Optimal Nonrecursive Datalog Rewritings of Linear TGDs and Bounded (Hyper)Tree-Width Queries

Meghyn Bienvenu, Stanislav Kikot, Roman Kontchakov, Vladislav Ryzhikov, Michael Zakharyaschev

Optimal Nonrecursive Datalog Rewritings of Linear TGDs and Bounded (Hyper)Tree-Width Queries

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

[1] CNRS & University of Montpellier, France (meghyn@lirmm.fr)
[2] Birkbeck, University of London, UK ({kikot,roman,michael}@dcs.bbk.ac.uk)
[3] Free University of Bozen-Bolzano, Italy (ryzhikov@inf.unibz.it)

Abstract. Our concern is answering ontology-mediated queries $(\mathcal{O}, \boldsymbol{q})$, where $\mathcal{O}$ is a set of linear tgds and $\boldsymbol{q}$ a conjunctive query (CQ) of bounded hypertree width. Assuming that the arity of predicates is bounded, we show that polynomial-size nonrecursive Datalog rewritings can be constructed and executed in (*i*) LOGCFL for OMQs with ontologies of bounded existential depth; (*ii*) NL for OMQs with ontologies of bounded depth and CQs whose hypertree decompositions have a bounded number of leaves; (*iii*) LOGCFL for OMQs with acyclic CQs whose join trees have a bounded number of leaves.

1 Introduction

As shown in [3, 4, 13], the optimal combined complexity (LOGCFL and NL) of answering ontology-mediated queries (OMQs) with *OWL 2 QL* ontologies of bounded depth and conjunctive queries (CQs) of bounded treewidth can be achieved by means of rewriting them into nonrecursive datalog (NDL) queries, though not via positive-existential rewritings. (Note that in these cases the complexity of OMQs matches the complexity of evaluating the underlying CQs.) Our recent experiments have demonstrated that such NDL rewritings, reformulated as Spark SQL queries with views, are efficiently executed by Apache Spark taking advantage of their parallelisable structure.

The aim of this paper is to extend the above mentioned results to ontologies and CQs with predicates of *arbitrary fixed arity*. We consider ontologies that consist of linear TGDs (linear existential rules or atomic-hypothesis rules) [2, 6, 11, 12], which are instances of finite unification sets [18, 15, 16]. Our interest in this problem is also motivated by the system ETAP [5] designed to answer natural language questions by translating them into SPARQL and executing—along with background knowledge—over RDF data extracted from texts. To illustrate, suppose that the data contains the atoms $\mathit{Purchased}(j, c)$ and $\mathit{Car}(c)$ representing the sentence 'John purchased a car'. To answer the question 'has a car been sold?' ETAP utilises the ontology rules (with omitted universal quantifiers)

$$\begin{aligned}
&\mathit{Purchased}(x, y) \to \exists v z \, \big(\mathit{Purchase}(v) \land \mathit{hasAgent1}(v, x) \land \\
&\qquad\qquad \mathit{hasObject}(v, y) \land \mathit{hasAgent2}(v, z)\big), \\
&\mathit{Purchase}(v) \land \mathit{hasAgent1}(v, x) \land \mathit{hasObject}(v, y) \land \mathit{hasAgent2}(v, z) \to \\
&\qquad \exists v' \, \big(\mathit{Sale}(v') \land \mathit{hasAgent1}(v', z) \land \mathit{hasObject}(v', x) \land \mathit{hasAgent2}(v', y)\big),
\end{aligned}$$

where v and v' represent the acts of purchase and sale, respectively. The rules are clearly beyond the limitations of *OWL 2 QL*; however, the knowledge they represent can also be captured by means of linear TGDs with ternary predicates:

$$\mathit{Purchased}(x, y) \to \exists z\, \mathit{Purchase}(x, y, z), \quad \mathit{Purchase}(x, y, z) \to \mathit{Sale}(z, y, x),$$

which are enough to answer the query $\exists xyz\, (\mathit{Car}(y) \land \mathit{Sale}(x, y, z))$.

We classify OMQs $\boldsymbol{Q} = (\mathcal{O}, \boldsymbol{q})$ with linear TGDs and predicates of any *fixed arity* $\boldsymbol{n} < \omega$ along three axes: (1) the existential depth $\boldsymbol{d}$ of $\mathcal{O}$, that is, the maximal depth of Skolem terms in the chases of $\mathcal{O}$ over arbitrary data (cf. [8]), (2) the hypertree width $\boldsymbol{t}$ of $\boldsymbol{q}$, and (3) the number $\boldsymbol{\ell}$ of leaves in the tree underlying a hypertree decomposition of $\boldsymbol{q}$. Thus, $\mathsf{OMQ}(p_1, p_2, p_3)$ denotes the class of OMQs in which parameter (i) is bounded by $p_i \in \mathbb{N} \cup \{\infty\}$. We show that, for any fixed $\boldsymbol{d}, \boldsymbol{t}, \boldsymbol{\ell} < \omega$, answering OMQs in the classes $\mathsf{OMQ}(\boldsymbol{d}, \boldsymbol{t}, \infty)$ and $\mathsf{OMQ}(\infty, 1, \boldsymbol{\ell})$ can be done in LOGCFL (for combined complexity) by means of NDL-rewritings, and even in NL for $\mathsf{OMQ}(\boldsymbol{d}, \boldsymbol{t}, \boldsymbol{\ell})$. On the other hand, one can show that answering OMQs in $\mathsf{OMQ}(\infty, \boldsymbol{t}, \boldsymbol{\ell})$, for $\boldsymbol{t}, \boldsymbol{\ell} \geq 2$, is NP-hard by observing that the sequence of tree-shaped CQs from the proof of [3, Theorem 20] is of path width 2. Thus, we obtain a full classification of the classes $\mathsf{OMQ}(p_1, p_2, p_3)$, for $p_i \in \mathbb{N} \cup \{\infty\}$, with respect to combined complexity.

2 Preliminaries

Ontology-mediated queries. Let Σ be a *relational schema* with the maximum arity $\mathit{ar}(\Sigma)$ of its predicates bounded by $\boldsymbol{n}$. By writing $P(\boldsymbol{x})$, for a predicate name P and an n-tuple $\boldsymbol{x}$ of variables (with possible repetitions), we mean that P is n-ary. By writing $\gamma(\boldsymbol{x})$, we mean that the free variables of formula γ are $\boldsymbol{x}$, where $\boldsymbol{x}$ contains no repetitions. If the meaning is clear from the context, we use set-theoretic notation for lists.

A *data instance*, $\mathcal{D}$, over Σ is any finite set of ground atoms $P(\boldsymbol{a})$ with predicate symbols P from Σ. We denote by $\mathsf{ind}(\mathcal{D})$ the set of individual constants in $\mathcal{D}$. An *ontology* is any finite set, $\mathcal{O}$, of sentences of the form

$$\forall \boldsymbol{x}\, (\gamma_0(\boldsymbol{x}) \to \exists \boldsymbol{y}\, \gamma_1(\boldsymbol{x}', \boldsymbol{y})) \qquad \text{and} \qquad \forall \boldsymbol{x}\, (\gamma_0(\boldsymbol{x}) \to \gamma_2(\boldsymbol{x}')),$$

where γ_0, γ_1 and γ_2 are atoms with predicate symbols from Σ and $\boldsymbol{x}' \subseteq \boldsymbol{x}$, for disjoint sets $\boldsymbol{x}$ and $\boldsymbol{y}$ of variables. When writing rules, we omit the universal quantifiers.

An *ontology-mediated query* (OMQ) $\boldsymbol{Q}(\boldsymbol{x})$ is a pair $(\mathcal{O}, \boldsymbol{q}(\boldsymbol{x}))$, in which $\mathcal{O}$ is an ontology and $\boldsymbol{q}(\boldsymbol{x})$ a *conjunctive query* (*CQ*), that is, a formula of the form $\exists \boldsymbol{y}\, \varphi(\boldsymbol{x}, \boldsymbol{y})$, where φ is a conjunction of atoms $P(\boldsymbol{z})$ over Σ with $\boldsymbol{z} \subseteq \boldsymbol{x} \cup \boldsymbol{y}$. A tuple $\boldsymbol{a} \in \mathsf{ind}(\mathcal{D})^{|\boldsymbol{x}|}$ is a *certain answer* to $\boldsymbol{Q}(\boldsymbol{x})$ over $\mathcal{D}$ if $\mathfrak{M} \models \boldsymbol{q}(\boldsymbol{a})$, for every model $\mathfrak{M}$ of $\mathcal{O} \cup \mathcal{D}$; in this case we write $\mathcal{O}, \mathcal{D} \models \boldsymbol{q}(\boldsymbol{a})$. If the list $\boldsymbol{x}$ of *answer variables* is empty, a *certain answer* to $\boldsymbol{Q}$ over $\mathcal{D}$ is 'yes' if $\mathfrak{M} \models \boldsymbol{q}$, for every model $\mathfrak{M}$ of $\mathcal{O} \cup \mathcal{D}$, and 'no' otherwise. OMQs and CQs without answer variables are called *Boolean*. We often regard CQs as *sets* of their atoms. We abuse notation and use sets of variables in place of sequences assuming that they are ordered in some (fixed) way. Also, given $\boldsymbol{c} \in \mathsf{ind}(\mathcal{D})^{|\boldsymbol{z}|}$ and $z \in \boldsymbol{z}$, we write $\boldsymbol{c}(z)$ to refer to the component of $\boldsymbol{c}$ that corresponds to z.

Canonical models. An important property of tgds is the fact [1] that, for any $\mathcal{O}$ and $\mathcal{D}$, there is a (possibly infinite) *canonical* (or *universal*) *model* $\mathfrak{C}_{\mathcal{O},\mathcal{D}}$ such that, for every

CQ $\boldsymbol{q}(\boldsymbol{x})$ and $\boldsymbol{a} \in \mathsf{ind}(\mathcal{D})^{|\boldsymbol{x}|}$, we have $\mathcal{O}, \mathcal{D} \models \boldsymbol{q}(\boldsymbol{a})$ iff $\mathfrak{C}_{\mathcal{O},\mathcal{D}} \models \boldsymbol{q}(\boldsymbol{a})$. Such a canonical model can be constructed by the following (*oblivious*) *chase procedure* that, intuitively, 'repairs' $\mathcal{D}$ with respect to $\mathcal{O}$ (though not in the most economical way). With each rule ϱ of the form $\gamma_0(\boldsymbol{x}) \to \exists \boldsymbol{y}\, \gamma_1(\boldsymbol{x}', \boldsymbol{y})$, where $\boldsymbol{x} = (x_1, \dots, x_n)$, $\boldsymbol{y} = (y_1, \dots, y_k)$ and $k > 0$, we associate the k-tuple $\boldsymbol{s}_\varrho = (s^1_\varrho, \dots, s^k_\varrho)$ of distinct n-ary *Skolem function* symbols. An *application of ϱ to $\mathcal{D}$* under a map $h\colon \boldsymbol{x} \to \mathsf{ind}(\mathcal{D})$ such that $h(\gamma_0) \in \mathcal{D}$ adds $h'(\gamma_1)$ to $\mathcal{D}$, where h' is defined by taking $h'(x_i) = h(x_i)$, for $1 \leq i \leq n$, and $h'(y_j) = s^j_\varrho(h(\boldsymbol{x}))$, for $1 \leq j \leq k$. An application of a rule $\gamma_0(\boldsymbol{x}) \to \gamma_2(\boldsymbol{x}')$ to $\mathcal{D}$ under such an h adds $h(\gamma_2)$ to $\mathcal{D}$. The *chase algorithm* applies these two rules exhaustively to $\mathcal{O}$ and $\mathcal{D}$ in a breadth-first manner. More precisely, we set $\mathfrak{C}^0_{\mathcal{O},\mathcal{D}} = \mathcal{D}$ and say that the atoms in $\mathfrak{C}^0_{\mathcal{O},\mathcal{D}}$ are of (*derivation*) *level* 0. Assuming that $\mathfrak{C}^{n-1}_{\mathcal{O},\mathcal{D}}$ has already been constructed, we define $\mathfrak{C}^n_{\mathcal{O},\mathcal{D}}$ as follows. Take some enumeration of all distinct pairs (ϱ_i, h_i) such that $\varrho_i \in \mathcal{O}$ with γ_i on the left-hand side is applicable to $\mathfrak{C}^{n-1}_{\mathcal{O},\mathcal{D}}$ under h_i. If none of the atoms in the $h_i(\gamma_i)$ is of level $n-1$, then we set $\mathfrak{C}^n_{\mathcal{O},\mathcal{D}} = \mathfrak{C}^{n-1}_{\mathcal{O},\mathcal{D}}$. Otherwise, we apply the ϱ_i under h_i to $\mathfrak{C}^{n-1}_{\mathcal{O},\mathcal{D}}$ one after the other and say that the newly added atoms are of (*derivation*) *level* n; the resulting extension of $\mathfrak{C}^{n-1}_{\mathcal{O},\mathcal{D}}$ is denoted by $\mathfrak{C}^n_{\mathcal{O},\mathcal{D}}$. The *canonical model* $\mathfrak{C}_{\mathcal{O},\mathcal{D}}$ is then the union of all $\mathfrak{C}^n_{\mathcal{O},\mathcal{D}}$, for $n < \omega$.

The domain $\Delta^{\mathfrak{C}_{\mathcal{O},\mathcal{D}}}$ of $\mathfrak{C}_{\mathcal{O},\mathcal{D}}$ consists of terms built from the constants in $\mathcal{D}$ using Skolem functions s^j_ϱ, for $\varrho \in \mathcal{O}$. The *depth* of such a term is the maximal number of nested occurrences of function symbols in it. We say that $\mathcal{O}$ is of *depth* $k \leq \omega$ if k is the minimal ordinal such that $\Delta^{\mathfrak{C}_{\mathcal{O},\mathcal{D}}}$ contains no terms of depth $> k$, for any data $\mathcal{D}$. For an ontology $\mathcal{O}$ and a ground atom $P(\boldsymbol{a})$, we set $\mathsf{term}_\mathcal{O}(P(\boldsymbol{a})) = \Delta^{\mathfrak{C}_{\mathcal{O},\{P(\boldsymbol{a})\}}} \setminus \boldsymbol{a}$. We denote by $\mathsf{term}_\mathcal{O}$ the union of $\mathsf{term}_\mathcal{O}(P(\boldsymbol{a}))$, for all possible (up to renaming the constants) atoms $P(\boldsymbol{a})$ with predicates in $\mathcal{O}$ (assuming that distinct $P(\boldsymbol{a})$ do not share constants). It should be clear that $\mathcal{O}$ is of finite depth iff $\mathsf{term}_\mathcal{O}$ is finite. By counting the number of possible linear derivations of Skolem terms, we see that, for $\mathcal{O}$ of depth k, $|\,\mathsf{term}_\mathcal{O}\,| \leq (ar(\Sigma)^{ar(\Sigma)}|\mathcal{O}|)^k$. We assume that any constant a occurring in $\mathsf{term}_\mathcal{O}$ has a *twin variable* $\widetilde{a}$ and denote by $\widetilde{\boldsymbol{a}}$ the result of replacing all constants in $\boldsymbol{a}$ with their twin variables. Given a tuple $\boldsymbol{b} \subseteq \mathsf{ind}(\mathcal{D})$, we denote by $\boldsymbol{a}/\boldsymbol{b}(\widetilde{\boldsymbol{a}})$ the substitution that maps each a in $\boldsymbol{a}$ to the corresponding $\boldsymbol{b}(\widetilde{a})$ in $\boldsymbol{b}$.

NDL-rewritings. A *datalog program*, Π, is a finite set of Horn clauses of the form $\forall \boldsymbol{z}\, (\gamma_0 \leftarrow \gamma_1 \wedge \dots \wedge \gamma_m)$, where each γ_i is an atom $Q(\boldsymbol{y})$ with $\boldsymbol{y} \subseteq \boldsymbol{z}$ or an equality $(z = z')$ with $z, z' \in \boldsymbol{z}$. (As usual, we omit $\forall \boldsymbol{z}$ from clauses.) The atom γ_0 is the *head* of the clause, and $\gamma_1, \dots, \gamma_m$ its *body*. All variables in the head must 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 Q *depends* on P in Π if Π has a clause with Q in the head and P in the body. Π is a *nonrecursive datalog* (NDL) *program* if the (directed) *dependence graph* of the dependence relation is acyclic. The size $|\Pi|$ of Π is the number of symbols in it. An *NDL query* is a pair $(\Pi, G(\boldsymbol{x}))$, where Π is an NDL program and G a predicate. A tuple $\boldsymbol{a} \in \mathsf{ind}(\mathcal{D})^{|\boldsymbol{x}|}$ is an *answer to* $(\Pi, G(\boldsymbol{x}))$ *over* a data instance $\mathcal{D}$ if $G(\boldsymbol{a})$ holds in the first-order structure with domain $\mathsf{ind}(\mathcal{D})$ obtained by closing $\mathcal{D}$ under the clauses in Π; in this case we write $\Pi, \mathcal{D} \models G(\boldsymbol{a})$. The problem of checking whether $\boldsymbol{a}$ is an answer to $(\Pi, G(\boldsymbol{x}))$ over $\mathcal{D}$ is called the *query evaluation problem*. 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.

An NDL query $(\Pi, G(\boldsymbol{x}))$ is an *NDL-rewriting of an OMQ* $\boldsymbol{Q}(\boldsymbol{x}) = (\mathcal{O}, \boldsymbol{q}(\boldsymbol{x}))$ in case $\mathcal{O}, \mathcal{D} \models \boldsymbol{q}(\boldsymbol{a})$ iff $\Pi, \mathcal{D} \models G(\boldsymbol{a})$, for any $\mathcal{D}$ and any $\boldsymbol{a} \in \mathsf{ind}(\mathcal{D})^{|\boldsymbol{x}|}$. Every OMQ is known to have an NDL-rewriting [2, 6].

Tree decomposition. A *tree decomposition* of a CQ $\boldsymbol{q}$ with variables $\mathit{var}(\boldsymbol{q})$ is a pair (T, λ) of an (undirected) tree $T = (V, E)$ and $\lambda \colon V \to 2^{\mathit{var}(\boldsymbol{q})}$ such that

- for any atom $P(\boldsymbol{z}) \in \boldsymbol{q}$, there exists $v \in V$ with $\boldsymbol{z} \subseteq \lambda(v)$;
- for any variable z in $\boldsymbol{q}$, the set of vertices $\{v \in V \mid z \in \lambda(v)\}$ is connected in T.

We call $\lambda(v)$ the *bag for* v. The *width* of (T, λ) is $\max_{v \in V} |\lambda(v)| - 1$. The *treewidth of* $\boldsymbol{q}$ is the minimum width over all tree decompositions of $\boldsymbol{q}$. It is known [9] that, for CQs of bounded arity $\boldsymbol{n}$, the notions of bounded treewidth and *bounded hypertree width* [10] are interchangeable. Indeed, in this case, every hypertree decomposition of width $\boldsymbol{t}$ induces a tree decomposition of width $\boldsymbol{t} \cdot \boldsymbol{n}$. A CQ $\boldsymbol{q}$ is called *acyclic* if it has a *join tree* whose nodes are the atoms of $\boldsymbol{q}$ and, whenever atoms γ_1 and γ_2 share a variable, this variable occurs in all atoms along the (unique) path in the tree linking γ_1 and γ_2. It is known [10] that a CQ $\boldsymbol{q}$ is acyclic iff $\boldsymbol{q}$ is of hypertree width 1.

3 NL and LogCFL Fragments of NDL

In this section we present two classes of NDL queries that enjoy NL- and LOGCFL-complete evaluation. First, observe that if the number of variables in each clause of an NDL query is bounded, then the size of its grounding (obtained by replacing variables by all possible combinations of constants) is polynomial. So, evaluation of such NDL queries is tractable. However, if we bound the number of variables in clauses of NDL rewritings of OMQs, then we will also effectively impose a bound on the number of answer variables in their CQs. To avoid this limitation, we treat answer variables of the CQs (and the predicate positions they occur in) differently from all other variables in the NDL-rewritings. Intuitively, answer variables get their values fixed by a candidate certain answer and thus do not cause an exponential blowup of groundings.

Formally, an NDL query $(\Pi, G(\boldsymbol{x}))$ is called *ordered* if each of its IDB predicates Q has a fixed list of variables $\boldsymbol{x}_Q \subseteq \boldsymbol{x}$, the *parameters of* Q, such that

- the parameters of G are $\boldsymbol{x}$ and, in every clause, the parameters of the head include all the parameters of the predicates in the body;
- the parameters $\boldsymbol{x}_Q$ of each Q occupy the last $|\boldsymbol{x}_Q|$ positions in every occurrence of Q in Π; they can, however, occur in other positions too.

The *width* $\mathsf{w}(\Pi, G)$ of an ordered $(\Pi, G(\boldsymbol{x}))$ is the maximum number of non-parameter variables in a clause of Π. Observe that Boolean NDL queries are trivially ordered (their IDB predicates have no parameters), and the width of such queries is simply the maximum number of variables in a clause of Π. As all the NDL-rewritings we construct are ordered, with their parameters being the answer variables, in the sequel we will consider only ordered NDL queries. We say that a class of NDL queries is of *bounded width* if there is $\mathsf{w} > 0$ such that $\mathsf{w}(\Pi, G) \leq \mathsf{w}$, for all $(\Pi, G(\boldsymbol{x}))$ in the class. As we observed above, evaluation of NDL queries of bounded width is P-complete.

Our first subclass of NDL queries is based on linear rules. An NDL program is *linear* [1] if the body of its every clause contains at most one IDB predicate.

Theorem 1. *Evaluation of linear NDL queries of bounded width is* NL*-complete for combined complexity.*

Our second subclass was inspired by semi-unbounded fan-in circuits. Recall that the class LOGCFL of problems reducible in logarithmic space to context-free languages can equivalently be defined in terms of L-uniform families of semi-unbounded fan-in circuits (where OR-gates have arbitrarily many inputs, and AND-gates two inputs) of polynomial size and logarithmic depth. Alternatively, LOGCFL can be defined using *nondeterministic auxiliary pushdown automata* (NAuxPDAs) [7], which are nondeterministic Turing machines with an additional work tape constrained to operate as a pushdown store. Sudborough [17] proved that LOGCFL coincides with the class of problems that are solved by NAuxPDAs in logarithmic space and polynomial time (the space on the pushdown tape is not subject to the logarithmic bound). Moreover, there is an algorithm that, given a semi-unbounded fan-in circuit $\boldsymbol{C}$ and an input, computes the output using an NAuxPDA in logarithmic space in the size of $\boldsymbol{C}$ and exponential time in the depth of $\boldsymbol{C}$ [19, pp. 392–397]. Using these results, it can be shown that any $(\Pi, G(\boldsymbol{x}))$ with *at most two atoms* in the body of any clause can be evaluated on a data instance $\mathcal{D}$ by an NAuxPDA in space $\log|\Pi| + \mathsf{w}(\Pi, G) \cdot \log|\mathcal{D}|$ and time $2^{O(\mathsf{d}(\Pi,G))}$ (thus, in LOGCFL provided the query width is bounded and its depth is logarithmic).

In the rewritings we propose in Sections 5 and 7, however, the number of atoms in the clauses is not bounded. We require the following to generalise the idea. A function ν from the predicate names in Π to non-negative integers $\mathbb{N}$ is called a *weight function* for an NDL query $(\Pi, G(\boldsymbol{x}))$ if, for any clause $Q(\boldsymbol{z}) \leftarrow P_1(\boldsymbol{z}_1) \wedge \dots \wedge P_k(\boldsymbol{z}_k)$ in Π, we have

$$\nu(Q) > 0 \quad \text{and} \quad \nu(Q) \;\geq\; \nu(P_1) + \dots + \nu(P_k),$$

Note that $\nu(P)$ can be 0 for an EDB predicate P. To illustrate, we consider NDL queries with the following dependency graphs:

The one on the left has a weight function bounded by the number of predicates (i.e., linear in the size of the query); intuitively, this function corresponds to the number of directed paths from a vertex to the leaves. In contrast, any NDL query with the dependency graph on the right can only have a weight function whose values (numbers of paths) are exponential. Linear NDL queries have weight functions bounded by 1.

Let e_Π be the maximum number of EDB predicates in a clause of Π. The *skinny depth* $\mathsf{sd}(\Pi, G)$ of $(\Pi, G(\boldsymbol{x}))$ is the minimum value of

$$2\mathsf{d}(\Pi, G) + \log \nu(G) + \log \mathsf{e}_\Pi$$

over possible weight functions ν. One can show, using Huffman coding, that any NDL query $(\Pi, G(\boldsymbol{x}))$ can be transformed into an equivalent skinny NDL query $(\Pi', G(\boldsymbol{x}))$ of depth not exceeding $\mathsf{sd}(\Pi, G)$ and such that $|\Pi'| = O(|\Pi|^2)$ and $\mathsf{w}(\Pi', G) \leq \mathsf{w}(\Pi, G)$. We say that a class of NDL queries has *logarithmic skinny depth* if there is $c > 0$ such that $\mathsf{sd}(\Pi, G) \leq c \log|\Pi|$, for all $(\Pi, G(\boldsymbol{x}))$ in the class. We now obtain:

Theorem 2. *Evaluation of NDL queries of logarithmic skinny depth and bounded width is* LOGCFL*-complete for combined complexity.*

3.1 NDL Rewritings over (Complete) Data

We say that a data instance $\mathcal{D}$ is *complete for an ontology* $\mathcal{O}$ if $\mathcal{O}, \mathcal{D} \models P(\boldsymbol{a})$ implies $P(\boldsymbol{a}) \in \mathcal{D}$, for any ground atom $P(\boldsymbol{a})$, where P in Σ and $\boldsymbol{a} \subseteq \mathsf{ind}(\mathcal{D})$. An NDL query $(\Pi, G(\boldsymbol{x}))$ is an *NDL-rewriting of an OMQ* $\boldsymbol{Q}(\boldsymbol{x}) = (\mathcal{O}, \boldsymbol{q}(\boldsymbol{x}))$ *over complete data* in case $\mathcal{O}, \mathcal{D} \models \boldsymbol{q}(\boldsymbol{a})$ iff $\Pi, \mathcal{D} \models G(\boldsymbol{a})$, for any $\mathcal{D}$ complete for $\mathcal{O}$ and any $\boldsymbol{a} \subseteq \mathsf{ind}(\mathcal{D})$.

Given an NDL-rewriting $(\Pi, G(\boldsymbol{x}))$ of $\boldsymbol{Q}(\boldsymbol{x})$ over complete data, we denote by Π^* the result of replacing each EDB predicate P in Π with a fresh IDB predicate P^* of the same arity and adding the clauses $P^*(\boldsymbol{z}) \leftarrow \gamma$ for every atom γ with a predicate symbol from $\mathcal{O}$ such that $\mathcal{O} \models \gamma \to P(\boldsymbol{z})$, where $\boldsymbol{z}$ is a tuple of variables (with possible repetitions). Clearly, $(\Pi^*, G(\boldsymbol{x}))$ is an NDL-rewriting of $\boldsymbol{Q}(\boldsymbol{x})$ over arbitrary data instances and $|\Pi^*| \leq |\Pi| + ar(\Sigma)^{ar(\Sigma)} \cdot |\mathcal{O}|^2$.

We say that a class of OMQs is *skinny-reducible* if there are $c > 0$ and $\mathsf{w} > 0$ and an $\mathrm{L}^{\mathrm{LOGCFL}}$-transducer that, given any OMQ $\boldsymbol{Q}(\boldsymbol{x})$ in the class, computes its NDL-rewriting $(\Pi, G(\boldsymbol{x}))$ over complete data with $\mathsf{sd}(\Pi, G) \leq c \log |\Pi|$ and $\mathsf{w}(\Pi, G) \leq \mathsf{w}$. Theorem 2 and the transformation * give the following:

Corollary 1. *Answering OMQs is in* LOGCFL *for combined complexity for any skinny-reducible class.*

The transformation *, however, does not preserve linearity because it replaces occurrences of EDB predicates P by IDB predicates P^*. A more involved 'linear' construction is given in the proof of the following, where a possible increase of the width is due to the 'replacement' of atoms $P(\boldsymbol{z})$ by atoms γ whenever $\mathcal{O} \models \gamma \to P(\boldsymbol{z})$:

Lemma 1. *Fix any* $\mathsf{w} > 0$. *There is an* L^{NL}*-transducer that, for a linear NDL-rewriting* $(\Pi, G(\boldsymbol{x}))$ *of an OMQ* $\boldsymbol{Q}(\boldsymbol{x})$ *over complete data with* $\mathsf{w}(\Pi, G) \leq \mathsf{w}$*, computes its linear NDL-rewriting* $(\Pi', G(\boldsymbol{x}))$ *over arbitrary data with* $\mathsf{w}(\Pi', G) \leq \mathsf{w} + ar(\Sigma)$.

4 Conditional Rewritings

Let $\boldsymbol{Q}(\boldsymbol{x}) = (\mathcal{O}, \boldsymbol{q}(\boldsymbol{x}))$ be an OMQ with an ontology of finite depth. Intuitively, we recursively split $\boldsymbol{q}(\boldsymbol{x})$ into subqueries $\boldsymbol{q}_D$ based on subtrees D of a tree decomposition of $\boldsymbol{q}$ and combine the rewritings of $\boldsymbol{q}_D$ into a rewriting of $\boldsymbol{q}$. To guarantee 'compatibility' of the rewritings of the subqueries, we take account of the *types* of points on the boundaries of the $\boldsymbol{q}_D$. So, for each D and each type $\boldsymbol{w}$, we take a fresh IDB predicate $G_D^{\boldsymbol{w}}$ to represent the *conditional rewriting* of $\boldsymbol{q}_D$ provided that its boundary satisfies the type. We now give formal definitions.

A *type* is a partial map $\boldsymbol{s}$ from the variables of $\boldsymbol{q}$ to $\mathsf{term}_{\mathcal{O}} \cup \{\varepsilon\}$; its domain is denoted by $\mathsf{dom}(\boldsymbol{s})$. The unique partial type with $\mathsf{dom}(\boldsymbol{\varepsilon}) = \emptyset$ is denoted by $\boldsymbol{\varepsilon}$. We use types to represent how variables are mapped into the canonical model: $\boldsymbol{s}(z) = \varepsilon$ means that z is mapped to an individual constant and $\boldsymbol{s}(z) = f(\boldsymbol{a})$, for a Skolem term $f(\boldsymbol{a})$, means that z is mapped to an element of the form $f(\boldsymbol{c})$, for some $\boldsymbol{c} \subseteq \mathsf{ind}(\mathcal{D})$. Given a type $\boldsymbol{s}$ and a tuple $\boldsymbol{z} = (z_1, \dots, z_n) \subseteq \mathsf{dom}(\boldsymbol{s})$, we denote the tuple $(\boldsymbol{s}(z_1), \dots, \boldsymbol{s}(z_n))$ by $\boldsymbol{s}(\boldsymbol{z})$. A type $\boldsymbol{s}$ is *compatible* with a bag $\lambda(t)$ if $\boldsymbol{s}(x) = \varepsilon$, for all $x \in \boldsymbol{x} \cap \mathsf{dom}(\boldsymbol{s})$, and, for every $S(\boldsymbol{z}) \in \boldsymbol{q}$ with $\boldsymbol{z} \subseteq \lambda(t) \cap \mathsf{dom}(\boldsymbol{s})$, one of the following applies:

(d) $\boldsymbol{s}(\boldsymbol{z}) \subseteq \{\varepsilon\}$;

(b) there is $P(\boldsymbol{a})$ such that $\boldsymbol{s}(\boldsymbol{z}) \subseteq \mathsf{term}_{\mathcal{O}}(P(\boldsymbol{a})) \cup \{\varepsilon\}$ but neither $\boldsymbol{s}(\boldsymbol{z}) \subseteq \{\varepsilon\}$ nor $\boldsymbol{s}(\boldsymbol{z}) \subseteq \mathsf{term}_{\mathcal{O}}(P(\boldsymbol{a}))$;
(i) there is $P(\boldsymbol{a})$ such that $\boldsymbol{s}(\boldsymbol{z}) \subseteq \mathsf{term}_{\mathcal{O}}(P(\boldsymbol{a}))$ and $S(\boldsymbol{s}(\boldsymbol{z})) \in \mathfrak{C}_{\mathcal{O},\{P(\boldsymbol{a})\}}$.

Given a type $\boldsymbol{s}$, we take a tuple of variables $\mathit{var}(\boldsymbol{s})$ that contains, for $z \in \mathsf{dom}(\boldsymbol{s}) \setminus \boldsymbol{x}$,

$$\text{variable } z, \text{ if } \boldsymbol{s}(z) = \varepsilon, \qquad \text{and} \qquad \text{variables } \widetilde{\boldsymbol{a}}, \text{ if } \boldsymbol{s}(z) \in \mathsf{term}_{\mathcal{O}}(P(\boldsymbol{a})).$$

Denote the answer variables that occur in $\mathsf{dom}(\boldsymbol{s})$ by $\boldsymbol{x_s}$. Our rewritings use conjunctions $\mathsf{At}^{\boldsymbol{s}}(\mathit{var}(\boldsymbol{s}), \boldsymbol{x_s})$ of the following formulas, for all $S(\boldsymbol{z}) \in \boldsymbol{q}$ with $\boldsymbol{z} \subseteq \mathsf{dom}(\boldsymbol{s})$:

(d′) $S(\boldsymbol{z})$ if $\boldsymbol{s}(\boldsymbol{z}) \subseteq \{\varepsilon\}$;
(b′) the disjunction $\displaystyle\bigvee_{g\colon \boldsymbol{z}' \to \boldsymbol{a}} \Big[P(\widetilde{\boldsymbol{a}}) \;\wedge \bigwedge_{z \in \boldsymbol{z}' \text{ and } g(z)=a} (z = \widetilde{a})\Big]$

over *grounding functions* $g\colon \boldsymbol{z}' \to \boldsymbol{a}$ such that $\boldsymbol{z}' = \{z \in \boldsymbol{z} \mid \boldsymbol{s}(z) = \varepsilon\} \neq \emptyset$, $\boldsymbol{z}'' = \{z \in \boldsymbol{z} \mid \boldsymbol{s}(z) \in \mathsf{term}_{\mathcal{O}}(P(\boldsymbol{a}))\} \neq \emptyset$ and $\mathfrak{C}_{\mathcal{O},\{P(\boldsymbol{a})\}}$ contains the result of replacing $\boldsymbol{z}'$ and $\boldsymbol{z}''$ in $S(\boldsymbol{z})$ by $g(\boldsymbol{z}')$ and $\boldsymbol{s}(\boldsymbol{z}'')$, respectively;
(i′) $P(\widetilde{\boldsymbol{a}})$ if $\boldsymbol{s}(\boldsymbol{z}) \subseteq \mathsf{term}_{\mathcal{O}}(P(\boldsymbol{a}))$.

Strictly speaking, the resulting rewritings will not be NDL programs because of disjunctions in **(b′)**, but we can get rid of them using an extra predicate and (if needed) the construction from the proof of Lemma 1 keeping the size and the execution time polynomial.

5 LOGCFL Rewritings for $\mathsf{OMQ}(d, t, \infty)$

We now construct skinny-reducible NDL rewritings for the CQs of bounded treewidth.

Theorem 3. *For any $\boldsymbol{d} \geq 0$ and $\boldsymbol{t} \geq 1$, the class $\mathsf{OMQ}(\boldsymbol{d}, \boldsymbol{t}, \infty)$ is skinny-reducible.*

Fix a connected CQ $\boldsymbol{q}(\boldsymbol{x})$ and a tree decomposition (T, λ) of its Gaifman graph. Let D be a subtree of T. The *size* of D is the number of nodes in it. A node t of D is called *boundary* if T has an edge $\{t, t'\}$ with $t' \notin D$. We denote by ∂D the union of all $\lambda(t) \cap \lambda(t')$ for boundary nodes t of D and its neighbours t' in T *outside* D. The *degree* $\mathsf{deg}(D)$ of D is the number of its boundary nodes (so, the only subtree of T of degree 0 is T itself). 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 2. *Let D be a subtree of T of size $n > 1$.*
If $\mathsf{deg}(D) = 2$, then there is a node t splitting D into subtrees of size $\leq n/2$ and degree ≤ 2 and, possibly, one subtree of size $< n - 1$ and degree 1.
If $\mathsf{deg}(D) \leq 1$, then there is t splitting D into subtrees of size $\leq n/2$ and degree ≤ 2.

We define recursively a set $\mathfrak{R}$ of subtrees of T, a binary 'predecessor' relation $\prec$ on $\mathfrak{R}$, and a function β on $\mathfrak{R}$ indicating the bag of the splitting node. We begin by adding T to $\mathfrak{R}$. Take any $D \in \mathfrak{R}$ that has not been split yet. If D is of size 1, then $\beta(D) = \lambda(t)$ for the only node t of D. Otherwise, by Lemma 2, we find a node t in D

that splits it into $D_1, \dots, D_k$. We set $\beta(D) = \lambda(t)$ and, for $1 \leq i \leq k$, add D_i to $\mathfrak{R}$ and set $D_i \prec D$; then, we apply the procedure to each of $D_1, \dots, D_k$. For each $D \in \mathfrak{R}$, we recursively define a set of atoms

$$\boldsymbol{q}_D \;=\; \big\{S(\boldsymbol{z}) \in \boldsymbol{q} \mid \boldsymbol{z} \subseteq \beta(D)\big\} \cup \bigcup\nolimits_{D' \prec D} \boldsymbol{q}_{D'}.$$

Let $\boldsymbol{x}_D$ be the set of variables from $\boldsymbol{x}$ that occur in $\boldsymbol{q}_D$. By the definition of tree decomposition, $\boldsymbol{q}_T = \boldsymbol{q}$ and $\boldsymbol{x}_T = \boldsymbol{x}$.

We now define an NDL-rewriting of $\boldsymbol{Q}(\boldsymbol{x}) = (\mathcal{O}, \boldsymbol{q}(\boldsymbol{x}))$. Fix $D \in \mathfrak{R}$ and a type $\boldsymbol{w}$ with $\mathsf{dom}(\boldsymbol{w}) = \partial D$. Let $G^{\boldsymbol{w}}_D(\mathit{var}(\boldsymbol{w}), \boldsymbol{x}_D)$ be a fresh IDB predicate with parameters $\boldsymbol{x}_D$. As we described above, a node is selected in D to split it into smaller trees (provided that it contains more than one node). We extend the type $\boldsymbol{w}$ to cover the variables $\beta(D)$ of the selected bag: more precisely, we consider types $\boldsymbol{s}$ with $\mathsf{dom}(\boldsymbol{s}) = \beta(D)$ such that they are compatible with bag $\beta(D)$ and agree with $\boldsymbol{w}$ on their common domain. Observe that, if D' is a subtree resulting from splitting D, then the domain of the extended type, $\boldsymbol{s} \cup \boldsymbol{w}$, includes $\partial D'$, and thus $\partial D'$ coincides with the domain of the restriction of $\boldsymbol{s} \cup \boldsymbol{w}$ to $\partial D'$, denoted $(\boldsymbol{s} \cup \boldsymbol{w}) \restriction_{\partial D'}$. Now, for each type $\boldsymbol{s}$ with $\mathsf{dom}(\boldsymbol{s}) = \beta(D)$ such that $\boldsymbol{s}$ is compatible with bag $\beta(D)$ and agrees with $\boldsymbol{w}$ on their common domain, the NDL program $\Pi^{\mathrm{LOG}}_{\boldsymbol{Q}}$ contains

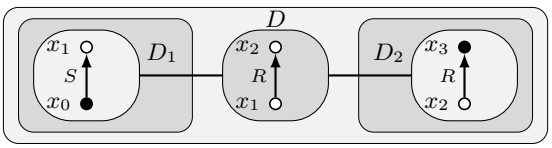
$$G^{\boldsymbol{w}}_D(\mathit{var}(\boldsymbol{w}), \boldsymbol{x}_D) \leftarrow \mathsf{At}^{\boldsymbol{s}}(\mathit{var}(\boldsymbol{s}), \boldsymbol{x}_{\boldsymbol{s}}) \wedge \bigwedge\nolimits_{D' \prec D} G^{(\boldsymbol{s} \cup \boldsymbol{w}) \restriction_{\partial D'}}_{D'}(\mathit{var}((\boldsymbol{s} \cup \boldsymbol{w}) \restriction_{\partial D'}), \boldsymbol{x}_{D'}).$$

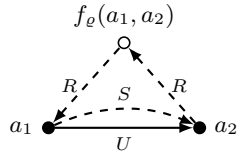
By induction on $\prec$, one can now show that $(\Pi^{\mathrm{LOG}}_{\boldsymbol{Q}}, G^{\boldsymbol{\varepsilon}}_T)$ is a rewriting of $\boldsymbol{Q}(\boldsymbol{x})$.

Example 1. Let $\boldsymbol{q}(x_0, x_3) = \exists x_1 x_2 \big(S(x_0, x_1) \wedge R(x_1, x_2) \wedge R(x_2, x_3)\big)$ and $\mathcal{O}$ consist of the following linear rules:

$$\begin{array}{lll} \varrho\colon & U(x,y) \to \exists v\, T(x,v,y), & T(x,v,y) \to R(v,x), \\ & T(x,v,y) \to R(y,v), & T(x,v,y) \to S(x,y). \end{array}$$

The subtree structure of the tree decomposition of $\boldsymbol{q}(x_0, x_3)$ and the canonical model are as follows:





The goal predicate for the rewriting of $\boldsymbol{q}(x_0, x_3)$ is $G^{\boldsymbol{\varepsilon}}_D(x_0, x_3)$ with parameters x_0 and x_3. For the type $\boldsymbol{s}$ for the middle bag sending x_1 to ε and x_2 to $f_\varrho(a_1, a_2)$, we have

$$G^{\boldsymbol{\varepsilon}}_D(x_0, x_3) \leftarrow G^{x_1 \mapsto \varepsilon}_{D_1}(x_1, x_0) \wedge U(\widetilde{a}_1, \widetilde{a}_2) \wedge (x_1 = \widetilde{a}_2) \wedge G^{x_2 \mapsto f_\varrho(a_1, a_2)}_{D_2}(\widetilde{a}_1, \widetilde{a}_2, x_3),$$

where and $\widetilde{a}_1$ and $\widetilde{a}_2$ are the twin variables in $\mathit{var}(\boldsymbol{s})$. Note that the type for $G^{x_2 \mapsto f_\varrho(a_1, a_2)}_{D_2}$ has no non-twin variables, and we have the following rule for this predicate

$$G^{x_2 \mapsto f_\varrho(a_1, a_2)}_{D_2}(\widetilde{a}_1, \widetilde{a}_2, x_3) \leftarrow U(\widetilde{a}_1, \widetilde{a}_2) \wedge (x_3 = \widetilde{a}_1).$$

Lemma 3. *For any $\mathcal{D}$ complete for $\mathcal{O}$, any predicate $G_D^{\boldsymbol{w}}$ and any $\boldsymbol{b} \in \mathsf{ind}(\mathcal{D})^{|var(\boldsymbol{w})|+|\boldsymbol{x}_D|}$, we have $\Pi_{\boldsymbol{Q}}^{\text{LOG}}, \mathcal{D} \models G_D^{\boldsymbol{w}}(\boldsymbol{b})$ iff there is a homomorphism $h\colon \boldsymbol{q}_D \to \mathfrak{C}_{\mathcal{O},\mathcal{D}}$ such that*

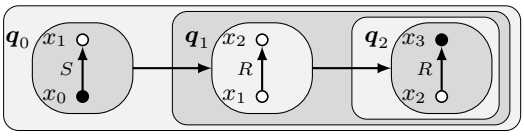
$$h(z) = \begin{cases} \boldsymbol{b}(z), & \text{for all } z \in \boldsymbol{x}_D \text{ and all } z \in \partial D \text{ with } \boldsymbol{w}(z) = \varepsilon, \\ \boldsymbol{w}(z)[\boldsymbol{a}/\boldsymbol{b}(\widetilde{\boldsymbol{a}})], & \text{for all } z \in \partial D \text{ with } \boldsymbol{w}(z) \in \mathsf{term}_{\mathcal{O}}(P(\boldsymbol{a})). \end{cases}$$

6 NL Rewritings for OMQ($\boldsymbol{d}$, $\boldsymbol{t}$, $\boldsymbol{\ell}$)

For OMQs based upon bounded leaf queries and bounded depth ontologies, we establish the following theorem:

Theorem 4. *Let $\boldsymbol{d} \geq 0$, $\boldsymbol{t} \geq 1$ and $\boldsymbol{\ell} \geq 2$ be fixed. There is an L^{NL}-transducer that, given any OMQ in $\mathsf{OMQ}(\boldsymbol{d}, \boldsymbol{t}, \boldsymbol{\ell})$, constructs its polynomial-size linear NDL-rewriting of width $\leq \boldsymbol{\ell}(\boldsymbol{t}+1)$.*

Let $\mathcal{O}$ be an ontology of finite depth $\boldsymbol{d}$ and $\boldsymbol{q}(\boldsymbol{x})$ a CQ with a tree decomposition (T, λ) of width $\leq \boldsymbol{t}$ having $\leq \boldsymbol{\ell}$ leaves. Fix one of the nodes of T as root, and let M be the maximum distance to a leaf from the root. For $0 \leq n \leq M$, by an *n-slice* we mean the set of all nodes of T located at distance n from the root. Denote by $\boldsymbol{y}^n$ the union of all bags $\lambda(t)$ for a node t in the n-slice. For $1 \leq n \leq M$, let $\boldsymbol{z}^n$ be the union of all $\lambda(t) \cap \lambda(t')$ for a node t in the n-slice and its predecessor t' in T (which is in $(n-1)$-slice), and let $\boldsymbol{z}^0 = \emptyset$. By definition, $\boldsymbol{z}^{n+1} \subseteq \boldsymbol{y}^{n+1} \cap \boldsymbol{y}^n$ and, clearly, $|\boldsymbol{z}^n| \leq |\boldsymbol{y}^n| \leq \boldsymbol{\ell}(\boldsymbol{t}+1)$. Denote by $\boldsymbol{q}_n(\boldsymbol{z}_\exists^n, \boldsymbol{x}^{\geq n})$ the query consisting of all atoms $S(\boldsymbol{z})$ of $\boldsymbol{q}$ with $\boldsymbol{z} \subseteq \bigcup_{k \geq n} \boldsymbol{y}^k$, where $\boldsymbol{z}_\exists^n = \boldsymbol{z}^n \setminus \boldsymbol{x}$ and $\boldsymbol{x}^{\geq n} = \boldsymbol{x} \cap \bigcup_{k \geq n} \boldsymbol{y}^k$. These queries and sets of variables for the CQ from Example 1 are shown below:



$\boldsymbol{y}^2 = \{x_2, x_3\}$ $\boldsymbol{z}^2 = \{x_2\}$ $\boldsymbol{x}^2 = \{x_3\}$
$\boldsymbol{y}^1 = \{x_1, x_2\}$ $\boldsymbol{z}^1 = \{x_1\}$ $\boldsymbol{x}^1 = \emptyset$
$\boldsymbol{y}^0 = \{x_0, x_1\}$ $\boldsymbol{z}^0 = \emptyset$ $\boldsymbol{x}^0 = \{x_0\}$

A *type for* $\boldsymbol{z}^n$ is a total map $\boldsymbol{w}$ from $\boldsymbol{z}^n$ to $\mathsf{term}_{\mathcal{O}} \cup \{\varepsilon\}$. Likewise, a *type for* $\boldsymbol{y}^n$ is a total map $\boldsymbol{s}$ from $\boldsymbol{y}^n$ to $\mathsf{term}_{\mathcal{O}} \cup \{\varepsilon\}$. We say $\boldsymbol{s}$ *compatible* with $\boldsymbol{y}^n$ if it is compatible with every bag $\lambda(t)$ in the n-slice.

Consider the NDL program $\Pi_{\boldsymbol{Q}}^{\text{LIN}}$ defined as follows. For every $0 \leq n < M$ and every type $\boldsymbol{w}$ for $\boldsymbol{z}^n$, we introduce a new IDB predicate $G_n^{\boldsymbol{w}}(var(\boldsymbol{w}), \boldsymbol{x}^{\geq n})$ with parameters $\boldsymbol{x}^{\geq n}$. For each type $\boldsymbol{s}$ for $\boldsymbol{y}^n$ such that $\boldsymbol{s}$ is compatible with $\boldsymbol{y}^n$ and agrees with $\boldsymbol{w}$ on $\boldsymbol{z}^n$, the program $\Pi_{\boldsymbol{Q}}^{\text{LIN}}$ contains the clause

$$G_n^{\boldsymbol{w}}(var(\boldsymbol{w}), \boldsymbol{x}^{\geq n}) \leftarrow \mathsf{At}^{\boldsymbol{s}}(var(\boldsymbol{s}), \boldsymbol{x}_{\boldsymbol{s}}) \wedge G_{n+1}^{\boldsymbol{s}\restriction \boldsymbol{z}^{n+1}}(var(\boldsymbol{s}\restriction_{\boldsymbol{z}^{n+1}}), \boldsymbol{x}^{\geq n+1}).$$

For every type $\boldsymbol{w}$ for $\boldsymbol{z}^M$ and every type $\boldsymbol{s}$ for $\boldsymbol{y}^M$ such that $\boldsymbol{s}$ is compatible with $\boldsymbol{y}^M$ and agrees with $\boldsymbol{w}$ on $\boldsymbol{z}^M$, we include the clause

$$G_M^{\boldsymbol{w}}(var(\boldsymbol{w}), \boldsymbol{x}^{\geq M}) \leftarrow \mathsf{At}^{\boldsymbol{s}}(var(\boldsymbol{s}), \boldsymbol{x}_{\boldsymbol{s}}).$$

Finally, we use $G_0^{\boldsymbol{\varepsilon}}$ with parameters $\boldsymbol{x}$ as the goal predicate (note that $\boldsymbol{z}^0 = \emptyset$, and so the domain of any type for $\boldsymbol{z}^0$ is empty).

Lemma 4. *For any $\mathcal{D}$ complete for $\mathcal{O}$, any predicate $G_n^{\boldsymbol{w}}$, any $\boldsymbol{b} \in \mathsf{ind}(\mathcal{D})^{|var(\boldsymbol{w})|+|\boldsymbol{x}^{\geq n}|}$, we have $\mathit{\Pi}_{\boldsymbol{Q}}^{\text{LIN}}, \mathcal{D} \models G_n^{\boldsymbol{w}}(\boldsymbol{b})$ iff there is a homomorphism $h\colon \boldsymbol{q}_n \to \mathfrak{C}_{\mathcal{O},\mathcal{D}}$ such that*

$$h(z) = \begin{cases} \boldsymbol{b}(z), & \text{for all } z \in \boldsymbol{x}^{\geq n} \text{ and all } z \in \boldsymbol{z}_{\exists}^n \text{ with } \boldsymbol{w}(z) = \varepsilon, \\ \boldsymbol{w}(z)[\boldsymbol{a}/\boldsymbol{b}(\widetilde{\boldsymbol{a}})], & \text{for all } z \in \boldsymbol{z}_{\exists}^n \text{ with } \boldsymbol{w}(z) \in \mathsf{term}_{\mathcal{O}}(P(\boldsymbol{a})). \end{cases}$$

It should be clear that $\mathit{\Pi}_{\boldsymbol{Q}}^{\text{LIN}}$ is a linear NDL program of width $\leq \boldsymbol{\ell}(\boldsymbol{t}+1)$ and containing $\leq |\boldsymbol{q}| \cdot |\,\mathsf{term}_{\mathcal{O}}\,|^{\boldsymbol{\ell}(\boldsymbol{t}+1)}$ predicates. Moreover, it takes only logarithmic space to store a type $\boldsymbol{w}$, which allows us to show that $\mathit{\Pi}_{\boldsymbol{Q}}^{\text{LIN}}$ can be computed by an L^{NL}-transducer. We apply Lemma 1 to obtain an NDL-rewriting for arbitrary data instances, and then use Theorem 1 to conclude that the resulting program can be evaluated in NL.

7 LOGCFL Rewritings for $\mathsf{OMQ}(\infty, 1, \ell)$

Unlike the previous two classes, answering OMQs from the class $\mathsf{OMQ}(\infty, 1, \ell)$ can be harder—LOGCFL-complete—than evaluating their CQs, which can be done in NL.

Theorem 5. *For any fixed $\ell \geq 2$, the class $\mathsf{OMQ}(\infty, 1, \ell)$ is skinny-reducible.*

For OMQs with ontologies of unbounded depth and acyclic CQs whose join trees have a bounded number of leaves, our rewriting uses the notion of Skolem witness that generalises tree witnesses [14].

Let $\boldsymbol{Q}(\boldsymbol{x}) = (\mathcal{O}, \boldsymbol{q}(\boldsymbol{x}))$ be an OMQ, let $\mathfrak{s} = (\mathfrak{s}_{\mathsf{r}}^1, \dots, \mathfrak{s}_{\mathsf{r}}^n, \mathfrak{s}_{\mathsf{i}})$ be a tuple of disjoint sets of variables in $\boldsymbol{q}(\boldsymbol{x})$ such that $\mathfrak{s}_{\mathsf{i}} \neq \emptyset$ and $\mathfrak{s}_{\mathsf{i}} \cap \boldsymbol{x} = \emptyset$, and let $\mathfrak{s}_{\mathsf{r}} = \mathfrak{s}_{\mathsf{r}}^1 \cup \dots \cup \mathfrak{s}_{\mathsf{r}}^n$,

$$\boldsymbol{q}_{\mathfrak{s}} \;=\; \big\{\, S(\boldsymbol{z}) \in \boldsymbol{q} \mid \boldsymbol{z} \subseteq \mathfrak{s}_{\mathsf{r}} \cup \mathfrak{s}_{\mathsf{i}} \text{ and } \boldsymbol{z} \not\subseteq \mathfrak{s}_{\mathsf{r}} \,\big\}.$$

If $\boldsymbol{q}_{\mathfrak{s}}$ is a minimal subset of $\boldsymbol{q}$ containing every atom of $\boldsymbol{q}$ with at least one variable from $\mathfrak{s}_{\mathsf{i}}$ and such that there is a homomorphism $h\colon \boldsymbol{q}_{\mathfrak{s}} \to \mathfrak{C}_{\mathcal{O},\{P(\boldsymbol{a})\}}$ with $\boldsymbol{a} = (a_1, \dots, a_n)$ and $h^{-1}(a_j) = \mathfrak{s}_{\mathsf{r}}^j$ for $1 \leq j \leq n$, then we call $\mathfrak{s}$ a *Skolem witness for $\boldsymbol{Q}(\boldsymbol{x})$ generated by $P(\boldsymbol{a})$*. Intuitively, $\mathfrak{s}$ identifies a minimal subset of $\boldsymbol{q}$ that can be mapped to the *Skolem part* of the canonical model $\mathfrak{C}_{\mathcal{O},\{P(\boldsymbol{a})\}}$ consisting of Skolem terms: the variables in $\mathfrak{s}_{\mathsf{r}}$ are mapped to constants from $\boldsymbol{a}$ and the variables in $\mathfrak{s}_{\mathsf{i}}$ to Skolem terms in $\mathsf{term}_{\mathcal{O}}(P(\boldsymbol{a}))$.

The logarithmic-depth NDL-rewriting for $\mathsf{OMQ}(\infty, 1, \ell)$ is based on the following:

Lemma 5. *Every tree T of size n has a node splitting it into subtrees of size $\leq \lceil n/2 \rceil$.*

Let $\boldsymbol{Q}(\boldsymbol{x}_0) = (\mathcal{O}, \boldsymbol{q}_0(\boldsymbol{x}_0))$ be an OMQ with an acyclic CQ having a join tree T_0. We repeatedly apply Lemma 5 to decompose the CQ into smaller and smaller subqueries. Formally, for an acyclic CQ $\boldsymbol{q}$, we denote by $\gamma_{\boldsymbol{q}}$ a vertex in the join tree T for $\boldsymbol{q}$ that satisfies the condition of Lemma 5. Let $\mathfrak{Q}$ be the smallest set containing $\boldsymbol{q}_0(\boldsymbol{x}_0)$ and the following CQs, for every $\boldsymbol{q}(\boldsymbol{x}) \in \mathfrak{Q}$ with at least one existentially quantified variable:

(1) the CQs $\boldsymbol{q}_i(\boldsymbol{x}_i)$ corresponding to the connected components T_i with root $\gamma_{\boldsymbol{q}_i}$ adjacent to $\gamma_{\boldsymbol{q}}$ of the result of removing $\gamma_{\boldsymbol{q}}$ from T, where $\boldsymbol{x}_i$ consists of the restriction of $\boldsymbol{x}$ to the variables in $\boldsymbol{q}_i$ together with the common variables of $\gamma_{\boldsymbol{q}_i}$ and $\gamma_{\boldsymbol{q}}$;

(2) for each Skolem witness $\mathfrak{s}$ for $(\mathcal{O}, \boldsymbol{q}(\boldsymbol{x}))$ with $\mathfrak{s}_{\mathsf{r}} \neq \emptyset$ and $\gamma_{\boldsymbol{q}} \in \boldsymbol{q}_{\mathfrak{s}}$, the CQs $\boldsymbol{q}^{\mathfrak{s}}_1(\boldsymbol{x}^{\mathfrak{s}}_1), \dots, \boldsymbol{q}^{\mathfrak{s}}_k(\boldsymbol{x}^{\mathfrak{s}}_k)$ that correspond to the connected components $T^{\mathfrak{s}}_i$ of the results of removing $\boldsymbol{q}_{\mathfrak{s}}$ from T (note that $\boldsymbol{q}_{\mathfrak{s}}$ is connected in T), where each $\boldsymbol{x}^{\mathfrak{s}}_i$ is the set of variables in $\boldsymbol{x} \cup \mathfrak{s}_{\mathsf{r}}$ that occur in $\boldsymbol{q}^{\mathfrak{s}}_i$.

The NDL program $\Pi^{\mathrm{SW}}_{\boldsymbol{Q}}$ uses IDB predicates $G_{\boldsymbol{q}}(\boldsymbol{x})$, for $\boldsymbol{q}(\boldsymbol{x}) \in \mathfrak{Q}$, whose parameters are the variables in $\boldsymbol{x}_0$ that occur in $\boldsymbol{q}(\boldsymbol{x})$. For each $\boldsymbol{q}(\boldsymbol{x}) \in \mathfrak{Q}$ that has no existentially quantified variables, we include the clause $G_{\boldsymbol{q}}(\boldsymbol{x}) \leftarrow \boldsymbol{q}(\boldsymbol{x})$. For any $\boldsymbol{q}(\boldsymbol{x}) \in \mathfrak{Q}$ with existential variables, we include

$$G_{\boldsymbol{q}}(\boldsymbol{x}) \quad \leftarrow \gamma_{\boldsymbol{q}} \wedge \bigwedge\nolimits_{1 \leq i \leq n} G_{\boldsymbol{q}_i}(\boldsymbol{x}_i),$$

where $\boldsymbol{q}_1(\boldsymbol{x}_1), \dots, \boldsymbol{q}_n(\boldsymbol{x}_n)$ are the subqueries obtained by splitting $\boldsymbol{q}$ by $\gamma_{\boldsymbol{q}}$ in (1), and, for any Skolem witness $\mathfrak{s}$ of $(\mathcal{O}, \boldsymbol{q}(\boldsymbol{x}))$ with $\mathfrak{s}_{\mathsf{r}} \neq \emptyset$ and $\gamma_{\boldsymbol{q}} \in \boldsymbol{q}_{\mathfrak{s}}$ and any $P(\boldsymbol{a})$ generating $\mathfrak{s}$, the clause

$$G_{\boldsymbol{q}}(\boldsymbol{x}) \quad \leftarrow P(\widetilde{\boldsymbol{a}}) \wedge \bigwedge\nolimits_{z \in \mathfrak{s}^j_{\mathsf{r}}} (z = \widetilde{a}_j) \quad \wedge \bigwedge\nolimits_{1 \leq i \leq k} G_{\boldsymbol{q}^{\mathfrak{s}}_i}(\boldsymbol{x}^{\mathfrak{s}}_i),$$

where $\boldsymbol{q}^{\mathfrak{s}}_1, \dots, \boldsymbol{q}^{\mathfrak{s}}_k$ are the connected components of $\boldsymbol{q}$ without $\boldsymbol{q}_{\mathfrak{s}}$. Finally, if $\boldsymbol{q}_0$ is Boolean, then we include $G_{\boldsymbol{q}_0} \leftarrow P(\widetilde{\boldsymbol{a}})$ for all atoms $P(\boldsymbol{a})$ such that $\mathcal{O}, \{P(\boldsymbol{a})\} \models \boldsymbol{q}_0$.

Lemma 6. *For any OMQ with an acyclic CQ, any data $\mathcal{D}$ complete for $\mathcal{O}$, any query $\boldsymbol{q}(\boldsymbol{x}) \in \mathfrak{Q}$ and any $\boldsymbol{b} \in \mathsf{ind}(\mathcal{D})^{|\boldsymbol{x}|}$, we have $\Pi^{\mathrm{SW}}_{\boldsymbol{Q}}, \mathcal{D} \models G_{\boldsymbol{q}}(\boldsymbol{b})$ iff there is a homomorphism $h \colon \boldsymbol{q} \to \mathfrak{C}_{\mathcal{O},\mathcal{D}}$ with $h(\boldsymbol{x}) = \boldsymbol{b}$.*

Now fix $\boldsymbol{\ell} > 1$ and consider $\boldsymbol{Q}(\boldsymbol{x}) = (\mathcal{O}, \boldsymbol{q}_0(\boldsymbol{x}))$ from the class $\mathsf{OMQ}(\infty, 1, \boldsymbol{\ell})$ (remember that we have fixed arity $\boldsymbol{n}$). The size of the program $\Pi^{\mathrm{SW}}_{\boldsymbol{Q}}$ is polynomially bounded in $|\boldsymbol{Q}|$ since $\boldsymbol{q}_0$ has polynomially-many subtrees of $T_{\boldsymbol{q}_0}$ and $O(|\boldsymbol{q}_0|^{\boldsymbol{\ell}})$ Skolem witnesses (there are at most $O(|\boldsymbol{q}_0|^{\boldsymbol{\ell}} \cdot |\Sigma| \cdot \boldsymbol{n}^{\boldsymbol{n}})$ pairs of a Skolem witness $\mathfrak{s}$ and its generating atom $P(\boldsymbol{a})$). It is readily seen that the function ν defined by $\nu(G_{\boldsymbol{q}}) = |\boldsymbol{q}|$, for each $\boldsymbol{q} \in \mathfrak{Q}$, is a weight function for $(\Pi^{\mathrm{SW}}_{\boldsymbol{Q}}, G_{\boldsymbol{q}_0}(\boldsymbol{x}))$ with $\nu(G_{\boldsymbol{q}_0}) \leq |\boldsymbol{Q}|$. Moreover, by Lemma 5, $\mathsf{d}(\Pi^{\mathrm{SW}}_{\boldsymbol{Q}}, G_{\boldsymbol{q}_0}) \leq \log \nu(G_{\boldsymbol{q}_0}) + 1$; also, $\mathsf{w}(\Pi^{\mathrm{SW}}_{\boldsymbol{Q}}, G_{\boldsymbol{q}_0}) \leq \boldsymbol{\ell} + 1$. 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 5, and in LOGCFL to decide whether $\mathcal{O}, \{P(\boldsymbol{a})\} \models \boldsymbol{q}(\boldsymbol{a})$, for bounded-leaf acyclic CQs $\boldsymbol{q}(\boldsymbol{x})$ (see the full version[1]), or whether a (logspace) representation of a possible Skolem witness is indeed a Skolem witness. This allows us to show that $(\Pi^{\mathrm{SW}}_{\boldsymbol{Q}}, G_{\boldsymbol{q}_0}(\boldsymbol{x}))$ can be generated by an $\mathrm{L}^{\mathrm{LOGCFL}}$-transducer. By Corollary 1, the obtained NDL-rewritings can be evaluated in LOGCFL.

8 Conclusion

We presented NDL rewritings for three classes of OMQs with CQs of bounded (hyper)-tree width and ontologies given as linear TGDs. These NDL rewritings can be constructed and evaluated in LOGCFL, NL and LOGCFL, respectively (provided that the arity of predicates is bounded). Since the three upper bounds match the lower bounds inherited from the OWL 2 QL setting [4], the proposed rewritings are theoretically optimal.

[1] `http://www.dcs.bbk.ac.uk/~kikot/DL17-1-full.pdf`

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley (1995)
2. Baget, J.-F., Leclère, M., Mugnier, M.-L., Salvat, E.: On rules with existential variables: Walking the decidability line. Artificial Intelligence 175(9–10), 1620–1654 (2011)
3. Bienvenu, M., Kikot, S., Kontchakov, R., Podolskii, V.V., Ryzhikov, V., Zakharyaschev, M.: The complexity of ontology-based data access with OWL 2 QL and bounded treewidth queries. In: Proc. of the 26th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS (2017)
4. 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. IEEE Computer Society (2015)
5. Boguslavsky, I., Dikonov, V., Iomdin, L., Lazursky, A., Sizov, V., Timoshenko, S.: Semantic analysis and question answering: a system under development. In: Computational Linguistics and Intellectual Technologies. Papers from the Annual International Conference Dialogue. p. 21. No. 14 (2015)
6. 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)
7. Cook, S.A.: Characterizations of pushdown machines in terms of time-bounded computers. Journal of the ACM 18(1), 4–18 (1971)
8. Cuenca Grau, B., Horrocks, I., Krötzsch, M., Kupke, C., Magka, D., Motik, B., Wang, Z.: Acyclicity notions for existential rules and their application to query answering in ontologies. Journal of Artificial Intelligence Research 47, 741–808 (2013)
9. Gottlob, G., Greco, G., Leone, N., Scarcello, F.: Hypertree decompositions: Questions and answers. In: Proc. of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS (2016)
10. Gottlob, G., Leone, N., Scarcello, F.: Hypertree decompositions and tractable queries. Journal of Computer and System Sciences 64(3), 579–627 (2002)
11. Gottlob, G., Manna, M., Pieris, A.: Polynomial Rewritings for Linear Existential Rules, pp. 2992–2998. In: Proc. of the 24th Int. Joint Conf. on Artificial Intelligence, IJCAI (2015)
12. Gottlob, G., Orsi, G., Pieris, A.: Query rewriting and optimization for ontological databases. ACM Transactions on Database Systems 39(3), 25:1–25:46 (2014)
13. Kikot, S., Kontchakov, R., Podolskii, V., Zakharyaschev, M.: On the succinctness of query rewriting over shallow ontologies. In: Proc. of the Joint Meeting of the 23rd EACSL Annual Conf. on Computer Science Logic (CSL 2014) and the 29th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2014). pp. 57:1–57:10. ACM (2014)
14. 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)
15. König, M., Leclère, M., Mugnier, M.-L., Thomazo, M.: Sound, complete and minimal UCQ-rewriting for existential rules. Semantic Web 6(5), 451–475 (2015)
16. König, M., Leclere, M., Mugnier, M.-L.: Query rewriting for existential rules with compiled preorder. In: Proc. of the 24th Int. Joint Conf. on Artificial Intelligence, IJCAI (2015)
17. Sudborough, I.H.: On the tape complexity of deterministic context-free languages. Journal of the ACM 25(3), 405–414 (1978)
18. Thomazo, M.: Compact rewriting for existential rules. In: Proc. of the 23rd Int. Joint Conf. on Artificial Intelligence, IJCAI (2013)
19. Venkateswaran, H.: Properties that characterize LOGCFL. Journal of Computer and System Sciences 43(2), 380–404 (1991)