Theoretically Optimal Datalog Rewritings for *OWL 2 QL* Ontology-Mediated Queries

M. Bienvenu[1], S. Kikot[2], R. Kontchakov[2], V. Podolskii[3], and M. Zakharyaschev[2]

[1] CNRS & University of Montpellier, France (`meghyn@lirmm.fr`)
[2] Birkbeck, University of London, U.K. (`{kikot,roman,michael}@dcs.bbk.ac.uk`)
[3] Steklov Mathematical Institute, Moscow, Russia (`podolskii@mi.ras.ru`)

Abstract. We show that, for *OWL 2 QL* ontology-mediated queries with (*i*) ontologies of bounded depth and conjunctive queries of bounded treewidth, (*ii*) ontologies of bounded depth and bounded-leaf tree-shaped conjunctive queries, and (*iii*) arbitrary ontologies and bounded-leaf tree-shaped conjunctive queries, one can construct and evaluate nonrecursive datalog rewritings by, respectively, LOGCFL, NL and LOGCFL algorithms, which matches the optimal combined complexity.

1 Introduction

Ontology-based data access (OBDA) via query rewriting [18] reduces the problem of finding answers to conjunctive queries (CQs) mediated by *OWL 2 QL* ontologies to standard database query answering. The question we are concerned with here is whether this reduction is optimal with respect to the combined complexity of query evaluation. Figure 1 (a) summarises what is known about the size of positive existential (PE), nonrecursive datalog (NDL) and first-order (FO) rewritings of *OWL 2 QL* ontology-mediated queries (OMQs) depending on the existential depth of their ontologies and the shape of their CQs [13, 9, 12, 3]. Figure 1 (b) shows the combined complexity of OMQ evaluation for the corresponding classes of OMQs [5, 14, 12, 3]. Thus, we see, for example, that PE-rewritings for OMQs with ontologies of bounded depth and CQs of bounded treewidth can be of super-polynomial size, and so not evaluable in polynomial time, while the evaluation problem for these OMQs is decidable in LOGCFL $\subseteq$ P. On the other hand, the OMQs in this class enjoy polynomial-size NDL-rewritings. However, these rewritings were defined using an argument from circuit complexity [3], and it has been unclear whether they can be constructed and evaluated in LOGCFL. The same concerns the class of OMQs with ontologies of bounded depth and bounded-leaf tree-shaped queries, which can be evaluated in NL, and the class of OMQs with arbitrary ontologies and bounded-leaf tree-shaped queries, which can be evaluated in LOGCFL.

In this paper, we consider OMQs in these three classes and construct NDL-rewritings that are theoretically optimal in the sense that the rewriting and evaluation can be carried out by algorithms of optimal combined complexity, that is, from the complexity classes LOGCFL, NL and LOGCFL, respectively. Such algorithms are known to be space efficient and highly parallelisable. We compared our optimal NDL rewritings with those produced by query rewriting engines Clipper [8] and Rapid [6], using a sequence of OMQs with linear CQs and a fixed ontology of depth 1.

The full version is available at `http://tinyurl.com/LogNDL-DL`.

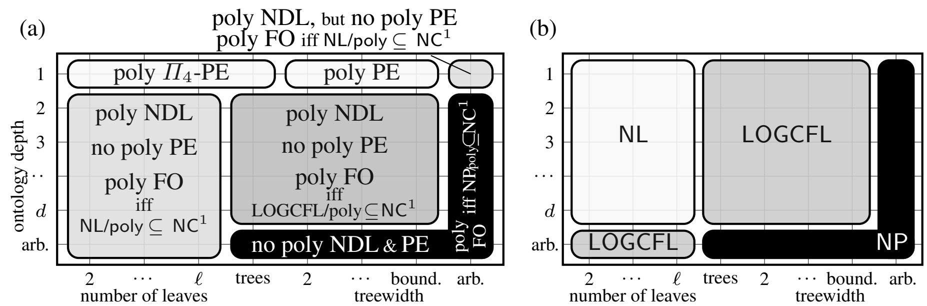


Fig. 1. (a) Size of OMQ rewritings; (b) combined complexity of OMQ evaluation.

2 Preliminaries

We give *OWL 2 QL* in the DL syntax with *individual names* a_i, *concept names* A_i, and *role names* P_i ($i \geq 1$). *Roles* R and *basic concepts* B are defined by

$$R \quad ::= \quad P_i \quad | \quad P_i^-, \qquad\qquad B \quad ::= \quad A_i \quad | \quad \exists R.$$

A *TBox*, $\mathcal{T}$, is a finite set of inclusions of the form

$$B_1 \sqsubseteq B_2, \qquad B_1 \sqcap B_2 \sqsubseteq \bot, \qquad R_1 \sqsubseteq R_2, \qquad R_1 \sqcap R_2 \sqsubseteq \bot.$$

An *ABox*, $\mathcal{A}$, is a finite set of atoms of the form $A_k(a_i)$ or $P_k(a_i, a_j)$. We denote by $\mathsf{ind}(\mathcal{A})$ the set of individual names in $\mathcal{A}$, and by $\boldsymbol{R}_\mathcal{T}$ the set of role names occurring in $\mathcal{T}$ and their inverses. We use $A \equiv B$ for $A \sqsubseteq B$ and $B \sqsubseteq A$. The semantics for *OWL 2 QL* is defined in the usual way based on interpretations $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ [2].

For every role $R \in \boldsymbol{R}_\mathcal{T}$, we take a fresh concept name A_R and add $A_R \equiv \exists R$ to $\mathcal{T}$. The resulting TBox is said to be in *normal form*, and we assume, without loss of generality, that all our TBoxes are in normal form. The subsumption relation induced by $\mathcal{T}$ is denoted by $\sqsubseteq_\mathcal{T}$: we write $S_1 \sqsubseteq_\mathcal{T} S_2$ if $\mathcal{T} \models S_1 \sqsubseteq S_2$, where S_1, S_2 are both either concepts or roles. We write $R(a,b) \in \mathcal{A}$ if $P(a,b) \in \mathcal{A}$ and $R = P$, or $P(b,a) \in \mathcal{A}$ and $R = P^-$; we also write $(\exists R)(a) \in \mathcal{A}$ if $R(a,b) \in \mathcal{A}$ for some b. An ABox $\mathcal{A}$ is called *H-complete with respect to* $\mathcal{T}$ in case

$$P(a,b) \in \mathcal{A} \quad \text{if} \quad R(a,b) \in \mathcal{A}, \ \text{for roles } P \text{ and } R \text{ with } R \sqsubseteq_\mathcal{T} P,$$
$$A(a) \in \mathcal{A} \quad \text{if} \quad B(a) \in \mathcal{A}, \ \text{for a concept name } A \text{ and basic concept } B \text{ with } B \sqsubseteq_\mathcal{T} A.$$

A *conjunctive query* (CQ) $\boldsymbol{q}(\boldsymbol{x})$ is a formula $\exists \boldsymbol{y}\, \varphi(\boldsymbol{x}, \boldsymbol{y})$, where φ is a conjunction of atoms $A_k(z_1)$ or $P_k(z_1, z_2)$ with $z_i \in \boldsymbol{x} \cup \boldsymbol{y}$ (without loss of generality, we assume that CQs do not contain constants). We denote by $\mathsf{var}(\boldsymbol{q})$ the variables $\boldsymbol{x} \cup \boldsymbol{y}$ of $\boldsymbol{q}$ and by $\mathsf{avar}(\boldsymbol{q})$ the *answer variables* $\boldsymbol{x}$. An *ontology-mediated query* (OMQ) is a pair $\boldsymbol{Q}(\boldsymbol{x}) = (\mathcal{T}, \boldsymbol{q}(\boldsymbol{x}))$, where $\mathcal{T}$ is a TBox and $\boldsymbol{q}(\boldsymbol{x})$ a CQ. A tuple $\boldsymbol{a}$ in $\mathsf{ind}(\mathcal{A})$ is a *certain answer* to $\boldsymbol{Q}(\boldsymbol{x})$ over an ABox $\mathcal{A}$ if $\mathcal{I} \models \boldsymbol{q}(\boldsymbol{a})$ for all models $\mathcal{I}$ of $\mathcal{T}$ and $\mathcal{A}$; in this case we write $\mathcal{T}, \mathcal{A} \models \boldsymbol{q}(\boldsymbol{a})$. If $\boldsymbol{x} = \emptyset$, then a certain answer to $\boldsymbol{Q}$ over $\mathcal{A}$ is 'yes' if $\mathcal{T}, \mathcal{A} \models \boldsymbol{q}$ and 'no' otherwise. We often regard a CQ $\boldsymbol{q}$ as the set of its atoms.

Every consistent *OWL 2 QL knowledge base* (KB) $(\mathcal{T}, \mathcal{A})$ has a *canonical model* $\mathcal{C}_{\mathcal{T},\mathcal{A}}$ with the property that $\mathcal{T}, \mathcal{A} \models \boldsymbol{q}(\boldsymbol{a})$ iff $\mathcal{C}_{\mathcal{T},\mathcal{A}} \models \boldsymbol{q}(\boldsymbol{a})$, for any CQ $\boldsymbol{q}$ and any $\boldsymbol{a}$ in $\mathsf{ind}(\mathcal{A})$. Thus, CQ answering in *OWL 2 QL* amounts to finding a homomorphism from the given CQ into the canonical model. Informally, $\mathcal{C}_{\mathcal{T},\mathcal{A}}$ is obtained from $\mathcal{A}$ by repeatedly applying the axioms in $\mathcal{T}$, introducing fresh elements as needed to serve as witnesses for the existential quantifiers. According to the standard construction (cf. [16]), the domain $\Delta^{\mathcal{C}_{\mathcal{T},\mathcal{A}}}$ of $\mathcal{C}_{\mathcal{T},\mathcal{A}}$ consists of words of the form $aR_1 \dots R_n$ $(n \geq 0)$ with $a \in \mathsf{ind}(\mathcal{A})$ and $R_1 \dots R_n \in \boldsymbol{R}_{\mathcal{T}}^*$ such that (*i*) $\mathcal{T}, \mathcal{A} \models \exists R_1(a)$ and (*ii*) $\exists R_i^- \sqsubseteq_{\mathcal{T}} \exists R_{i+1}$ and $R_i^- \not\sqsubseteq_{\mathcal{T}} R_{i+1}$, for $1 \leq i < n$. We let $\mathbf{W}_{\mathcal{T}}$ consist of all words $R_1 \dots R_n \in \boldsymbol{R}_{\mathcal{T}}^*$ satisfying condition (*ii*). A TBox $\mathcal{T}$ is *of depth* ω if $\mathbf{W}_{\mathcal{T}}$ is infinite, and *of depth* $d < \omega$, if d is the maximum length of the words in $\mathbf{W}_{\mathcal{T}}$.

A *datalog program*, Π, is a finite set of Horn clauses $\forall \boldsymbol{z}\, (\gamma_0 \leftarrow \gamma_1 \wedge \dots \wedge \gamma_m)$, where each γ_i is an atom $S(\boldsymbol{y})$ with $\boldsymbol{y} \subseteq \boldsymbol{z}$ or an equality $(z = z')$ with $z, z' \in \boldsymbol{z}$. (As usual, when writing clauses, we omit $\forall \boldsymbol{z}$.) The atom γ_0 is the *head* of the clause, and $\gamma_1, \dots, \gamma_m$ its *body*. All variables in the head must also occur in the body, and $=$ can only occur in the body. The predicates in the heads of clauses in Π are *IDB predicates*, the rest (including $=$) *EDB predicates*. A predicate S *depends* on S' in Π if Π has a clause with S in the head and S' in the body; Π is a *nonrecursive datalog* (NDL) *program* if the (directed) *dependence graph* of the dependence relation is acyclic.

An *NDL query* is a pair $(\Pi, G(\boldsymbol{x}))$, where Π is an NDL program and $G(\boldsymbol{x})$ a predicate. A tuple $\boldsymbol{a}$ in $\mathsf{ind}(\mathcal{A})$ is an *answer to* $(\Pi, G(\boldsymbol{x}))$ *over* an ABox $\mathcal{A}$ if $G(\boldsymbol{a})$ holds in the first-order model with domain $\mathsf{ind}(\mathcal{A})$ obtained by closing $\mathcal{A}$ under the clauses in Π; in this case we write $\Pi, \mathcal{A} \models G(\boldsymbol{a})$. The problem of checking whether $\boldsymbol{a}$ is an answer to $(\Pi, G(\boldsymbol{x}))$ over $\mathcal{A}$ is called the *query evaluation problem*. The *arity of* Π is the maximal arity, $\mathsf{r}(\Pi)$, of predicates in Π. The *depth* of $(\Pi, G(\boldsymbol{x}))$ is the length, $\mathsf{d}(\Pi, G)$, of the longest directed path in the dependence graph for Π starting from G. NDL queries are *equivalent* if they have exactly the same answers over any ABox.

An NDL query $(\Pi, G(\boldsymbol{x}))$ is an *NDL-rewriting of an OMQ* $\boldsymbol{Q}(\boldsymbol{x}) = (\mathcal{T}, \boldsymbol{q}(\boldsymbol{x}))$ *over H-complete ABoxes* in case $\mathcal{T}, \mathcal{A} \models \boldsymbol{q}(\boldsymbol{a})$ iff $\Pi, \mathcal{A} \models G(\boldsymbol{a})$, for any H-complete ABox $\mathcal{A}$ and any tuple $\boldsymbol{a}$ in $\mathsf{ind}(\mathcal{A})$. Rewritings *over arbitrary ABoxes* are defined by dropping the condition that the ABoxes are H-complete. Let $(\Pi, G(\boldsymbol{x}))$ be an NDL-rewriting of $\boldsymbol{Q}(\boldsymbol{x})$ over H-complete ABoxes. Denote by Π^* the result of replacing each predicate S in Π with a fresh predicate S^* and adding the clauses $A^*(x) \leftarrow B'(x)$, for $B \sqsubseteq_{\mathcal{T}} A$, and $P^*(x,y) \leftarrow R'(x,y)$, for $R \sqsubseteq_{\mathcal{T}} P$, where $B'(x)$ and $R'(x,y)$ are the obvious first-order translations of B and R (for example, $B'(x) = \exists y\, R(x,y)$ if $B = \exists R$). It is easy to see that $(\Pi^*, G^*(\boldsymbol{x}))$ is an NDL-rewriting of $\boldsymbol{Q}(\boldsymbol{x})$ over arbitrary ABoxes.

It is well-known [4] that, without loss of generality, we can only consider NDL-rewritings of OMQs $(\mathcal{T}, \boldsymbol{q}(\boldsymbol{x}))$ over ABoxes $\mathcal{A}$ that are *consistent* with $\mathcal{T}$.

We call an NDL query $(\Pi, G(x_1, \dots, x_n))$ *ordered* if each of its IDB predicates S comes with fixed variables $x_{i_1}, \dots, x_{i_k}$ $(1 \leq i_1 < \dots < i_k \leq n)$, called the *parameters of* S, such that (*i*) every occurrence of S in Π is of the form $S(y_1, \dots, y_m, x_{i_1}, \dots, x_{i_k})$, (*ii*) the x_i are the parameters of G, and (*iii*) if $\boldsymbol{x}'$ are all the parameters in the body of a clause, then the head has $\boldsymbol{x}'$ among its parameters. The *width* $\mathsf{w}(\Pi, G)$ of a ordered (Π, G) is the maximal number of non-parameter variables in a clause of Π. All our NDL-rewritings in Secs. 4–6 are ordered, so we now only consider ordered NDL queries.

3 NL and LOGCFL Fragments of Nonrecursive Datalog

In this section, we identify two classes of (ordered) NDL queries with the evaluation problem in the complexity classes NL and LOGCFL for combined complexity. Recall [1] that an NDL program is called *linear* if the body of its every clause contains at most one IDB predicate (remember that equality is an EDB predicate).

Theorem 1. *Fix some* $\mathsf{w} > 0$*. The combined complexity of evaluating linear NDL queries of width at most* w *is* NL*-complete.*

Proof. Let $(\Pi, G(\boldsymbol{x}))$ be a linear NDL query. Deciding whether $\Pi, \mathcal{A} \models G(\boldsymbol{a})$ is reducible to finding a path to $G(\boldsymbol{a})$ from a certain set X in the *grounding graph* $\mathfrak{G}(\Pi, \mathcal{A}, \boldsymbol{a})$ constructed as follows. The vertices of the graph are the ground atoms obtained by taking an IDB atom from Π, replacing each of its parameters by the corresponding constant from $\boldsymbol{a}$, and replacing each non-parameter variable by some constant from $\mathcal{A}$. The graph has an edge from $S(\boldsymbol{c})$ to $S'(\boldsymbol{c}')$ iff the grounding of Π contains a clause $S'(\boldsymbol{c}') \leftarrow S(\boldsymbol{c}) \land E_1(\boldsymbol{e}_1) \land \dots \land E_k(\boldsymbol{e}_k)$ with $E_j(\boldsymbol{e}_j) \in \mathcal{A}$, for $1 \leq j \leq k$ (we assume that $(c = c) \in \mathcal{A}$). The set X consists of all vertices $S(\boldsymbol{c})$ with IDB predicates S being of in-degree 0 in the dependency graph of Π for which there is a clause $S(\boldsymbol{c}) \leftarrow E_1(\boldsymbol{e}_1) \land \dots \land E_k(\boldsymbol{e}_k)$ in the grounding of Π with $E_j(\boldsymbol{e}_j) \in \mathcal{A}$ $(1 \leq j \leq k)$. Bounding the width of (Π, G) ensures that $\mathfrak{G}(\Pi, \mathcal{A}, \boldsymbol{a})$ is of polynomial size and can be constructed by a deterministic Turing machine with separate input, write-once output and logarithmic-size working tapes. ❑

The transformation of NDL-rewritings over H-complete ABoxes into rewritings for arbitrary ABoxes in Section 2 does not preserve linearity. However, we can still show that it suffices to consider the H-complete case:

Lemma 2. *For any fixed* $\mathsf{w} > 0$*, there is an* L^{NL}*-transducer that, given a linear NDL-rewriting of an OMQ* $\boldsymbol{Q}(\boldsymbol{x})$ *over H-complete ABoxes that is of width at most* w*, computes a linear NDL-rewriting of* $\boldsymbol{Q}(\boldsymbol{x})$ *over arbitrary ABoxes whose width is at most* $\mathsf{w} + 1$.

The complexity class LOGCFL can be defined in terms of *nondeterministic auxiliary pushdown automata* (NAuxPDAs) [7], which are nondeterministic Turing machines with an additional work tape constrained to operate as a pushdown store. Sudborough [19] proved that LOGCFL coincides with the class of problems that are solved by NAuxPDAs running in logarithmic space and polynomial time (the space on the pushdown tape is not subject to the logarithmic bound).

We call an NDL query (Π, G) *skinny* if the body of any clause in Π has ≤ 2 atoms.

Lemma 3. *For any skinny NDL query* $(\Pi, G(\boldsymbol{x}))$ *and ABox* $\mathcal{A}$*, query evaluation can be done by an NAuxPDA in space* $\log |\Pi| + \mathsf{w}(\Pi, G) \cdot \log |\mathcal{A}|$ *and time* $2^{O(\mathsf{d}(\Pi, G))}$.

Proof. Let $\Pi_{\mathcal{A}}^{\boldsymbol{a}}$ be the set of ground clauses obtained by first replacing each parameter in Π by the corresponding constant from $\boldsymbol{a}$, and then performing the standard grounding of Π using the constants from $\mathcal{A}$. Consider the monotone Boolean circuit $\boldsymbol{C}(\Pi, \mathcal{A}, \boldsymbol{a})$ constructed as follows. The output of $\boldsymbol{C}(\Pi, \mathcal{A}, \boldsymbol{a})$ is $G(\boldsymbol{a})$. For every atom γ occurring in the head of a clause in $\Pi_{\mathcal{A}}^{\boldsymbol{a}}$, we take an OR-gate whose output is γ and inputs are the bodies of the clauses with head γ; for every such body, we take an AND-gate whose inputs are the atoms in the body. We set an input gate γ to 1 iff $\gamma \in \mathcal{A}$. Clearly, $\boldsymbol{C}(\Pi, \mathcal{A}, \boldsymbol{a})$ is a semi-unbounded fan-in circuit (where OR-gates have arbitrarily many inputs, and AND-gates two inputs) with $O(|\Pi| \cdot |\mathcal{A}|^{\mathsf{w}(\Pi,G)})$ gates and depth $O(\mathsf{d}(\Pi, G))$. It is known that the nonuniform analog of LOGCFL can be defined using families of semi-unbounded fan-in circuits of polynomial size and logarithmic depth. Moreover, there is an algorithm that, given such a circuit $\boldsymbol{C}$, computes the output using an NAuxPDA in logarithmic space in the size of $\boldsymbol{C}$ and exponential time in the depth of $\boldsymbol{C}$ [20, pp. 392–397]. Observing that $\boldsymbol{C}(\Pi, \mathcal{A}, \boldsymbol{a})$ can be computed by a deterministic logspace Turing machine, we conclude that the query evaluation problem can be solved by an NAuxPDA in space $\log |\Pi| + \mathsf{w}(\Pi, G) \cdot \log |\mathcal{A}|$ and time $2^{O(\mathsf{d}(\Pi,G))}$. ❑

A function ν from the predicate names in Π to $\mathbb{N}$ is a *weight function for* an NDL-query $(\Pi, G(\boldsymbol{x}))$ if $\nu(P) > 0$, for any IDB P in Π, and $\nu(P) \geq \nu(Q_1) + \cdots + \nu(Q_n)$, for any $P(\boldsymbol{z}) \leftarrow Q_1(\boldsymbol{z}_1) \land \cdots \land Q_n(\boldsymbol{z}_n)$ in Π.

Lemma 4. *If $(\Pi, G(\boldsymbol{x}))$ has a weight function ν, then it is equivalent to a skinny NDL query $(\Pi', G(\boldsymbol{x}))$ such that $|\Pi'|$ is polynomial in $|\Pi|$, $\mathsf{d}(\Pi', G) \leq \mathsf{d}(\Pi, G) + \log \nu(G)$ and $\mathsf{w}(\Pi', G) \leq \mathsf{w}(\Pi, G)$.*

Proof. The proof is by induction on $\mathsf{d}(\Pi, G)$. If $\mathsf{d}(\Pi, G) = 0$, we take $\Pi' = \Pi$. Suppose Π contains a clause ψ of the form $G(\boldsymbol{z}) \leftarrow P_1(\boldsymbol{z}_1) \land \cdots \land P_k(\boldsymbol{z}_k)$ and, for each $1 \leq j \leq k$, we have an NDL query (Π'_{P_j}, P_j) which is equivalent to (Π, P_j) and such that

$$\mathsf{d}(\Pi'_{P_j}, P_j) \;\leq\; \mathsf{d}(\Pi_{P_j}, P_j) + \log \nu(P_j) \;\leq\; \mathsf{d}(\Pi, G) - 1 + \log \nu(P_j). \qquad (1)$$

We construct the Huffman tree [11] for the alphabet $\{1, \dots, k\}$, where the frequency of j is $\nu(P_j)/\nu(G)$ (by definition, $\nu(G) > 0$). The Huffman tree is binary and has k leaves, denoted $1, \dots, k$, and $k - 1$ internal nodes (including the root, g), and the length of the path from g to any leaf j at most $\lceil \log(\nu(G)/\nu(P_j)) \rceil$. For each internal node v of the tree (but the root), we take a predicate $P_v(\boldsymbol{z}_v)$, where $\boldsymbol{z}_v$ is the union of $\boldsymbol{z}_u$ for all descendants u of v; for the root g, we take $P_g(\boldsymbol{z}_g) = G(\boldsymbol{z})$. Let Π'_ψ be the extension of the union of Π'_{P_j}, for $1 \leq j \leq k$, with clauses $P_v(\boldsymbol{z}_v) \leftarrow P_{u_1}(\boldsymbol{z}_{u_1}) \land P_{u_2}(\boldsymbol{z}_{u_2})$, for each v with immediate successors u_1 and u_2. The number of the new clauses is $k - 1$. Consider the NDL query $(\Pi'_\psi, G(\boldsymbol{z}))$. By (1), we have:

$$\begin{aligned} \mathsf{d}(\Pi'_\psi, G) \leq \max\nolimits_j \{ \lceil \log(\nu(G)/\nu(P_j)) \rceil + \mathsf{d}(\Pi'_{P_j}, P_j) \} \leq \\ \max\nolimits_j \{ \log(\nu(G)/\nu(P_j)) + \mathsf{d}(\Pi, G) + \log \nu(P_j) \} = \log \nu(G) + \mathsf{d}(\Pi, G). \end{aligned}$$

Let Π' be the result of applying this transformation to each clause in Π with head $G(\boldsymbol{z})$. It is readily seen that (Π', G) is as required; in particular, $|\Pi'| = O(|\Pi|^2)$. ❑

Theorem 5. *Fix $c \geq 1$, $\mathsf{w} \geq 1$ and a polynomial p. Query evaluation for NDL queries $(\Pi, G(\boldsymbol{x}))$ with a weight function ν such that $\nu(G) \leq p(|\Pi|)$, $\mathsf{w}(\Pi, G) \leq \mathsf{w}$ and $\mathsf{d}(\Pi, G) \leq c \log \nu(G)$ is in LOGCFL for combined complexity.*

Proof. By Lemma 4, (Π, G) is equivalent to a skinny NDL query (Π', G') with $|\Pi'|$ polynomial in $|\Pi|$, $\mathsf{w}(\Pi', G) \leq \mathsf{w}$, and $\mathsf{d}(\Pi', G') \leq (c+1)\log \nu(G)$. By Lemma 3, query evaluation for (Π', G') over $\mathcal{A}$ is solved by an NAuxPDA in space $\log |\Pi'| + \mathsf{w}(\Pi', G) \cdot \log |\mathcal{A}| = O(\log |\Pi| + \log |\mathcal{A}|)$ and time $2^{O(\mathsf{d}(\Pi', G'))} \leq 2^{O(\log \nu(G))} = (\nu(G))^{O(1)} \leq p'(|\Pi|)$, for some polynomial p'. ❑

Corollary 6. *Suppose there is an algorithm that, given any OMQ $\boldsymbol{Q}(\boldsymbol{x})$ from some class $\mathcal{C}$, constructs its NDL-rewriting $(\Pi, G(\boldsymbol{x}))$ over H-complete ABoxes having a weight function ν with $\nu(G) \leq |\boldsymbol{Q}|$ and $\mathsf{d}(\Pi, G) \leq c \log \nu(G)$, and such that $\mathsf{w}(\Pi, G) \leq \mathsf{w}$ and $|\boldsymbol{Q}| \leq |\Pi| \leq p(|\boldsymbol{Q}|)$, for some fixed constants c, w and polynomial p. Then the evaluation problem for the NDL-rewritings $(\Pi^*, G^*(\boldsymbol{x}))$ of the OMQs in $\mathcal{C}$ over arbitrary ABoxes* (*defined in Section 2*) *is in* LOGCFL *for combined complexity.*

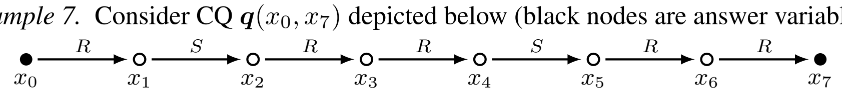
4 Bounded Treewidth CQs and Bounded-Depth TBoxes

With every CQ $\boldsymbol{q}$, we associate its *Gaifman graph* $\mathcal{G}$ whose vertices are the variables of $\boldsymbol{q}$ and edges are the pairs $\{u, v\}$ such that $P(u, v) \in \boldsymbol{q}$, for some P. We call $\boldsymbol{q}$ *tree-shaped* if $\mathcal{G}$ is a tree; $\boldsymbol{q}$ is *connected* if the graph $\mathcal{G}$ is connected. A *tree decomposition* of an undirected graph $\mathcal{G} = (V, E)$ is a pair (T, λ), where T is an (undirected) tree and λ a function from the set of nodes of T to 2^V such that the following conditions hold:

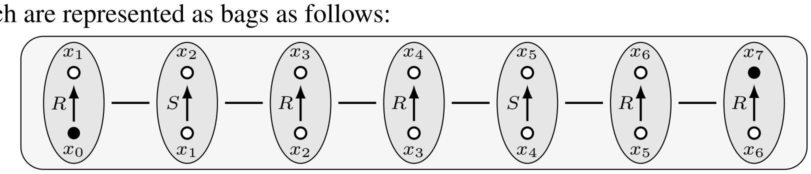
– for every $v \in V$, there exists a node t with $v \in \lambda(t)$;
– for every $e \in E$, there exists a node t with $e \subseteq \lambda(t)$;
– for every $v \in V$, the nodes $\{t \mid v \in \lambda(t)\}$ induce a connected subtree of T.

We call the set $\lambda(t) \subseteq V$ a *bag for* t. The *width* of (T, λ) is $\max_{t \in T} |\lambda(t)| - 1$. The *treewidth of a graph* $\mathcal{G}$ is the minimum width over all tree decompositions of $\mathcal{G}$. The *treewidth of a CQ* is the treewidth of its Gaifman graph.

Example 7. Consider CQ $\boldsymbol{q}(x_0, x_7)$ depicted below (black nodes are answer variables):



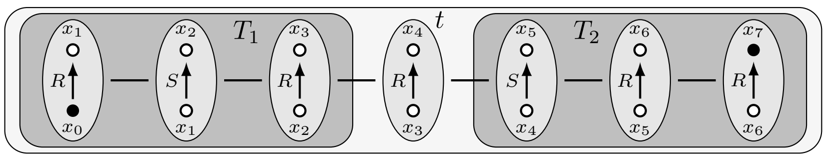
Its natural tree decomposition of treewidth 1 is based on the the chain T of 7 vertices, which are represented as bags as follows:



Fix a connected CQ $\boldsymbol{q}(\boldsymbol{x})$ and a tree decomposition (T, λ) of its Gaifman graph $\mathcal{G} = (V, E)$. Let D be a subtree of T. The *size* of D is the number of nodes in it. We call a node t of D *boundary* if T has an edge $\{t, t'\}$ with $t' \notin D$, and let the *degree* $\mathsf{deg}(D)$ of D be the number of its boundary nodes. Note that T itself is the only subtree of T of degree 0. We say that a node t *splits* D into subtrees $D_1, \dots, D_k$ if the D_i partition D without t: each node of D except t belongs to exactly one D_i.

Lemma 8 **([3]).** *Let D be a subtree of T of size $m > 1$.*
If $\mathsf{deg}(D) = 2$, then there is a node t splitting D into subtrees of size $\leq m/2$ and degree ≤ 2 and, possibly, one subtree of size $< m - 1$ and degree 1.
If $\mathsf{deg}(D) \leq 1$, then there is t splitting D into subtrees of size $\leq m/2$ and degree ≤ 2.

In Example 7, t splits T into T_1 and T_2 depicted below:



We define recursively a set $\mathsf{sub}(T)$ of subtrees of T, a binary relation $\prec$ on $\mathsf{sub}(T)$ and a function σ on $\mathsf{sub}(T)$ indicating the splitting node. We begin by adding T to $\mathsf{sub}(T)$. Take $D \in \mathsf{sub}(T)$ that has not been split yet. If D is of size 1 then let $\sigma(D)$ be the only node of D. Otherwise, by Lemma 8, we find a node t in D that splits it into $D_1, \dots, D_k$. We set $\sigma(D) = t$ and, for each $1 \leq i \leq k$, add D_i to $\mathsf{sub}(T)$ and set $D_i \prec D$; then, we apply the procedure recursively to each of $D_1, \dots, D_k$. In Example 7 with t splitting T, we have $\sigma(T) = t$, $T_1 \prec T$ and $T_2 \prec T$.

For each $D \in \mathsf{sub}(T)$, we recursively define a set of atoms $\boldsymbol{q}_D$ by taking

$$\boldsymbol{q}_D \quad = \quad \big\{S(\boldsymbol{v}) \in \boldsymbol{q} \mid \boldsymbol{v} \subseteq \lambda(\sigma(D))\big\} \ \cup \ \bigcup\nolimits_{D' \prec D} \boldsymbol{q}_{D'}.$$

By the definition of tree decomposition, $\boldsymbol{q}_T = \boldsymbol{q}$. Denote by $\boldsymbol{x}_D$ the subset of $\mathsf{avar}(\boldsymbol{q})$ that occur in $\boldsymbol{q}_D$. In our running example, $\boldsymbol{x}_T = \{x_0, x_7\}$, $\boldsymbol{x}_{T_1} = \{x_0\}$ and $\boldsymbol{x}_{T_2} = \{x_7\}$. Denote by ∂D the union of all $\lambda(t) \cap \lambda(t')$ for a boundary node t of D and its unique neighbour t' in T *outside* D. In our example, $\partial T = \emptyset$, $\partial T_1 = \{x_3\}$ and $\partial T_2 = \{x_4\}$.

Let $\mathcal{T}$ be a TBox of finite depth k. A *type* is a partial map $\boldsymbol{w}$ from V to $\mathbf{W}_\mathcal{T}$; its domain is denoted by $\mathsf{dom}(\boldsymbol{w})$. By $\boldsymbol{\varepsilon}$ we denote the unique partial type with $\mathsf{dom}(\boldsymbol{\varepsilon}) = \emptyset$. We use types to represent how variables are mapped into $\mathcal{C}_{\mathcal{T},\mathcal{A}}$, with $\boldsymbol{w}(u) = w$ indicating that u is mapped to an element of the form aw (for some $a \in \mathsf{ind}(\mathcal{A})$), and with $\boldsymbol{w}(u) = \varepsilon$ that u is mapped to an ABox individual. We say that a type $\boldsymbol{w}$ is *compatible* with a bag t if, for all $u, v \in \lambda(t) \cap \mathsf{dom}(\boldsymbol{w})$, we have

- if $v \in \mathsf{avar}(\boldsymbol{q})$, then $\boldsymbol{w}(v) = \varepsilon$;
- if $A(v) \in \boldsymbol{q}$, then either $\boldsymbol{w}(v) = \varepsilon$ or $\boldsymbol{w}(v) = wR$ with $\exists R^- \sqsubseteq_\mathcal{T} A$;
- if $R(v, u) \in \boldsymbol{q}$, then $\boldsymbol{w}(v) = \boldsymbol{w}(u) = \varepsilon$, or $\boldsymbol{w}(u) = \boldsymbol{w}(v)R'$ with $R' \sqsubseteq_\mathcal{T} R$, or $\boldsymbol{w}(v) = \boldsymbol{w}(u)R'$ with $R' \sqsubseteq_\mathcal{T} R^-$.

In the sequel, we abuse notation and use sets of variables in place of sequences assuming that they are ordered in some (fixed) way. For example, we use $\boldsymbol{x}_D$ for a tuple of variables in the set $\boldsymbol{x}_D$ (ordered in some way). Also, given a tuple $\boldsymbol{a}$ in $\mathsf{ind}(\mathcal{A})$ of length $|\boldsymbol{x}_D|$ and $x \in \boldsymbol{x}_D$, we write $\boldsymbol{a}(x)$ to refer to the element of $\boldsymbol{a}$ that corresponds to x (that is, to the component of the tuple with the same index).

Let $\Pi_{\boldsymbol{Q}}$ be an NDL program that, for any $D \in \mathsf{sub}(T)$, types $\boldsymbol{w}$ and $\boldsymbol{s}$ such that $\mathsf{dom}(\boldsymbol{w}) = \partial D$, $\mathsf{dom}(\boldsymbol{s}) = \lambda(\sigma(D))$, $\boldsymbol{s}$ is compatible with $\sigma(D)$ and agrees with $\boldsymbol{w}$ on their common domain, contains the clause

$$G_D^{\boldsymbol{w}}(\partial D, \boldsymbol{x}_D) \quad \leftarrow \quad \mathsf{At}^{\boldsymbol{s}} \wedge \bigwedge\nolimits_{D' \prec D} G_{D'}^{(\boldsymbol{s} \cup \boldsymbol{w}) \restriction \partial D'}(\partial D', \boldsymbol{x}_{D'}), \qquad (2)$$

where $\boldsymbol{x}_D$ are the parameters of predicate $G_D^{\boldsymbol{w}}$, $(\boldsymbol{s} \cup \boldsymbol{w}) \upharpoonright \partial D'$ is the restriction[1] of the union $\boldsymbol{s} \cup \boldsymbol{w}$ of $\boldsymbol{s}$ and $\boldsymbol{w}$ to $\partial D'$, and $\mathsf{At}^{\boldsymbol{s}}$ is defined as follows:

$$\mathsf{At}^{\boldsymbol{s}} \quad = \bigwedge_{\substack{A(u)\in \boldsymbol{q}\\ \boldsymbol{s}(u)=\varepsilon}} A(u) \;\wedge \bigwedge_{\substack{R(u,v)\in \boldsymbol{q}\\ \boldsymbol{s}(u)=\boldsymbol{s}(v)=\varepsilon}} R(u,v) \;\wedge \bigwedge_{\substack{R(u,v)\in \boldsymbol{q}\\ \boldsymbol{s}(u)\neq\varepsilon \text{ or } \boldsymbol{s}(v)\neq\varepsilon}} (u=v) \;\wedge \bigwedge_{\substack{\boldsymbol{s}(u)=Sw'\\ \text{for some } w'}} A_S(u). \quad (3)$$

The first two conjunctions in $\mathsf{At}^{\boldsymbol{s}}$ ensure that atoms all of whose variables are assigned ε are present in the ABox. The third conjunction ensures that if one of the variables in a role atom is not mapped to ε, then the images of the variables share the same initial individual. Finally, atoms in the final conjunction ensure that if a variable is to be mapped to aSw', then the individual a satisfies $\exists S$ (so aSw' is part of the domain of $\mathcal{C}_{\mathcal{T},\mathcal{A}}$).

Example 9. Now we fix an ontology $\mathcal{T}$ with the following axioms:

$$A \equiv \exists P, \quad P \sqsubseteq S, \quad P \sqsubseteq R^-, \qquad B \equiv \exists Q, \quad Q \sqsubseteq R, \quad Q \sqsubseteq S^-.$$

Since $\lambda(t) = \{x_3, x_4\}$, there are only three types compatible with t:

$$\boldsymbol{s}_1 \colon x_3 \mapsto \varepsilon, x_4 \mapsto \varepsilon, \qquad \boldsymbol{s}_2 \colon x_3 \mapsto P, x_4 \mapsto \varepsilon \quad \text{and} \quad \boldsymbol{s}_3 \colon x_3 \mapsto \varepsilon, x_4 \mapsto Q.$$

So, $\mathsf{At}^{\boldsymbol{s}_1} = R(x_3, x_4)$, $\mathsf{At}^{\boldsymbol{s}_2} = A(x_3) \wedge (x_3 = x_4)$, $\mathsf{At}^{\boldsymbol{s}_3} = B(x_4) \wedge (x_3 = x_4)$. Thus, predicate $G_T^{\boldsymbol{\varepsilon}}$ is defined by the following clauses, for $\boldsymbol{s}_1$, $\boldsymbol{s}_2$ and $\boldsymbol{s}_3$, respectively:

$$\begin{aligned}
&G_T^{\boldsymbol{\varepsilon}}(x_0, x_7) \leftarrow G_{T_1}^{x_3 \mapsto \varepsilon}(x_3, x_0) \wedge R(x_3, x_4) \wedge G_{T_2}^{x_4 \mapsto \varepsilon}(x_4, x_7),\\
&G_T^{\boldsymbol{\varepsilon}}(x_0, x_7) \leftarrow G_{T_1}^{x_3 \mapsto P}(x_3, x_0) \wedge A(x_3) \wedge (x_3 = x_4) \wedge G_{T_2}^{x_4 \mapsto \varepsilon}(x_4, x_7),\\
&G_T^{\boldsymbol{\varepsilon}}(x_0, x_7) \leftarrow G_{T_1}^{x_3 \mapsto \varepsilon}(x_3, x_0) \wedge B(x_4) \wedge (x_3 = x_4) \wedge G_{T_2}^{x_4 \mapsto Q}(x_4, x_7).
\end{aligned}$$

By induction on $\prec$ on $\mathsf{sub}(T)$, we show that $(\Pi_{\boldsymbol{Q}}, G_T^{\boldsymbol{\varepsilon}})$ is a rewriting of $\boldsymbol{Q}(\boldsymbol{x})$.

Lemma 10. *For any ABox $\mathcal{A}$, any $D \in \mathsf{sub}(T)$, any type $\boldsymbol{w}$ with $\mathsf{dom}(\boldsymbol{w}) = \partial D$, any $\boldsymbol{b} \in \mathsf{ind}(\mathcal{A})^{|\partial D|}$ and $\boldsymbol{a} \in \mathsf{ind}(\mathcal{A})^{|\boldsymbol{x}_D|}$, we have $\Pi_{\boldsymbol{Q}}, \mathcal{A} \models G_D^{\boldsymbol{w}}(\boldsymbol{b}, \boldsymbol{a})$ iff there is a homomorphism $h \colon \boldsymbol{q}_D \to \mathcal{C}_{\mathcal{T},\mathcal{A}}$ such that*

$$h(x) = \boldsymbol{a}(x), \text{ for } x \in \boldsymbol{x}_D, \quad \text{and} \quad h(v) = \boldsymbol{b}(v)\boldsymbol{w}(v), \text{ for } v \in \partial D.$$

Fix now k and t, and consider the class of OMQs $\boldsymbol{Q}(\boldsymbol{x}) = (\mathcal{T}, \boldsymbol{q}(\boldsymbol{x}))$ with $\mathcal{T}$ of depth $\leq k$ and $\boldsymbol{q}$ of treewidth $\leq t$. Let T be a tree decomposition of $\boldsymbol{q}$ of treewidth $\leq t$. We take the following weight function: $\nu(G_D^{\boldsymbol{w}}) = |D|$. Clearly, $\nu(G_T^{\boldsymbol{\varepsilon}}) \leq |\boldsymbol{Q}|$. By Lemma 8, $\mathsf{d}(\Pi_{\boldsymbol{Q}}, G_T^{\boldsymbol{\varepsilon}}) \leq 2\log|T| = 2\log\nu(G_T^{\boldsymbol{\varepsilon}}) \leq 2\log|\boldsymbol{Q}|$. Since $|\mathsf{sub}(T)| \leq |T|^2$ and there are at most $|\mathcal{T}|^{2tk}$ options for $\boldsymbol{w}$, there are polynomially many predicates $G_D^{\boldsymbol{w}}$ and so, $\Pi_{\boldsymbol{Q}}$ is of polynomial size. Thus, by Corollary 6, the obtained NDL-rewriting over arbitrary ABoxes can be evaluated in LOGCFL. Finally, we note that a tree decomposition of treewidth $\leq t$ can be computed using an $\mathsf{L}^{\mathsf{LOGCFL}}$-transducer [10], and so the NDL-rewriting can also be constructed by an $\mathsf{L}^{\mathsf{LOGCFL}}$-transducer.

[1] By construction, $\mathsf{dom}(\boldsymbol{s} \cup \boldsymbol{w})$ covers $\partial D'$, and so the domain of $(\boldsymbol{s} \cup \boldsymbol{w}) \upharpoonright \partial D'$ is $\partial D'$.

5 Bounded-Leaf CQs and Bounded-Depth TBoxes

We next consider OMQs with tree-shaped CQs in which both the depth of the ontology and the number of leaves in the CQ are bounded. Let $\mathcal{T}$ be a TBox of finite depth k, and let $\boldsymbol{q}(\boldsymbol{x})$ be a tree-shaped CQ with at most ℓ leaves. Fix one of the variables of $\boldsymbol{q}$ as root, and let M be the maximal distance to a leaf from the root. For $n \leq M$, let $\boldsymbol{z}^n$ denote the set of all variables of $\boldsymbol{q}$ at distance n from the root; clearly, $|\boldsymbol{z}^n| \leq \ell$. We call the $\boldsymbol{z}^n$ *slices* of $\boldsymbol{q}$ and observe that they satisfy the following: for every $R(u,v) \in \boldsymbol{q}$ with $u \neq v$, there exists $0 \leq n < M$ such that either $u \in \boldsymbol{z}^n$ and $v \in \boldsymbol{z}^{n+1}$ or $u \in \boldsymbol{z}^{n+1}$ and $v \in \boldsymbol{z}^n$. For $0 \leq n \leq M$, we denote by $\boldsymbol{q}_n(\boldsymbol{z}_\exists^n, \boldsymbol{x}^n)$ the query consisting of all atoms $S(\boldsymbol{u})$ of $\boldsymbol{q}$ such that $\boldsymbol{u} \subseteq \bigcup_{n \leq m \leq M} \boldsymbol{z}^m$, where $\boldsymbol{x}^n = \mathsf{var}(\boldsymbol{q}_n) \cap \boldsymbol{x}$ and $\boldsymbol{z}_\exists^n = \boldsymbol{z}^n \setminus \boldsymbol{x}$.

By *type of a slice* $\boldsymbol{z}^n$, we mean a total map $\boldsymbol{w}$ from $\boldsymbol{z}^n$ to $\mathbf{W}_\mathcal{T}$. Analogously to Section 4, we define what it means for a type (or pair of types) to be compatible with a slice (pair of adjacent slices). We call $\boldsymbol{w}$ *locally compatible* with $\boldsymbol{z}^n$ if for every $z \in \boldsymbol{z}^n$:

- if $z \in \mathsf{avar}(\boldsymbol{q})$, then $\boldsymbol{w}(z) = \varepsilon$;
- if $A(z) \in \boldsymbol{q}$, then either $\boldsymbol{w}(z) = \varepsilon$ or $\boldsymbol{w}(z) = wR$ with $\exists R^- \sqsubseteq_\mathcal{T} A$;
- if $R(z,z) \in \boldsymbol{q}$, then $\boldsymbol{w}(z) = \varepsilon$.

If $\boldsymbol{w}, \boldsymbol{s}$ are types for $\boldsymbol{z}^n$ and $\boldsymbol{z}^{n+1}$ respectively, then we call $(\boldsymbol{w}, \boldsymbol{s})$ *compatible* with $(\boldsymbol{z}^n, \boldsymbol{z}^{n+1})$ if $\boldsymbol{w}$ is locally compatible with $\boldsymbol{z}^n$, $\boldsymbol{s}$ is locally compatible with $\boldsymbol{z}^{n+1}$, and for every atom $R(z^n, z^{n+1}) \in \boldsymbol{q}$, one of the following holds: (*i*) $\boldsymbol{w}(z^n) = \boldsymbol{s}(z^{n+1}) = \varepsilon$, (*ii*) $\boldsymbol{s}(z^{n+1}) = \boldsymbol{w}(z^n)R'$ with $R' \sqsubseteq_\mathcal{T} R$, or (*iii*) $\boldsymbol{w}(z^n) = \boldsymbol{s}(z^{n+1})R'$ with $R' \sqsubseteq_\mathcal{T} R^-$.

Consider the NDL program $\Pi'_{\boldsymbol{Q}}$ defined as follows. For every $0 \leq n < M$ and every pair of types $(\boldsymbol{w}, \boldsymbol{s})$ that is compatible with $(\boldsymbol{z}^n, \boldsymbol{z}^{n+1})$, we include the clause:

$$P_n^{\boldsymbol{w}}(\boldsymbol{z}_\exists^n, \boldsymbol{x}^n) \leftarrow \mathsf{At}^{\boldsymbol{w} \cup \boldsymbol{s}}(\boldsymbol{z}^n, \boldsymbol{z}^{n+1}) \wedge P_{n+1}^{\boldsymbol{s}}(\boldsymbol{z}_\exists^{n+1}, \boldsymbol{x}^{n+1}),$$

where $\boldsymbol{x}^n$ are the parameters of $P_n^{\boldsymbol{w}}$ and $\mathsf{At}^{\boldsymbol{w} \cup \boldsymbol{s}}(\boldsymbol{z}^n, \boldsymbol{z}^{n+1})$ is the conjunction of atoms (3), as defined in Section 4, for the union $\boldsymbol{w} \cup \boldsymbol{s}$ of types $\boldsymbol{w}$ and $\boldsymbol{s}$. For every type $\boldsymbol{w}$ locally compatible with $\boldsymbol{z}^M$, we include the clause:

$$P_M^{\boldsymbol{w}}(\boldsymbol{z}_\exists^M, \boldsymbol{x}^M) \leftarrow \mathsf{At}^{\boldsymbol{w}}(\boldsymbol{z}^M).$$

(Recall that $\boldsymbol{z}^M$ is a disjoint union of $\boldsymbol{z}_\exists^M$ and $\boldsymbol{x}^M$.) We use G, with parameters $\boldsymbol{x}$, as the goal predicate and include $G(\boldsymbol{x}) \leftarrow P_0^{\boldsymbol{w}}(\boldsymbol{z}^0, \boldsymbol{x})$ for every predicate $P_0^{\boldsymbol{w}}(\boldsymbol{z}^0, \boldsymbol{x}^0)$ occurring in the head of one of the preceding clauses.

The following lemma (which is proved by induction) is the key step in showing that $(\Pi'_{\boldsymbol{Q}}, G(\boldsymbol{x}))$ is a rewriting of $(\mathcal{T}, \boldsymbol{q})$ over H-complete ABoxes:

Lemma 11. *For any H-complete ABox $\mathcal{A}$, any $0 \leq n \leq M$, any predicate $P_n^{\boldsymbol{w}}$, any $\boldsymbol{b} \in \mathsf{ind}(\mathcal{A})^{|\boldsymbol{z}_\exists^n|}$ and any $\boldsymbol{a} \in \mathsf{ind}(\mathcal{A})^{|\boldsymbol{x}^n|}$, we have $\Pi'_{\boldsymbol{Q}}, \mathcal{A} \models P_n^{\boldsymbol{w}}(\boldsymbol{b}, \boldsymbol{a})$ iff there is a homomorphism $h\colon \boldsymbol{q}_n \to \mathcal{C}_{\mathcal{T},\mathcal{A}}$ such that*

$$h(x) = \boldsymbol{a}(x),\ \textit{for } x \in \boldsymbol{x}^n, \quad \textit{and} \quad h(z) = \boldsymbol{b}(z)\boldsymbol{w}(z),\ \textit{for } z \in \boldsymbol{z}_\exists^n. \tag{4}$$

It should be clear that $\Pi'_{\boldsymbol{Q}}$ is a linear NDL program of width at most 2ℓ. Moreover, when ℓ and k are bounded by fixed constants, it takes only logarithmic space to store a type $\boldsymbol{w}$, which allows us to show that $\Pi'_{\boldsymbol{Q}}$ can be computed by an L^{NL}-transducer. We can apply Lemma 2 to obtain an NDL rewriting for arbitrary ABoxes, and then use Theorem 1 to conclude that the resulting program can be evaluated in NL.

6 Bounded-Leaf CQs and Arbitrary TBoxes

For OMQs with bounded-leaf CQs and ontologies of unbounded depth, our rewriting utilises the notion of tree witness [15]. Let $\boldsymbol{Q}(\boldsymbol{x}) = (\mathcal{T}, \boldsymbol{q}(\boldsymbol{x}))$ with $\boldsymbol{q}(\boldsymbol{x}) = \exists \boldsymbol{y}\, \varphi(\boldsymbol{x}, \boldsymbol{y})$. For a pair $\mathsf{t} = (\mathsf{t_r}, \mathsf{t_i})$ of disjoint sets of variables in $\boldsymbol{q}$, with $\mathsf{t_i} \subseteq \boldsymbol{y}$ and $\mathsf{t_i} \neq \emptyset$, set

$$\boldsymbol{q}_{\mathsf{t}} \;=\; \big\{\, S(\boldsymbol{z}) \in \boldsymbol{q} \mid \boldsymbol{z} \subseteq \mathsf{t_r} \cup \mathsf{t_i} \text{ and } \boldsymbol{z} \not\subseteq \mathsf{t_r} \,\big\}.$$

If $\boldsymbol{q}_{\mathsf{t}}$ is a minimal subset of $\boldsymbol{q}$ for which there is a homomorphism $h\colon \boldsymbol{q}_{\mathsf{t}} \to \mathcal{C}_{\mathcal{T}}^{A_R(a)}$ such that $\mathsf{t_r} = h^{-1}(a)$ and $\boldsymbol{q}_{\mathsf{t}}$ contains every atom of $\boldsymbol{q}$ with at least one variable from $\mathsf{t_i}$, then we call $\mathsf{t} = (\mathsf{t_r}, \mathsf{t_i})$ a *tree witness for $\boldsymbol{Q}$ generated by R*. Note that the same tree witness $\mathsf{t} = (\mathsf{t_r}, \mathsf{t_i})$ can be generated by different roles R. By $\boldsymbol{q} \setminus \mathsf{t}$ we denote the query obtained from $\boldsymbol{q}$ by removing the atoms in $\boldsymbol{q}_{\mathsf{t}}$ and having $\boldsymbol{x} \cup \mathsf{t_r}$ as answer variables, and for every $v \in \mathsf{var}(\boldsymbol{q})$, we let $\mathsf{TW}_{\boldsymbol{Q}}(v)$ denote the set of all tree witnesses t for $\boldsymbol{Q}$ such that $v \in \mathsf{t_i}$.

The logarithmic depth NDL-rewriting for bounded-leaf queries and ontologies of unbounded depth is based upon the following observation.

Lemma 12. *Every tree T of size m has a node splitting it into subtrees of size $\leq m/2$.*

We will use repeated applications of this lemma to decompose the input CQ into smaller and smaller subqueries. Formally, for every tree-shaped CQ $\boldsymbol{q}$, we use $v_{\boldsymbol{q}}$ to denote a vertex in the Gaifman graph $\mathcal{G}$ of $\boldsymbol{q}$ that satisfies the condition of Lemma 12. Then, starting from an OMQ $\boldsymbol{Q}_0 = (\mathcal{T}, \boldsymbol{q}_0(\boldsymbol{x}_0))$, we define $\mathsf{SQ}_{\boldsymbol{Q}_0}$ as the smallest set of queries that contains $\boldsymbol{q}_0(\boldsymbol{x}_0)$ and is such that for every $\boldsymbol{q}(\boldsymbol{z}) \in \mathsf{SQ}_{\boldsymbol{Q}_0}$ with $|\mathsf{var}(\boldsymbol{q})| > 1$, the following queries also belong to $\mathsf{SQ}_{\boldsymbol{Q}_0}$:

- for every u_i that is adjacent to $v_{\boldsymbol{q}}$ in $\mathcal{G}$: the query $\boldsymbol{q}_i(\boldsymbol{z}_i)$ consisting of all role atoms linking $v_{\boldsymbol{q}}$ and u_i, as well as all atoms whose variables cannot reach $v_{\boldsymbol{q}}$ in $\mathcal{G}$ without passing by u_i, and with $\boldsymbol{z}_i$ equal to $(\boldsymbol{z} \cap \mathsf{var}(\boldsymbol{q}_i)) \cup \{v_{\boldsymbol{q}}\}$;
- for every $\mathsf{t} \in \mathsf{TW}_{\boldsymbol{Q}_0}(v_{\boldsymbol{q}})$ with $\mathsf{t_r} \neq \emptyset$: the queries $\boldsymbol{q}_1^{\mathsf{t}}(\boldsymbol{z}_1^{\mathsf{t}}), \dots, \boldsymbol{q}_{m_{\mathsf{t}}}^{\mathsf{t}}(\boldsymbol{z}_{m_{\mathsf{t}}}^{\mathsf{t}})$ corresponding to the connected components of $\boldsymbol{q} \setminus \mathsf{t}$, with $\boldsymbol{z}_i^{\mathsf{t}}$ equal to $\mathsf{var}(\boldsymbol{q}_i^{\mathsf{t}}) \cap \mathsf{avar}(\boldsymbol{q} \setminus \mathsf{t})$.

The NDL program $\Pi''_{\boldsymbol{Q}_0}$ uses IDB predicates $P_{\boldsymbol{q}}$, for $\boldsymbol{q} \in \mathsf{SQ}_{\boldsymbol{Q}_0}$, with arity $|\mathsf{avar}(\boldsymbol{q})|$ and parameters $\mathsf{var}(\boldsymbol{q}) \cap \boldsymbol{x}$. For each $\boldsymbol{q}(\boldsymbol{z}) \in \mathsf{SQ}_{\boldsymbol{Q}_0}$ with $|\mathsf{var}(\boldsymbol{q})| > 1$, we include the clause

$$P_{\boldsymbol{q}}(\boldsymbol{z}) \quad\leftarrow\quad \bigwedge_{A(v_{\boldsymbol{q}}) \in \boldsymbol{q}} A(v_{\boldsymbol{q}}) \quad\wedge\quad \bigwedge_{R(v_{\boldsymbol{q}}, v_{\boldsymbol{q}}) \in \boldsymbol{q}} R(v_{\boldsymbol{q}}, v_{\boldsymbol{q}}) \quad\wedge\quad \bigwedge_{1 \leq i \leq n} P_{\boldsymbol{q}_i}(\boldsymbol{z}_i),$$

where $\boldsymbol{q}_1(\boldsymbol{z}_1), \dots, \boldsymbol{q}_n(\boldsymbol{z}_n)$ are the subqueries induced by the neighbours of $v_{\boldsymbol{q}}$ in $\mathcal{G}$. We also include, for every $\mathsf{t} \in \mathsf{TW}_{\boldsymbol{Q}}(v_{\boldsymbol{q}})$ with $\mathsf{t_r} \neq \emptyset$ and role R generating t, the clause

$$P_{\boldsymbol{q}}(\boldsymbol{z}) \quad\leftarrow\quad \bigwedge_{u, u' \in \mathsf{t_r}} (u = u') \quad\wedge\quad \bigwedge_{u \in \mathsf{t_r}} A_R(u) \quad\wedge\quad \bigwedge_{1 \leq i \leq m_{\mathsf{t}}} P_{\boldsymbol{q}_i^{\mathsf{t}}}(\boldsymbol{z}_i^{\mathsf{t}})$$

where $\boldsymbol{q}_1^{\mathsf{t}}, \dots, \boldsymbol{q}_{m_{\mathsf{t}}}^{\mathsf{t}}$ are the connected components of $\boldsymbol{q} \setminus \mathsf{t}$. For every $\boldsymbol{q} \in \mathsf{SQ}_{\boldsymbol{Q}_0}$ with $|\mathsf{var}(\boldsymbol{q})| = 1$, we include the clause $P_{\boldsymbol{q}}(\boldsymbol{z}) \leftarrow \boldsymbol{q}$. Finally, if $\boldsymbol{q}_0$ is Boolean, we include clauses $P_{\boldsymbol{q}_0} \leftarrow A(x)$ for all concept names A such that $\mathcal{T}, \{A(a)\} \models \boldsymbol{q}_0$.

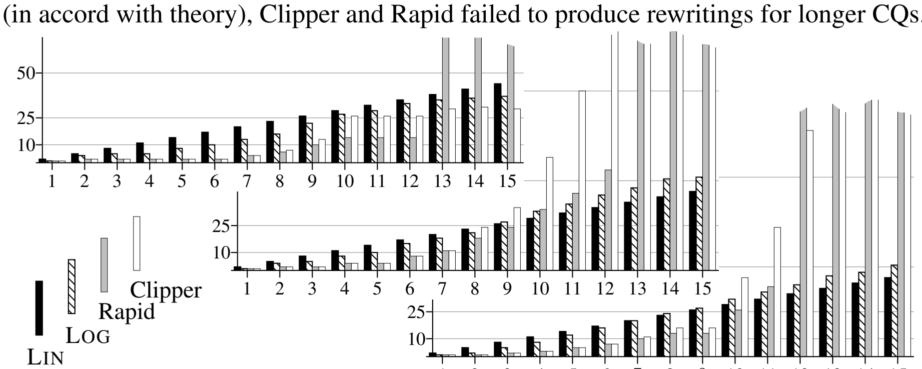
The program $\Pi''_{\boldsymbol{Q}_0}$ is inspired by a similar construction from [12]. By adapting results from the latter paper, we can show that $(\Pi''_{\boldsymbol{Q}_0}, P_{\boldsymbol{q}_0}(\boldsymbol{x}))$ is indeed a rewriting:

Lemma 13. *For any tree-shaped OMQ $\boldsymbol{Q}_0(\boldsymbol{x}) = (\mathcal{T}, \boldsymbol{q}_0(\boldsymbol{x}))$, any $\boldsymbol{q}(\boldsymbol{z}) \in \mathsf{SQ}_{\boldsymbol{Q}_0}$, any H-complete ABox $\mathcal{A}$, and any tuple $\boldsymbol{a}$ in $\mathsf{ind}(\mathcal{A})$, $\Pi''_{\boldsymbol{Q}_0}, \mathcal{A} \models P_{\boldsymbol{q}}(\boldsymbol{a})$ iff there exists a homomorphism $h\colon \boldsymbol{q} \to \mathcal{C}_{\mathcal{T},\mathcal{A}}$ such that $h(\boldsymbol{z}) = \boldsymbol{a}$.*

Now fix $\ell > 1$, and consider the class of OMQs $\boldsymbol{Q}(\boldsymbol{x}) = (\mathcal{T}, \boldsymbol{q}(\boldsymbol{x}))$ with tree-shaped $\boldsymbol{q}(\boldsymbol{x})$ having at most ℓ leaves. The size of $\Pi''_{\boldsymbol{Q}}$ is polynomially bounded in $|\boldsymbol{Q}|$, since bounded-leaf CQs have polynomially many tree witnesses and also polynomially many tree-shaped subCQs. It is readily seen that the function ν defined by setting $\nu(P_{\boldsymbol{q}'}) = |\boldsymbol{q}'|$ is a weight function for $(\Pi''_{\boldsymbol{Q}}, P_{\boldsymbol{q}})$ such that $\nu(P_{\boldsymbol{q}}) \leq |\boldsymbol{Q}|$. Moreover, by Lemma 12, $\mathsf{d}(\Pi, G) \leq \log \nu(P_{\boldsymbol{q}}) + 1$. We can thus apply Corollary 6 to conclude that the obtained NDL-rewritings can be evaluated in LOGCFL. Finally, we note that since the number of leaves is bounded, it is in NL to decide whether a vertex satisfies the conditions of Lemma 12, and it is in LOGCFL to decide whether $\mathcal{T}, \{A(a)\} \models \boldsymbol{q}_0$ [3] or whether a (logspace) representation of a possible tree witness is indeed a tree witness. This allows us to show that $(\Pi''_{\boldsymbol{Q}}, P_{\boldsymbol{q}})$ can be generated by an $\mathsf{L}^{\mathsf{LOGCFL}}$-transducer.

7 Conclusions

As shown above, for three important classes of OMQs, NDL-rewritings can be constructed and evaluated by theoretically optimal NL and LOGCFL algorithms. To see whether these rewritings are viable in practice, we generated three sequences of OMQs with the ontology from Example 9 and linear CQs of up to 15 atoms as in Example 7. We compared our NL and LOGCFL rewritings from Sections 5 and 4 (called LIN and LOG) with those produced by Clipper [8] and Rapid [6]. The barcharts below show the number of clauses in the rewritings over H-complete ABoxes. While LIN and LOG grow linearly (in accord with theory), Clipper and Rapid failed to produce rewritings for longer CQs.



We evaluated the rewritings over a few randomly generated ABoxes using off-the-shelf datalog engine RDFox [17]. The experiments (see the full version) show that our rewritings are usually executed faster than Clipper's and Rapid's when the number of answers is relatively small ($\lesssim 10^4$); for queries with $\gtrsim 10^6$ answers, the execution times are comparable. The version of RDFox we used did not seem to take advantage of the structure of the $\mathsf{NL}/\mathsf{LOGCFL}$ rewritings, and it would be interesting to see whether their nonrecursiveness and parallelisability can be utilised to produce efficient execution plans.

Acknowledgements: The first author was supported by contract ANR-12-JS02-007-01, the fourth by the Russian Foundation for Basic Research and the grant MK-7312.2016.1.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley (1995)
2. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.): The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press (2003)
3. Bienvenu, M., Kikot, S., Podolskii, V.V.: Tree-like queries in OWL 2 QL: succinctness and complexity results. In: Proc. of the 30th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2015). pp. 317–328. ACM (2015)
4. Calì, A., Gottlob, G., Lukasiewicz, T.: A general datalog-based framework for tractable query answering over ontologies. Journal of Web Semantics 14, 57–83 (2012)
5. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: the *DL-Lite* family. Journal of Automated Reasoning 39(3), 385–429 (2007)
6. Chortaras, A., Trivela, D., Stamou, G.: Optimized query rewriting for OWL 2 QL. In: Proc. of CADE-23. LNCS, vol. 6803, pp. 192–206. Springer (2011)
7. Cook, S.A.: Characterizations of pushdown machines in terms of time-bounded computers. Journal of the ACM 18(1), 4–18 (1971)
8. Eiter, T., Ortiz, M., Šimkus, M., Tran, T.K., Xiao, G.: Query rewriting for Horn-SHIQ plus rules. In: Proc. of the 26th AAAI Conf. on Artificial Intelligence (AAAI 2012). pp. 726–733. AAAI (2012)
9. Gottlob, G., Kikot, S., Kontchakov, R., Podolskii, V.V., Schwentick, T., Zakharyaschev, M.: The price of query rewriting in ontology-based data access. Artificial Intelligence 213, 42–59 (2014)
10. Gottlob, G., Leone, N., Scarcello, F.: Computing LOGCFL certificates. In: Proc. of the 26th Int. Colloquium on Automata, Languages and Programming (ICALP-99). LNCS, vol. 1644, pp. 361–371. Springer (1999)
11. Huffman, D.A.: A method for the construction of minimum-redundancy codes. Proc. of the Institute of Radio Engineers 40(9), 1098–1101 (1952)
12. Kikot, S., Kontchakov, R., Podolskii, V., Zakharyaschev, M.: On the succinctness of query rewriting over shallow ontologies. In: Proc. of the 29th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2014). ACM (2014)
13. Kikot, S., Kontchakov, R., Podolskii, V.V., Zakharyaschev, M.: Exponential lower bounds and separation for query rewriting. In: Proc. of the 39th Int. Colloquium on Automata, Languages and Programming (ICALP 2012). LNCS, vol. 7392, pp. 263–274. Springer (2012)
14. Kikot, S., Kontchakov, R., Zakharyaschev, M.: On (in)tractability of OBDA with OWL 2 QL. In: Proc. of the 24th Int. Workshop on Description Logics (DL 2011). vol. 745, pp. 224–234. CEUR-WS (2011)
15. Kikot, S., Kontchakov, R., Zakharyaschev, M.: Conjunctive query answering with OWL 2 QL. In: Proc. of the 13th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2012). pp. 275–285. AAAI (2012)
16. Kontchakov, R., Lutz, C., Toman, D., Wolter, F., Zakharyaschev, M.: The combined approach to query answering in DL-Lite. In: Proc. of the 12th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2010). AAAI Press (2010)
17. Nenov, Y., Piro, R., Motik, B., Horrocks, I., Wu, Z., Banerjee, J.: RDFox: A highly-scalable RDF store. In: Proc. of the 14th Int. Semantic Web Conf. (ISWC 2015), Part II. LNCS, vol. 9367, pp. 3–20. Springer (2015)
18. Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Linking data to ontologies. Journal on Data Semantics X, 133–173 (2008)

19. Sudborough, I.H.: On the tape complexity of deterministic context-free languages. Journal of the ACM 25(3), 405–414 (1978)
20. Venkateswaran, H.: Properties that characterize LOGCFL. Journal of Computer and System Sciences 43(2), 380–404 (1991)