 35. *For all $1 \leq k \leq n$,*

1. *$\mathcal{A}^n_k \in \mathcal{M}_{Q_k}$;*
2. *$\mathsf{br}(\mathcal{A}^n_k) = k$;*
3. *$\mathcal{A}^n_k$ has exactly $\ell^k_2(n)$ leaves.*

Proof. Let $k \geq 1$. We say that an individual a in an ABox $\mathcal{A}$ has type i if i is the largest number such that $\{A_0, A_1, \ldots, A_i\} \subseteq \mathsf{tp}_{\mathcal{A},\mathcal{T}_k}(a)$. We prove the following statement by induction on n:

(*) For $1 \leq k \leq n$, the root of $\mathcal{A}^n_k$ has type k.

If $n = 1$, it is easy to check that the root of $\mathcal{A}_1^1$ has type 1. Now, let $n > 1$ and $k \leq n$. We distinguish two cases. If $k = n$, then $\mathcal{A}_k^n$ with root a contains assertions $r(a, a_r), s(a, a_s)$ and each of a_r, a_s is the root of a copy of $\mathcal{A}_{k-1}^{n-1}$. By induction hypothesis and Lemma 5, a_r and a_s are of type $k-1$. By the construction of $\mathcal{T}_k$, a has type k. If $k < n$, then $\mathcal{A}_k^n$ with root a contains assertions $t(a, a_t), u(a, a_u)$, and a_t is the root of a copy of $\mathcal{A}_{k-1}^{n-1}$ and a_u is the root of a copy of $\mathcal{A}_k^{n-1}$. By induction hypothesis and Lemma 5, a_t has type $k-1$ and a_u has type k. By the construction of $\mathcal{T}_k$, a has type k. This finishes the proof of (*).

Now we prove point 1. By (*), the root of $\mathcal{A}_k^n$ is an answer to Q_k. We prove by induction on n that $\mathcal{A}_k^n$ is minimal. If $n = 1$, it is easy to check that removing any assertion from $\mathcal{A}_1^1$ makes Q_1 false at the root. If $n > 1$ and $k = n$, then $\mathcal{A}_k^n$ with root a contains assertions $r(a, a_r), s(a, a_s)$ and each of a_r, a_s is the root of a copy of $\mathcal{A}_{k-1}^{n-1}$. By (*) and Lemma 5 a_r and a_s are both of type $k-1$. By induction hypothesis, removing any assertion in one of the subtrees rooted at a_r or a_s reduces the type of a_r or a_s, which implies that the type of a is less than k. Similarly, if any of the assertions $r(a, a_r)$ and $s(a, a_s)$ is removed, the type of a becomes $k-1$. Thus, removing any assertion from $\mathcal{A}_k^n$ reduces the type of the root, so $\mathcal{A}_k^n$ is minimal. The case where $k < n$ is similar.

We now prove Point 2 by induction on n. If $n = 1$, then $k = 1$, and it is easy to see that $\mathsf{br}(\mathcal{A}_1^1) = 1$. Now, let $n > 1$ and $1 \leq k \leq n$. We distinguish two cases. If $k = n$, then $\mathcal{A}_k^n$ with root a contains assertions $r(a, a_r), s(a, a_s)$ and each of a_r, a_s is the root of a copy of $\mathcal{A}_{k-1}^{n-1}$. By induction hypothesis, $\mathsf{br}(\mathcal{A}^{a_r}) = \mathsf{br}(\mathcal{A}^{a_s}) = k - 1$. By the algorithm that computes the branching number given in Section 4, $\mathsf{br}(\mathcal{A}_k^n) = (k-1) + 1 = k$. If $k < n$, then $\mathcal{A}_k^n$ with root a contains assertions $t(a, a_t), u(a, a_u)$, and a_t is the root of a copy of $\mathcal{A}_{k-1}^{n-1}$ and a_u is the root of a copy of $\mathcal{A}_k^{n-1}$. By induction hypothesis, $\mathsf{br}(\mathcal{A}^{a_t}) = k - 1$ and $\mathsf{br}(\mathcal{A}^{a_u}) = k$. Thus, $\mathsf{br}(\mathcal{A}_k^n) = \mathsf{max}(k, k-1) = k$.

Now we prove Point 3 by induction on n. If $n = 1$, then $k = 1$, and $\mathcal{A}_1^1$ has two leaves. Since the full binary tree of depth one does not have the full binary tree of depth two as a minor, two is also the largest number of leaves that a binary tree of depth one can have if a full binary tree of depth two as a minor is avoided. If $n > 1$ and $k = n$, then $\mathcal{A}_k^n$ takes the form of the full binary tree of depth n, so it has 2^n leaves. This is clearly the largest number of leaves that a binary tree of depth n can have that avoids a binary tree of depth $n+1$ as a minor, since it is the largest number of leaves that any binary tree of depth n can have. Now let $n > 1$ and $k < n$. We have to argue that $\mathcal{A}_k^n$ has $\ell_2^k(n)$ leaves. Let T be a binary tree of depth n that avoids the full binary tree of depth $k+1$ as a minor, and such that T has the maximum number of leaves, so T has $\ell_2^k(n)$ leaves. The root of T needs to have two successors, since otherwise, adding another successor to the root would increase the number of leaves by one, and adding another successor to the root can not lead to the existence of a full binary tree of depth $k+1$ as a minor. Now consider the two subtrees T_1 and T_2 rooted at the two successors of the root of T. If none of them had the full binary tree of depth k as a minor, we could increase the number of leaves by replacing T_1 by the tree that has $\ell_2^k(n-1)$ leaves, using the fact that $\ell_2^k(n)$ is monotone in both n and k. So we can assume that one of T_1 and T_2 has the full binary tree of depth k as a minor. Thus, the other one can not have the full binary tree of depth k as a minor, since then T would have the binary tree of depth $k+1$ as a minor. Again, by monotonicity of $\ell_2^k(n)$, we can conclude that one of T_1 and T_2 has $\ell_2^k(n-1)$ leaves and the other one has $\ell_2^{k-1}(n-1)$ leaves. By induction hypothesis, $\mathcal{A}_k^{n-1}$ has $\ell_2^k(n-1)$ leaves and $\mathcal{A}_{k-1}^{n-1}$ has $\ell_2^{k-1}(n-1)$ leaves. Thus, the number of leaves of $\mathcal{A}_k^n$ is equal to the number of leaves of T, which is $\ell_2^k(n)$. □

Lemma 36. *For every $k \geq 1$, Q_k is rewritable into linear Datalog.*

Proof. We show that $\mathsf{br}(Q_k) = k$, which by Proposition 18 implies that Q_k has bounded pathwidth which by Theorem 17 implies that Q_k is rewritable into linear Datalog.

Let $\mathcal{A} \in \mathcal{M}_{Q_k}$. We show that $\mathsf{br}(\mathcal{A}) = k$. First, let us analyse the types $\mathsf{tp}_{\mathcal{A},\mathcal{T}_k}(a)$, $a \in \mathsf{ind}(\mathcal{A})$, and the structure of $\mathcal{A}$. Since $\top \sqsubseteq A_0 \in \mathcal{T}_k$, none of the types $\mathsf{tp}_{\mathcal{A},\mathcal{T}_k}(a)$ 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 $x \in \{r,s,t,u\}$. We say that *a is of type i* if i is the largest integer such that $A_i \in \mathsf{tp}_{\mathcal{A},\mathcal{T}_k}(a)$ and that *a is of x-type j_x* if j_x is the largest integer such that $B_{x,j_x} \in \mathsf{tp}_{\mathcal{A},\mathcal{T}_k}(a)$.

Claim 1. Every individual in $\mathcal{A}$ has degree at most two and every individual of degree two is an rs-individual or a tu-individual.

We first argue that every individual has at most one x-successor for every $x \in \{r,s,t,u\}$. Assume to the contrary that there are 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. Then $B_{x,j}$, $B_{x,\ell}$, $B_{x,m}$ and $B_{x,n}$ are derived 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 $B_{x,\mathsf{max}\{j,\ell,m,n\}}$. Thus one of the individuals b,c can be removed without altering the result of the query.

Now we argue that every individual with degree greater than one is either an rs-individual or a tu-individual. All other combinations do not appear due to the minimality of $\mathcal{A}$. For example, assume that there is an rst-node a. Then some $B_{r,j}, B_{s,\ell}, B_{t,m}$ are derived at a, assume that j,ℓ,m are maximal with this property. If a is the root of $\mathcal{A}$, then the t-edge can 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 can be removed. So assume, a is a t-successor of its parent. If now $m \leq \mathsf{min}(j,\ell)$, the t-edge below a can be removed. If $m > \mathsf{min}(j,\ell)$, but then both the r-edge and the s-edge can be removed. In either case, $\mathcal{A}$ is not minimal. This finishes the proof of Claim 1.

Using minimality of $\mathcal{A}$, it can be argued that for every x-individual a (an individual with only one outgoing edge) with $x \in \{r,s,t,u\}$, there is some $b \in \mathsf{ind}(\mathcal{A})$ with $x(b,a) \in \mathcal{A}$ and it follows that a path from one branching point to the next is always a chain of the same role.

Claim 2. a is of type i iff $\mathsf{br}(\mathcal{A}^a) = i$ 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 in $\mathcal{A}^a$. If $n = 1$, then a is a leaf itself, thus of type 0, and the statement follows. Now let $n > 1$ and let a be an individual of degree two with n leaves below it. We only argue the 'if' direction, the 'only if' direction can be argued similarly. So assume that $\mathsf{br}(\mathcal{A}^a) = i$ for some $i \geq 1$. Then by Claim 1, a has two outgoing paths that both reach two nodes b and c that are a leaf or of degree two. Let $j = \mathsf{br}(\mathcal{A}^b)$ and $\ell = \mathsf{br}(\mathcal{A}^c)$ and w.l.o.g. assume $j \geq \ell$. By induction hypothesis, b is of type j and c is of type ℓ. There are two possibilities: Either $j = \ell$, which implies $i = j + 1 = \ell + 1$, or $j > \ell$, which implies $i = j$. In case $j = \ell$, a must be an rs-individual. In fact, assuming a was a tu-individual, then $A_i(a)$ would be derived using $B_{t,i-1} \sqcap B_{u,i} \sqsubseteq A_i$, so a full binary tree of depth i below the t-successor of a is not needed and one could remove any leaf below the t-successor of a (contradicting minimality of $\mathcal{A}$), decreasing the depth of the largest binary tree minor by at most one. So since a is an rs-individual, $B_{r,i-1} \sqcap B_{s,i-1} \sqsubseteq A_i$ applies and a has type i. In case $j > \ell$, one can argue in a similar way that a must be a tu-individual and $j = \ell + 1$, and it follows that a has type i. This finishes the proof of Claim 2.

Since $\mathcal{A} \models Q_k(a)$ for the root a of $\mathcal{A}$, we know that a is of type k, so Claim 2 says that $\mathsf{br}(\mathcal{A}) = k$. ☐

Lemma 37. *$(d-1)^k(n-k)^k \leq \ell_d^k(n) \leq (k+1)(d-1)^k n^k$ for all $d, k \geq 0$ and $n \geq 2k$.*

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

$$\ell_d^k(n) = \sum_{i=0}^{k} (d-1)^i \binom{n}{i} \tag{$*$}$$

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 one for $i = k$ if $n \geq 2k$.

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

$$\ell_d^k(n) = \ell_d^k(n-1) + (d-1)\ell_d^{k-1}(n-1) \tag{$**$}$$

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 $T_1, \ldots, T_d$ 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 $\ell_d^k(0) = \ell_d^0(n) = 1$ for all d, k, n, thus $(*)$ holds for all cases where $k = 0$ or $n = 0$. Now let $k \geq 1$ and $n \geq 1$ and assume that $(*)$ holds for $\ell_d^k(n)$ and for $\ell_d^{k-1}(n)$. We show that it also holds for $\ell_d^k(n+1)$:

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

□

Proposition 38. *Let $m \geq 0$. For all $k \geq 1$, Q_{2k+3} is not rewritable into a linear Datalog program of diameter k on the class of ABoxes $\mathfrak{C}_m$.*

Proof. Let $m \geq 0$. 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 $\mathfrak{C}_m$. Choose n very large (we will make this precise later) and consider the ABox $\mathcal{A}$ that can be obtained from $\mathcal{A}_{2k+3}^n$ by subdividing every edge into a path of length $m+1$ of the same role. Using Lemma 35, it is easy to see that $\mathcal{A} \in \mathcal{M}_{Q_{2k+3}}$ and that $\mathcal{A}$ has $\ell_2^{2k+3}(n)$ leaves, so from Lemma 37 it follows that $\mathcal{A}$ has at least $(n-(2k+3))^{2k+3}$ leaves.

Let a_0 be the root of $\mathcal{A}$. From $\mathcal{A} \in \mathcal{M}_{Q_{2k+3}}$, it follows that $\mathcal{A}, \mathcal{T} \models \Pi(a_0)$ and thus there is a derivation D of $\Pi(a_0)$ in $\mathcal{A}$. Consider the ABox $\mathcal{A}_D$ induced by derivation D. By Lemma 27, we have the following:

1. $\mathcal{A}_D \models \Pi(a_0)$;
2. there is a homomorphism from $\mathcal{A}_D$ to $\mathcal{A}$ that is the identity on 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 6;
- remove all assertions that involve an individual a that is not reachable from a_0 in $G_{\mathcal{A}}$ by a path, this is possible according to Lemma 5.

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 $\mathcal{A}_D \models Q_k(a_0)$ iff $\mathcal{B} \models Q_k(a_0)$ iff $\mathcal{B} \models \Pi(a_0)$.

Choose a homomorphism h from $\mathcal{B}$ to $\mathcal{A}$ that is the identity on 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 $\mathcal{A} \in \mathcal{M}_{Q_{2k+3}}$. Let $a_1, \dots, a_w$ be the leaves of $\mathcal{A}$, $w \geq (n-2k-3)^{2k+3}$. For each a_i, choose a b_i with $h(b_i) = a_i$. All individuals in $b_1, \dots, b_w$ must be distinct since h has a different value for each of them.

By construction, $\mathcal{B}$ is connected. Since there is a homomorphism from $\mathcal{B}$ to $\mathcal{A}$, $\mathcal{B}$ must 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 $b_1, \dots, b_w$ 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 $n(m+1)$.

Assume that $\mathcal{B}_t$ does not contain the full binary tree of depth $2k+3$ as a minor. Then by Lemma 37, the number of leaves of $\mathcal{B}_t$ is at most

$$(2k+3)(|\mathcal{T}|-1)^{2k+2}(n(m+1))^{2k+2},$$

which is a polynomial in n of degree $2k+2$. So if we choose n such that

$$(n-2k-3)^{2k+3} > (2k+3)(|\mathcal{T}|-1)^{2k+2}(n(m+1))^{2k+2}$$

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 $k+1$ [49], in contradiction to $\mathcal{B}_t$ having pathwidth at most k. □

Proposition 39. *Let $\ell \geq 1$. $Q_{8\ell+13}$ 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 show that Π_0 can be rewritten into a linear Datalog rewriting Π' of $Q_{8\ell+13}$ of diameter $4\ell+5$, in contradiction to Proposition 38.

We carry out a sequence of three rewriting steps. Informally, in the first rewriting we normalize the shape of rule bodies, in the second one we control the number of disconnected components in the rule body (or rather its restriction to the EDB relations), and in the third one we actually bound the diameter to $4\ell + 5$ by replacing each rule with a collection of rules.

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 consistently 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 call a variable in $q(\mathbf{y})$ *special* if it occurs in $\mathbf{x}$ or in the IDB atom in $q(\mathbf{y})$, if existent. Since Π_1 has width ℓ, there are at most 2ℓ special variables in each rule. We obtain a new rule body $q'(\mathbf{y}')$ from $q(\mathbf{y})$ in the following way:

1. remove the IDB atom (if existent), obtaining a forest-shaped rule body;

2. remove all trees that do not contain a special variable;

3. re-add the IDB atom (if existent).

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 $q_1(\mathbf{x}_1), \dots, q_m(\mathbf{x}_m)$ be all trees that have been removed from a rule body during the construction of Π_2. Let $\mathcal{A}_i$ be $q_i(\mathbf{x}_i)$ viewed as a Σ-ABox, $1 \leq i \leq m$. Note that each $\mathcal{A}_i$ must be in $\mathfrak{C}_k$. Let $\mathcal{B}$ be the disjoint union of the ABoxes $\mathcal{A}, \mathcal{A}_1, \dots, \mathcal{A}_m$, 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.

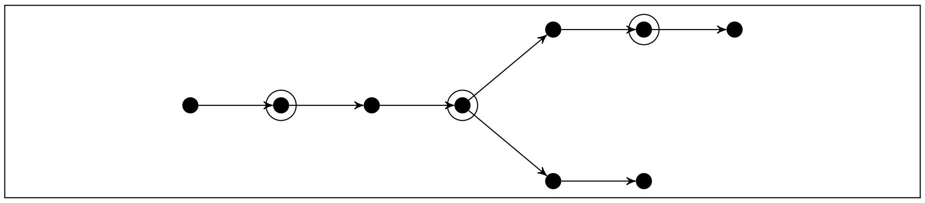

Figure E.9: The body $q(\mathbf{y})$ of a rule from Π_2 consists of one or several such trees, where at most one variable is branching. The branching variable and special variables are circled and they divide the body into five paths $q_i(\mathbf{y}_i)$.

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 Proposition 38.

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 (at most 2ℓ special variables and at most 2ℓ branching variables). Let $q'(\mathbf{y}')$ be the restriction of $q(\mathbf{y})$ to the variables in $\mathbf{y}'$; we can assume that each variable y from $\mathbf{y}'$ occurs in $q'(\mathbf{y}')$ since if this is not the case, we can add an atom $\top(y)$. By construction of Π_2, it can be verified that $q(\mathbf{y})$ is the union of $q'(\mathbf{y}')$ and path-shaped $q_1(\mathbf{y}_1), \ldots, q_n(\mathbf{y}_n)$ such that for $1 \leq i \leq n$

- $q_i(\mathbf{y}_i)$ contains only EDB atoms,
- each $q_i(\mathbf{y}_i)$ contains at most two variables from $\mathbf{y}'$ and each such variable is an end point of the path, and
- the queries $q_1(\mathbf{y}_1), \ldots, q_n(\mathbf{y}_n)$ only share variables from $\mathbf{y}'$.

The structure of $q(\mathbf{y})$ is illustrated in Figure E.9. We thus find linear Datalog programs $\Gamma_1, \ldots, \Gamma_n$ that are at most binary, of width at most two and diameter at most three such that for any Σ-ABox $\mathcal{A}$ and $\mathbf{a} \subseteq \mathsf{ind}(\mathcal{A})$, $\mathcal{A} \models \Gamma_i(\mathbf{a})$ if and only if there is a homomorphism h_i from $q_i(\mathbf{y}_i)$ to $\mathcal{A}$ such that $h_i(\mathbf{y}_i) = \mathbf{a}$. Let the goal relations of $\Gamma_1, \ldots, \Gamma_n$ be $G_1, \ldots, G_n$ and assume that G_i occurs in Γ_i only once, in a rule head $G_i(\mathbf{x}_i)$. We assume w.l.o.g. that the programs $\Gamma_1, \ldots, \Gamma_n$ do not share variables or IDB relations, and neither do they share variables or IDB relations with Π_2. In Π_3, we replace $\rho = S(\mathbf{x}) \leftarrow q(\mathbf{y})$ with the following rules:

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

- the rule $S(\mathbf{x}) \leftarrow X_\rho^{G_n}(\mathbf{y}', \mathbf{x}_n)$,

where the goal relations of $\Gamma_1, \dots, \Gamma_n$ become standard (non-goal) IDB relations. It can be verified that Π_3 is as required. $\square$

Appendix F. Proofs for Section 7

We describe the construction of $\mathfrak{A}_{\mathsf{derive}}$. By Lemma 28, $\mathcal{A}, \mathcal{T} \models A(a)$ if and only if there is a derivation tree for $A(a)$. We may thus construct $\mathfrak{A}_{\mathsf{derive}}$ so that it checks for the existence of a suitable derivation tree. The set of states of $\mathfrak{A}_{\mathsf{derive}}$ is

$$\begin{aligned} S_{\mathsf{derive}} = \{d_A \mid A \in \mathsf{CN}\} \cup \{d_A^a \mid A \in \mathsf{CN} \wedge a \in \mathsf{C}_{\mathsf{core}}\} \cup \\ \{d_r \mid r \in \mathsf{ROL}\} \cup \{d_a \mid a \in \mathsf{C}_{\mathsf{core}}\}. \end{aligned}$$

For brevity, let $\ell = |\mathcal{T}| \cdot |q|$ denote the outdegree of trees. First consider states of the form d_A and alphabet symbols $\sigma \in \Gamma_N$ that do not contain an element of $\mathsf{C}_{\mathsf{core}}$. If $A \in \sigma$ or $\top \sqsubseteq A \in \mathcal{T}$, set $\delta(d_A, \sigma) = \mathsf{true}$. Otherwise, set

$$\begin{aligned} \delta(d_A, \sigma) = & \Big(\bigvee_{\mathcal{T} \models A_1 \sqcap \ldots \sqcap A_n \sqsubseteq A} \bigwedge_{i=1}^{n} \langle 0 \rangle d_{A_i} \Big) \vee \Big(\bigvee_{\exists r.B \sqsubseteq A \in \mathcal{T}} \bigvee_{i=1}^{\ell} \langle i \rangle (d_B \wedge d_r) \Big) \\ & \vee \Big(\bigvee_{\exists r^-.B \sqsubseteq A} (d_r \wedge \langle -1 \rangle d_B) \Big) \end{aligned}$$

Now let $\sigma \in \Gamma_N$ contain $a \in \mathsf{C}_{\mathsf{core}}$. If $A \in \sigma$ or $\top \sqsubseteq A \in \mathcal{T}$, set $\delta(d_A, \sigma) = \mathsf{true}$. Otherwise, set

$$\begin{aligned} \delta(d_A, \sigma) = & \Big(\bigvee_{\mathcal{T} \models A_1 \sqcap \ldots \sqcap A_n \sqsubseteq A} \bigwedge_{i=1}^{n} \langle 0 \rangle d_{A_i} \Big) \vee \Big(\bigvee_{\exists r.B \sqsubseteq A \in \mathcal{T}} \bigvee_{i=1}^{\ell} \langle i \rangle (d_B \wedge d_r) \Big) \\ & \vee \Big(\bigvee_{\exists r^-.B \sqsubseteq A} (d_r \wedge \langle -1 \rangle d_B^a) \Big) \end{aligned}$$

Next, consider states of the form d_A^a and alphabet symbols $\sigma = (\mathcal{B}, \mathbf{a}) \in \Gamma_\varepsilon$. If $A(a) \in \mathcal{B}$ or $\top \sqsubseteq A \in \mathcal{T}$, set $\delta(d_A^a, \sigma) = \mathsf{true}$. Otherwise, set

$$\begin{aligned} \delta(d_A^a, \sigma) = & \Big(\bigvee_{\mathcal{T} \models A_1 \sqcap \ldots \sqcap A_n \sqsubseteq A} \bigwedge_{i=1}^{n} \langle 0 \rangle d_{A_i}^a \Big) \vee \Big(\bigvee_{\exists r.B \sqsubseteq A \in \mathcal{T}} \bigvee_{i=1}^{\ell} \langle i \rangle (d_B \wedge d_r \wedge d_a) \Big) \\ & \vee \Big(\bigvee_{\exists r.B \sqsubseteq A, r(a,b) \in \mathcal{B}} \langle 0 \rangle d_B^b \Big) \vee \Big(\bigvee_{\exists r^-.B \sqsubseteq A, r(b,a) \in \mathcal{B}} \langle 0 \rangle d_B^b \Big) \end{aligned}$$

Finally, consider states of the form d_r and d_a and alphabet symbols $\sigma \in \Gamma_N$. Set $\delta(d_r, \sigma) = \mathsf{true}$ if $r \in \sigma$ and $\delta(d_a, \sigma) = \mathsf{true}$ if $a \in \sigma$. For all remaining combinations of states and alphabet symbols, $\delta(\cdot, \cdot)$ is set to false.

Lemma 44. *Let $\mathcal{A}$ be a pseudo tree-shaped Σ-ABox with core $\mathcal{B}$, $\mathsf{ind}(\mathcal{B}) \subseteq \mathsf{C}_{\mathsf{core}}$, and $\mathbf{a}$ a tuple from $\mathsf{ind}(\mathcal{B})$. Then*

1. *if $\mathcal{A} \models Q(\mathbf{a})$ via a core close homomorphism, then the following is a matching completion of $\mathcal{B}$: $\{A(a) \mid a \in \mathsf{ind}(\mathcal{B})$ and $\mathcal{A}, \mathcal{T} \models A(a)\}$;*
2. *if there exists a matching completion M of $\mathcal{B}$ such that $\mathcal{A}, \mathcal{T} \models A(a)$ for all $A(a) \in M$, then $\mathcal{A} \models Q(\mathbf{a})$.*

Proof. Assume that $\mathcal{A} \models Q(\mathbf{a})$ via a core close homomorphism, that is, there is a core close homomorphism h from q to $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ with $h(\mathbf{x}) = \mathbf{a}$. Set $M = \{A(a) \mid a \in \mathsf{ind}(\mathcal{B})$ and $\mathcal{A}, \mathcal{T} \models A(a)\}$. Clearly, M is a completion of $\mathcal{B}$. It remains to argue that it is matching. Since h is core close, its range contains a core individual or an anonymous element that the chase has generated below a core individual.

We start with the former case. Let p be obtained from q by first identifying any two (quantified) variables x_1, x_2 such that $h(x_1) = h(x_2)$ is a non-core individual or an anonymous element, and then dropping all atoms that contain a variable x with $h(x)$ a core individual. Since $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ takes the form of $\mathcal{B}$ with a directed tree attached to each individual, p must be a disjoint union of tree-shaped CQs. Let $q_1, \dots, q_n$ be these CQs and for $1 \leq i \leq n$ let q_i' be obtained from q_i by reading the unique atom $r(x, y) \in p$ with $h(x)$ a core individual and y the root of q_i. Such an $r(x, y)$ must exist since the range of h contains a core individual and q is connected. Finally let $C_1, \dots, C_n$ be $q_1', \dots, q_n'$ viewed as $\mathcal{EL}$-concepts. It is not hard to verify that $C_1, \dots, C_n \in \mathsf{trees}(q)$ and thus $\mathcal{T}$ contains the CI $C_i \sqsubseteq A_{C_i}$ for $1 \leq i \leq n$. This implies that $A_{C_i}(b_i) \in M$ for $1 \leq i \leq n$, where b_i is such that $h(y) = b_i$, y the root of q_i'. We are now in the position to describe a homomorphism h' that shows $\mathcal{B}_M \models q(\mathbf{a})$. Start with setting $h'(x) = h(x)$ whenever $h(x)$ is a core individual. Then take any C_i. By construction, $\mathcal{B}_M$ contains a disjoint copy of $\mathcal{A}_{C_i}$ whose root is glued to b_i. We can thus extend h' to the variables in q_i' in a straightforward way.

Now for the case that the range of h contains no core individual, but an anonymous element that the chase has generated below a core individual b. Since q is connected, h then maps all variables in q to the subtree of anonymous elements that the chase has generated below b. From the definition of M and Lemma 3, it follows that the subtree of anonymous elements below b in $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ is isomorphic to the tree that is generated below b in $\mathcal{U}_{\mathcal{B}_M,\mathcal{T}}$. Consequently, we can find a homomorphism h' from q to the latter tree, showing that $\mathcal{B}_M, \mathcal{T} \models q$.

Now assume that there exists a matching completion M of $\mathcal{B}$ such that $\mathcal{A}, \mathcal{T} \models A(a)$ for all $A(a) \in M$. We argue that there is a homomorphism h from $\mathcal{B}_M$ to $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ that is the identity on $\mathcal{B}$. We construct h using the following set of homomorphisms. First, the identity on $\mathsf{ind}(\mathcal{B})$, and second, for every copy of some A_C that was glued to some $a \in \mathsf{ind}(\mathcal{B})$, the homomorphism $h_{A_C,a}$ defined as follows: Since $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ is a model of $\mathcal{A}$ and $\mathcal{T}$, and since $a \in A_C^{\mathcal{U}_{\mathcal{A},\mathcal{T}}}$, $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ contains a homomorphic image of A_C where the root is mapped to a. Let $h_{A_C,a}$ be this homomorphism from A_C to $\mathcal{U}_{\mathcal{A},\mathcal{T}}$. It can be checked that this gives a well-defined homomorphism h. In particular, the different homomorphism used to construct h are compatible, since the intersection of the domains of two such homomorphisms is either empty or consist of a single element a, but in the latter case $h_{A_C,a}(a) = a$, so it is compatible with the identity homomorphism and with every other homomorphism of the form $h_{A'_C,a}$.

Since M is matching, $\mathcal{B}_M \models q(\mathbf{a})$. Due to the homomorphism h from $\mathcal{B}_M$ to $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ we constructed, this implies $\mathcal{A} \models Q(\mathbf{a})$. □