On the Succinctness of Query Rewriting over Shallow Ontologies

Stanislav Kikot
Roman Kontchakov
Department of Computer Science and Information Systems, Birkbeck, University of London, UK
{kikot, roman}@dcs.bbk.ac.uk

Vladimir V. Podolskii
Steklov Mathematical Institute
Moscow, Russia
podolskii@mi.ras.ru

Michael Zakharyaschev
Department of Computer Science and Information Systems, Birkbeck, University of London, UK
michael@dcs.bbk.ac.uk

Abstract
We investigate the succinctness problem for conjunctive query rewritings over OWL 2 QL ontologies of depth 1 and 2 by means of hypergraph programs computing Boolean functions. Both positive and negative results are obtained. We show that, over ontologies of depth 1, conjunctive queries have polynomial-size nonrecursive datalog rewritings; tree-shaped queries have polynomial positive existential rewritings; however, in the worst case, positive existential rewritings can be superpolynomial. Over ontologies of depth 2, positive existential and nonrecursive datalog rewritings of conjunctive queries can suffer an exponential blowup, while first-order rewritings can be superpolynomial unless NP ⊆ P/poly. We also analyse rewritings of tree-shaped queries over arbitrary ontologies and note that query entailment for such queries is fixed-parameter tractable.

Categories and Subject Descriptors 1.2.4 [Knowledge Representation Formalisms and Methods]

General Terms Ontology-based data access, description logic.

Keywords First-order query rewriting, succinctness, Boolean circuit complexity.

1. Introduction
Our concern in this paper is the size of conjunctive query (CQ) rewritings over OWL 2 QL ontologies. OWL 2 QL is a profile of the Web Ontology Language OWL 2 designed for ontology-based data access (OBDA). In first-order logic, any OWL 2 QL ontology can be given as a finite set of sentences of the form
\[ \forall \vec{x} (\varphi(\vec{x}) \rightarrow \exists \vec{y} \psi(\vec{x}, \vec{y})) \text{ or } \forall \vec{x} (\varphi(\vec{x}) \land \varphi'(\vec{x}) \rightarrow \bot) \] (1)
where \( \varphi, \varphi' \) and \( \psi \) are unary or binary predicates (such sentences are known as linear tuple-generating dependencies—or tgdsof

1 www.w3.org/TR/owl2-profiles

arity 2 and disjointness constraints). OWL 2 QL is a (nearly) maximal fragment of OWL 2 enjoying first-order rewritability of CQs: given an ontology \( T \) and a CQ \( q(\vec{x}) \), one can construct a first-order (FO) formula \( q'(\vec{x}) \) in the signature of \( q \) and \( T \) such that \( T, A \models q(\vec{a}) \) iff \( A \models q'(\vec{a}) \), for any set \( A \) of ground atoms (data) and any tuple \( \vec{a} \) of constants in \( A \). Thus, to find certain answers to \( q(\vec{x}) \) over \( (T, A) \), we can first compute an FO-rewriting \( q'(\vec{x}) \) of \( q \) and \( T \), and then evaluate it over any given data \( A \) using, for example, a relational database management system. The ontology \( T \) in the OBDA paradigm serves as a high-level global schema providing the user with a convenient query language over possibly heterogeneous data sources and enriching the data with additional knowledge. OBDA is widely regarded as a key ingredient of the new generation of information systems. OWL 2 QL is based on the DL–Lite family of description logics [4, 12]; other languages supporting first-order rewritability of CQs include linear, sticky and sticky-join sets of tgdsof

In practice, rewriting-based OBDA systems\(^2\) can only work efficiently with those CQs and ontologies that have reasonably short rewritings. This obvious fact raises fundamental succinctness problems such as: What is the size of FO-rewritings of CQs and OWL 2 QL ontologies in the worst case? Can rewritings of one type (say, nonrecursive datalog) be substantially shorter than rewritings of another type (say, positive existential)?\(^3\) First answers to these questions were given in [21] which constructed a sequence of CQs \( q_n \) and ontologies \( T_n \), for \( n = 1, 2, \ldots \), with only exponential positive existential (PE) and nonrecursive datalog (NDL) rewritings, and superpolynomial FO-rewritings unless NP \( \subseteq \text{P/poly} \); [21] also showed that NDL-rewritings can be exponentially more succinct than PE-rewritings, whereas FO-rewritings can be superpolynomially more succinct than PE-rewritings. These prohibitively high lower bounds are caused by the fact that the chasies (canonical models) for \( T_n \) contain full binary trees of depth \( n \) and give rise to exponentially-many homomorphisms from \( q_n \) to the trees of labelled nulls of the chasies, all of which have to be reflected in the rewritings of \( q_n \) and \( T_n \).

In this paper, we investigate succinctness of CQ rewritings over ‘shallow’ ontologies whose (polynomial-size) chasies are finite trees of depth 1 or 2 (which do not have chains of more than 1 or 2 labelled nulls). From the theoretical point of view, ontologies of depth 1 are important because their chasies can only generate linearly-many homomorphisms of CQs to the labelled nulls; on the other hand, ontologies of finite depth are typical in the real-world

\(^2\) See, e.g., QuOnto [30], Presto/Presto [35, 36], Rapid [14], Ontop [34], Requiem/Blackout [28, 29], Nyaya [17], Clipper [15] and PURE [25].
OBDA applications. We obtain both positive and, unexpectedly, ‘negative’ results, which are summarised below:

(i) any CQ and ontology of depth 1 have a polynomial-size NL-rewriting (Theorem 9);
(ii) there exist CQs and ontologies of depth 1 whose PE-rewritings are of superpolynomial size (Theorem 13);
(iii) any tree-shaped CQ and ontology of depth 1 have a PE-rewriting of polynomial size (Corollary 25);
(iv) the existence of polynomial-size FO-rewritings for all CQs and ontologies of depth 1 is equivalent to an open problem ‘NL/poly ≤ NC1?’ (Theorem 14);
(v) there exist CQs and ontologies of depth 2 whose NDL- and PE-rewritings are of exponential size, while FO-rewritings are of superpolynomial size unless NP ⊆ P/poly (Theorem 17);
(vi) the existence of polynomial-size FO-rewritings for all CQs and ontologies of depth 2 with polynomially-many tree witnesses is equivalent to an open problem ‘NP/poly ≤ NC1?’ (Theorem 18).

We prove (i)–(vi) by establishing a fundamental connection between FO-, PE- and NDL-rewritings, on the one hand, and, respectively, formulas, monotone formulas and monotone circuits computing certain monotone Boolean functions, on the other. These functions are associated with hypergraph representations of the tree-witness rewritings [24], reflecting possible homomorphisms of the given CQ to the labelled nulls of the bases for the given ontology. In particular, hypergraphs H of degree 2 (every vertex in which belongs to 2 hyperedges) correspond to CQs q and ontologies T of depth 1 such that answering q over T and single-individual data instances amounts to computing the hypergraph function for H. We show that representing Boolean functions as hypergraphs of degree 2 is polynomially equivalent to representing their duals as nondeterministic branching programs (NBPs) [19]. This correspondence and known results on NBPs [20, 33] give (i), (ii) and (iv) above. To prove (v) and (vi), we observe that hypergraphs of degree 3 are computationally as powerful as nondeterministic Boolean circuits (NP/poly) and encode the function CLIQUE(n, k) (graph ε with k vertices has a k-clique) as CQs over ontologies of depth 2. Finally, we show that any tree-shaped CQ q and ontology T have a PE-rewriting of size $O(|T|^2 |\vec{x}|^{3+\log \gamma})$, where $d$ is a parameter related to the number of tree witnesses sharing a common variable. This gives (iii) since $d = 2$ for ontologies of depth 1. We also note that the problem ‘T, A |= q’?, for tree-shaped Boolean CQs and any T, is fixed-parameter tractable, with parameter $|q|$ (recall that the problem ‘A |= q’? is, for tree-shaped q, known to be tractable [37], while ‘T, A |= q’? is NP-hard [23]). All omitted proofs can be found in [22].

As shown in [18], exponential rewritings can be made polynomial at the expense of polynomially-many additional existential quantifiers over a domain with two constants not necessarily occurring in the CQs; cf. [6]. Intuitively, given q, T and A, the extra quantifiers guess a homomorphism from q to the chase for (T, A), whereas the standard rewritings (without extra constants) represent such homomorphisms explicitly (likewise non-deterministic finite automata are exponentially more succinct than deterministic ones, and quantified Boolean formulas are exponentially more succinct than Boolean formulas; see also [16] for more details and discussions. A more practical utilisation of additional constants was suggested in the combined approach to OBDA [26], where they are used to construct a polynomial-size encoding of the chase for the given ontology and data over which the original CQ is evaluated. This encoding may introduce (exponentially-many in the worst case) spurious answers that are eliminated by a special polynomial-time filtering procedure.

2. The Tree-Witness Rewriting

In this paper, we assume that an ontology, T, is a finite set of tuple-generating dependencies (tgds) of the form

$$\forall \vec{x} (\varphi(\vec{x}) \rightarrow \exists \vec{y} \bigwedge \psi_i(\vec{x}, \vec{y})),$$

(2) where $\varphi$ and the $\psi_i$ are unary or binary atomic without constants and $|\vec{x}|, |\vec{y}| \leq 2$. These tgds are expressive via tgds in (1) using fresh binary predicates, whereas disjointness constraints in (1) do not contribute much to the size of rewritings [10, Theorem 11]. Although the language given by (1) is slightly different from OWL 2 QL, all the results obtained here are applicable to OWL 2 QL ontologies as well; for more details, consult [16]. When writing tgds, we will omit the universal quantifiers. The size, $|T|$, of T is the number of predicate occurrences in T. A data instance, A, is a finite set of ground atoms. The set of individual constants in A is denoted by ind(A). Taken together, T and A form the knowledge base (KB) (T, A). To simplify notation, we will assume that the data instances in all KBs are complete in the following sense: for any ground atom S(\vec{a}) with \vec{a} from ind(A), if T, A ⊨ S(\vec{a}) then S(\vec{a}) ∈ A (see Remark 1 below).

A conjunctive query (CQ) q(\vec{x}) is a formula $\exists \vec{y} \varphi(\vec{x}, \vec{y})$, where $\varphi$ is a conjunction of unary or binary atoms $S(\vec{x})$ with $\vec{x} \subseteq \vec{y}$ (without loss of generality, we assume that CQs do not contain constants). A tuple \vec{a} of individual constants from A is a certain answer to q(\vec{x}) over (T, A) if T ⊨ q(\vec{a}) for all models T of T and A; in this case we write T, A ⊨ q(\vec{a}). If \vec{x} = ∅, the CQ q is called Boolean; a certain answer to such a q over (T, A) is ‘yes’ if T, A, q and ‘no’ otherwise. Where convenient, we regard a CQ as the set of its atoms. The size |q| of a CQ q is the number of symbols in q.

Given a CQ q(\vec{x}) and an ontology T, an FO-formula q(\vec{x}) without constants is called an FO-rewriting of q(\vec{x}) and T if, for any (complete) data instance A and any \vec{a} from ind(A), we have (T, A) ⊨ q(\vec{a}) iff A ⊨ q(\vec{a}). If q is a positive existential formula, we call it a PE-rewriting of q and T. We also consider rewritings in the form of nonrecursive datalog queries.

Recall [1] that a datalog program, Π, is a finite set of Horn clauses $\forall \vec{x} (\gamma_1 \land \cdots \land \gamma_m \rightarrow \gamma_0)$, where each $\gamma_i$ is an atom of the form $P(x_1, \ldots, x_l)$ with $x_i \in \vec{x}$. The atom $\gamma_0$ is the head of the clause, and $\gamma_1 \ldots, \gamma_m$ its body. All variables in the head must also occur in the body. A predicate $P$ depends on $Q$ in Π if Π has a clause with $P$ in the head and $Q$ in the body; Π is nonrecursive if this dependence relation is acyclic. For a nonrecursive program Π and an atom q(\vec{x}), (Π, q) is called an NDL-rewriting of q(\vec{x}) and T in case T, A ⊨ q(\vec{a}) iff Π, A ⊨ q(\vec{a}), for any (complete) A and tuple \vec{a} from ind(A). The size of a rewriting is the number of symbols in it.

Remark 1. Rewritings over arbitrary data are defined without stipulating that the data instances in KBs are complete. It is readily seen [22] that, for any NDL-rewriting (Π, q’)(q) of q and T over complete data, there is an NDL-rewriting (Π’, q’)(q) over arbitrary data with |Π’| ≤ |Π| + O(|T|). Similarly, for a PE-rewriting q’ of q and T over complete data, there is a PE-rewriting q’ over arbitrary data with |q’| ≤ O(|q’| · |T|).

We now define an improved version of the tree-witness PE-rewriting of [24] that will be used to establish links with formulas and circuits computing certain monotone Boolean functions.

As is well-known [1], for any consistent KB (T, A), there is a canonical model or chase $C_{T,A}$ such that $T, A \models q(\vec{a})$ iff

$^1$Thus, we do not allow the rewriting from [18] as it contains constants.
We call \( T = ( q \textit{conflicting} | \exists y S(y, z) ) \) of depth 1, whereas \( T \cap ( P(x, y) \to \exists z S(y, z) ) \) of infinite depth because \( C^\theta_P(A, q) \) contains an infinite chain of the form \( P(A, q_1) \ldots P(w_1, w_2) \). Suppose we are given a CQ \( q(x) \) and an ontology \( T \). For a pair \( t = (t_1, t_2) \) of disjoint sets of variables in \( q \), with \( t_1 \not\subseteq \bar{t} \) and \( t_2 \not\subseteq \bar{t} \), we set
\[
q_{tw}(t_1) = \bigvee_{q \in Q_T \text{ independent} \atop \theta \subseteq \theta_T} \exists \bar{y} \left( \bigwedge_{x \in t_1} S(z) \land \bigwedge_{x \in t_2} tw_i(t_2) \right). \tag{4}
\]

Example 2. Consider an ontology \( T \) with the tgs
\[
A_1(x) \to \exists y ( R(x, y) \land Q(x, y) )
\]
and the CQ
\[
q(x_1, x_2) = \exists y_1 y_2 \left( R_1(x_1, y_1) \land Q(y_2, y_1) \land R_2(x_2, y_2) \right).
\]
The CQ \( q \) is shown below alongside \( C_{T, A}^{A_1(q)} \) and \( C_{T, A}^{A_2(q)} \):
rewriting of $q$ and $T$. This gives the first claim in the following theorem; the second one requires some basic skills in datalog programming. (Recall [3] that monotone Boolean formulas and circuits contain only $\land$ and $\lor$.)

**Theorem 5.** If $f_{H^\phi}$ is computed by some (monotone) Boolean formula $\chi$ then there exists a (PE-) FO-rewriting of $q$ and $T$ of size $O(|\chi| \cdot |q| \cdot |T|)$.

If $f_{H^\phi}$ is computed by some monotone Boolean circuit $C$ then there exists an NDL-rewriting of $q$ and $T$ of size $O(|C| \cdot |q| \cdot |T|)$.

Thus, the problem of constructing short rewritings is reducible to the problem of finding short (monotone) Boolean formulas or circuits computing the hypergraph functions.

In the next section, we consider hypergraphs as programs for computing Boolean functions and compare them with the well-known formalisms of nondeterministic branching programs (NBP)s and nondeterministic Boolean circuits [3, 19].

### 4. Hypergraphs, NBP's and Boolean Circuits

Let $p_1, \ldots, p_n$ be propositional variables. An input to a hypergraph program or an NBP is a vector $\vec{\alpha} \in \{0, 1\}^n$ assigning the truth-value $\vec{\alpha}(p_i)$ to each of the $p_i$. We extend this notation to negated variables and constants by setting $\vec{\alpha}(\neg p_i) = \neg \vec{\alpha}(p_i)$, $\vec{\alpha}(0) = 0$ and $\vec{\alpha}(1) = 1$.

A hypergraph program (HGP) is a hypergraph $H = (V, E)$ in which every vertex is labelled with 0, 1, $p_i$, or $\neg p_i$. We say that the hypergraph program $H$ computes a Boolean function $f$ in case, for any input $\vec{\alpha}$, we have $f(\vec{\alpha}) = 1$ iff there is an independent subset in $E$ that covers all zeros—that is, contains all the vertices in $V$ labelled with 0 under $\vec{\alpha}$. A hypergraph program is monotone if there are no negated variables among its vertex labels. The size, $|H|$, of a hypergraph program $H$ is the number of hyperedges in it. We say that a hypergraph program (hypergraph) $H$ is of degree $\leq n$ if every vertex in it belongs to at most $n$ hyperedges; $H$ is of degree $n$ if every vertex in it belongs to exactly $n$ hyperedges. We denote by $HGP(f)$ (HGP$^n(f)$) the minimal size of hypergraph programs (of degree $\leq n$) computing $f$; HGP$^\Phi(f)$ and HGP$^\Phi_n(f)$ are used for the size of monotone programs.

Our first result in this section establishes a link between hypergraph programs of degree $\leq 2$ and NBP$s$. Note [22] that any (monotone) hypergraph program $H$ of degree $\leq 2$ computing a function $f$ can be converted to a (monotone) hypergraph program $H'$ of degree 2 computing $f$ with $|H'| = |H| + 3$.

Recall [19] that an NBP is a directed multigraph with two distinguished vertices, $s$ and $t$, and the arcs labelled with 0, 1, $p_i$, or $\neg p_i$ (the arcs of the first type have no effect, the arcs of the second type are called rectifiers, and those of the third and fourth types contacts). We assume that $s$ has no incoming and $t$ no outgoing arcs, and note that NBP$s$ may have multiple parallel arcs (with distinct labels) connecting two nodes. We write $v \rightarrow_\phi v'$ if there is a directed path from $v$ to $v'$ every edge of which is labelled with 1 under $\vec{\alpha}$. An NBP computes a Boolean function $f$ if $f(\vec{\alpha}) = 1$ just in case $s \rightarrow_\phi t$. The size of an NBP is the number of arcs in it. An NBP is monotone if it has no negated variables among its labels. We denote by NBP$^\Phi(f)$ (respectively, NBP$^\Phi_n(f)$) the minimal size of (monotone) NBP$s$ computing $f$. As usual, $f^*$ is the Boolean function dual to $f$.

**Theorem 6.** (i) For any Boolean function $f$, HGP$^2(f)$ and NBP$^\Phi(\neg f)$ are polynomially related.

(ii) For any monotone Boolean function $f$, HGP$^\Phi(f)$ and NBP$^\Phi(f^*)$ are polynomially related.

*Proof.* We only prove (i); (ii) is proved by the same argument. Suppose $\neg f$ is computed by an NBP $G$. We construct a hypergraph program $H$ of degree $\leq 2$ as follows. For each arc $e$ in $G$, $H$ has two vertices $e^1$ and $e^2$, which represent the beginning and the end of $e$. The vertex $e^1$ is labelled with the negated label of $e$ in $G$ and $e^2$ with 1. We also add to $H$ a vertex $t$ labelled with 0. For each arc $e$ in $G$, $H$ has an $e$-hyperedge $\{e^1, e^2\}$. For each vertex $v$ in $G$ but $s$ and $t$, $H$ has a $v$-hyperedge that consists of all vertices $e^i$, for the arcs $e$ leading to $v$, and all vertices $e^j$, for the arcs $e$ leaving $v$. For the vertex $t$, $H$ contains a hyperedge that consists of $t$ and all vertices $e^1$, for the arcs $e$ leading to $t$. We claim that the constructed hypergraph program $H$ computes $f$. Indeed, if $s \not\rightarrow_\phi t$ in $G$ then the following subset of hyperedges is independent and covers all zeros: all $e$-hyperedges, for the arcs $e$ reachable from $s$ and labelled with 1 under $\vec{\alpha}$, and all $v$-hyperedges with $s \not\rightarrow_\phi v$. Conversely, if $s \rightarrow_\phi t$ then it can be shown by induction that, for each arc $e_i$ of the path, the $e_i$-hyperedge must be in the cover of all zeros. Thus, no independent set can cover $t$, which is labelled with 0.

Suppose $f$ is computed by a hypergraph program $H$ of degree 2 with hyperedges $e_1, \ldots, e_k$. We first provide a graph-theoretic characterisation of independent sets covering all zeros based on the implication graph [5] (or the chain criterion of [9, Lemma 8.3.1]).

With any hyperedge $e_i$ we associate a propositional variable $p_i$, and with an input $\vec{\alpha}$ we associate the following set $\Phi_{\vec{\alpha}}$ of binary clauses:

- $\neg p_i \lor \neg p_j$, if $\vec{\alpha}(e_i \cap e_j) \neq \emptyset$ (informally: intersecting hyperedges cannot be chosen at the same time),
- $p_i \lor p_j$, if there is $v \in e_i \cap e_j$ such that $\vec{\alpha}(v) = 0$ (informally: all zeros must be covered; note that all vertices have at most two incident edges).

By definition, $X$ is an independent set covering all zeros just in case $X = \{e_i \mid \vec{\beta}(p_i) = 1\}$, for some assignment $\vec{\beta}$ satisfying $\Phi_{\vec{\alpha}}$. Let $B_{\vec{\alpha}}$ be a directed graph $(V, E_{\vec{\alpha}})$ with

- $V = \{e_i^+, e_i^- \mid 1 \leq i \leq k\}$,
- $E_{\vec{\alpha}} = \{(e_i^+, e_j^+) \mid e_i \cap e_j \neq \emptyset\} \cup \{(e_i^-, e_j^+) \mid v \in e_i \cap e_j \land \vec{\alpha}(v) = 0\}$

By [9, Lemma 8.3.1], $\Phi_{\vec{\alpha}}$ is satisfiable iff there is no $e_i$ with a (directed) cycle going through both $e_i^+$ and $e_i^-$. It will be convenient for us to regard the $B_{\vec{\alpha}}$, for assignments $\vec{\alpha}$, as a single labelled directed graph $B$ with arcs of the form $(e_i^+, e_j^-)$ labelled with 1 and arcs of the form $(e_i^-, e_j^+)$ labelled with $\neg v$, for $v \in e_i \cap e_j$. It should be clear that $B_{\vec{\alpha}}$ has a cycle going through $e_i^+$ and $e_i^-$ iff $e_i^- \rightarrow_\phi e_i^+$ and $e_i^+ \rightarrow_\phi e_i^-$ in $B$.

The required NBP contains two distinguished vertices, $s$ and $t$, and, for each hyperedge $e_i$, two copies, $B_i^+$ and $B_i^-$, of $B$ from arcs $s$ to the $e_i^+$ vertex of $B_i^+$, from the $e_i^-$ vertex of $B_i^+$ to the $e_i^+$ vertex of $B_i^-$ and from the $e_i^-$ vertex of $B_i^-$ to $t$; these arcs are labelled with 1. This construction guarantees that $s \rightarrow_\phi t$ if there is $e_i$ such that $B_i$ contains a cycle going through $e_i^+$ and $e_i^-$. 

In terms of expressive power, polynomial-size NBP$s$ are a non-uniform analogue of the complexity class NL; in symbols: NBP(poly) = NL/poly. Compared to other non-uniform computational models, (monotone) NBP$s$ sit between (monotone) Boolean formulas and Boolean circuits [33]. As shown above, a (monotone) Boolean function $f$ is computable by a polynomial-size (monotone) HGP of degree $\leq 2$ iff its dual $f^*$ is computable by a polynomial-size (monotone) NBP. (The problem whether $f^*$ can be replaced with $f$ open; a negative solution would give a solution to the open problem 5 from [33].) Thus, (monotone) HGPs of degree $\leq 2$ also sit between (monotone) Boolean formulas and Boolean circuits. However, (monotone) HGPs of degree $\leq 3$ turn
out to be much more powerful than those of degree $\leq 2$: we show now that polynomial-size (monotone) HGP's of degree $\leq 3$ can compute NP-hard Boolean functions.

A function $f : \{0, 1\}^n \to \{0, 1\}$ is computed by a nondeterministic Boolean circuit $C(\vec{x}, \vec{y})$, with $|\vec{x}| = n$, if for any $\vec{a} \in \{0, 1\}^n$, we have $f(\vec{a}) = 1$ if there is $\vec{b} \in \{0, 1\}^n$ with $C(\vec{a}, \vec{b}) = 1$ (the variables in $\vec{y}$ are also known as a certificate). We say that a nondeterministic circuit $C(\vec{x}, \vec{y})$ is monotone if the negations in $C$ are only applied to variables in $\vec{y}$. Denote by $NBC(f)$ (respectively, $NBC^+(f)$) the minimal size of (monotone) nondeterministic Boolean circuits computing $f$.

**Theorem 7.** (i) For any Boolean function $f$, $HGP(f)$, $HGP^3(f)$ and $NBC(f)$ are polynomially related.

(ii) For any monotone Boolean function $f$, $HGP^e(f)$, $HGP^3(f)$ and $NBC^e(f)$ are polynomially related.

**Proof.** It is easy to see that any function $f$ computable by a (monotone) HGP $C$ can also be computed by a monotone nondeterministic circuit of size poly$(|H|)$. Conversely, suppose $f$ is computed by a nondeterministic circuit $C(\vec{x}, \vec{y})$. Let $g_1, \ldots, g_n$ be the nodes of $C$ (including the inputs $\vec{x}$ and $\vec{y}$). We construct an HGP of degree $\leq 3$ computing $f$ by taking, for each $i$, a vertex $g_i$, labelled with 0 and a pair of hyperedges $e_{g_i}$ and $e_{g_i}$, both containing $g_i$. No other edge contains $g_i$, and so either $e_{g_i}$ or $e_{g'_i}$ should be present in any cover of zeros. (Intuitively, if the node $g_i$ is positive then $e_{g_i}$ belongs to the cover; otherwise, $e_{g'_i}$ is there.) To ensure this property, for each input variable $x_i$, we add a vertex labelled with $\neg x_i$, to $e_{g_i}$, and a fresh vertex labelled with $x_i$, to $e_{g'_i}$. For each gate $g_i$, we consider three cases.

- If $g_i = \neg g_j$, then we add a vertex labelled with 1 to $e_{g_i}$ and $e_{g_j}$, and a vertex labelled with 1 to $e_{g_j}$.
- If $g_i = g_j \lor g_j$, then we add a vertex labelled with 1 to $e_{g_i}$ and $e_{g_j}$, add a vertex labelled with 1 to $e_{g_i}$ and $e_{g_j}$; then, we add vertices $h_i$ and $h_j$ labelled with 1 to $e_{g_i}$ and $e_{g_j}$, respectively, and a vertex $u_i$ labeled with 0 to $e_{g_j}$; finally, we add hyperedges \{$h_i, u_i$\} and \{$h_j, u_i$\}.
- If $g_i = g_j \land g_j$, then we use the dual construction.

It is readily seen that $e_{g_i}$ is in the cover if it contains $e_{g_j}$ in the first case, and if it contains at least one of $e_{g_i}$ and $e_{g'_j}$ in the second case. Finally, we add a vertex labelled with 0 to $e_{g_j}$ for the output gate $g$ of $C$. By induction on the structure of $C$ one can show that, for each $\vec{a}$, there is $\vec{b}$ such that $C(\vec{a}, \vec{b}) = 1$ iff the constructed hypergraph program returns 1 on $\vec{a}$.

If $C$ is monotone, we remove all vertices labelled with $\neg x_i$. Then, for an input $\vec{a}$, there is a cover of zeros in the resulting hypergraph iff there are $\vec{b}$ and $\vec{a}' \leq \vec{a}$ with $C(\vec{a}', \vec{b}) = 1$.

Now, we use the developed machinery to investigate the size of CQ rewritings over ontologies of depth 1 and 2.

5. **Rewritings over Ontologies of Depth 1**

**Theorem 8.** For any ontology $T$ of depth 1 and any CQ $q$, the hypergraph $H^T_q$ is of degree $\leq 2$ and $|\Theta_f^T| \leq |q|$.

**Proof.** We have to show that every atom in $q$ belongs to at most two $q_{t}$, $t \in \Theta_f^T$. Let $t = (u, v)$ be a tree witness and $y \in v$. As $T$ is of depth 1, $t = (y)$ and $t$ consists of those variables $z$ in $q$ for which $S(y, z) \in q$ or $S(z, y) \in q$, for some $S$. So different tree witnesses have different internal variables $y$. An atom of the form $A(u) \in q$ is in $q_{t}$ if $u = y$. An atom of the form $P(u, v) \in q$ is in $q_{t}$ if either $u = y$ or $v = y$. Thus, $P(u, v) \in q$ can only be covered by the tree witness with internal $u$ and by the tree witness with internal $v$.

**Theorem 9.** Any CQ $q$ and ontology $T$ of depth 1 have a polynomial-size NDL-rewriting.

**Proof.** By Theorem 8, the hypergraph $H^T_q$ is of degree $\leq 2$, and so there is a polynomial-size HGP of degree $\leq 2$ computing $f_{H^T_q}$. By Theorem 6, we have a polynomial-size monotone NBP computing $f_{H^T_q}$. But then we also have a polynomial-size monotone Boolean circuit that computes $f_{H^T_q}$ (see, e.g., [33]). By swapping $\land$ and $\lor$ in this circuit, we obtain a polynomial-size monotone circuit computing $f_{H^T_q}$. It remains to apply Theorem 5.

We show next that any hypergraph $H$ of degree 2 is representable by means of a CQ $q_H$ and an ontology $T_H$ of depth 1 in the sense that $H$ is isomorphic to $H^T_q$ (H $\cong H^T_q$, in symbols). We can assume that $H = (V, E)$ comes with two fixed maps $t_1, t_2 : V \to E$ such that $t_i(v) \neq t_j(v), v \in t_i(v) \lor v \in t_i(v)$, for any $v \in V$. For each hyperedge $e \in E$, we take an individual variable $z_e$ and denote by $z$ the vector of all such variables. For every vertex $v \in V$, we take a binary predicate $R_v$ and set

$$q_H = \exists z \bigwedge_{v \in V} R_v(z_{t_1(v)}, z_{t_2(v)}).$$

Let $T_H$ be an ontology with the following tgd's, for $e \in E$:

$$R_e(x) \rightarrow \exists y \left[ \bigwedge_{v \in V} R_v(x, y) \land \bigwedge_{e \in V} R_v(y, x) \right].$$

**Example 10.** Consider $H = (V, E)$ with $V = \{v_1, v_2, v_3, v_4\}$ and $E = \{e_1, e_2, e_3\}$, where $e_1 = \{v_1, v_2, v_3\}$, $e_2 = \{v_3, v_4\}$, $e_3 = \{v_1, v_2, v_3\}$, and assume that

$$t_1 : v_1 \mapsto e_1, v_2 \mapsto e_3, v_3 \mapsto e_1, v_4 \mapsto e_2,$$
$$t_2 : v_1 \mapsto e_3, v_2 \mapsto e_1, v_3 \mapsto e_2, v_4 \mapsto e_3.$$

The hypergraph $H$ is shown in the picture below, where each $v_6$ is represented by an edge, $i_1(v_6)$ is indicated by the circle-shaped end of the edge and $i_2(v_6)$ by the diamond-shaped end of the edge; the $e_j$ are shown as large grey squares:

In this case,

$$q_H = \exists z_{e_1}, z_{e_2}, z_{e_3} \left( R_{e_1}(z_{e_1}, z_{e_3}) \land R_{e_2}(z_{e_2}, z_{e_3}) \land R_{e_3}(z_{e_1}, z_{e_2}) \land R_{e_3}(z_{e_1}, z_{e_3}) \right)$$

and the ontology $T_H$ consists of the following tgd's:

$$A_{e_1}(x) \rightarrow \exists y \left[ R_{e_1}(x, y) \land R_{e_2}(y, x) \land R_{e_3}(y, x) \right],$$
$$A_{e_2}(x) \rightarrow \exists y \left[ R_{e_2}(x, y) \land R_{e_4}(y, x) \right],$$
$$A_{e_3}(x) \rightarrow \exists y \left[ R_{e_1}(x, y) \land R_{e_2}(y, x) \land R_{e_4}(x, y) \right] .$$

The model $c^{\lambda}_{T_H}$ is shown on the right-hand side of the picture above. Note that each $z_e$ determines the tree witness $t^e$ in which $q_{t^e} = \{R_v(z_{t_1(v)}, z_{t_2(v)}) \mid v \in e\}$; $t^e$ and $t^f$ are conflicting iff
e ∩ e' ≠ ∅. It follows that H is isomorphic to $H^p_{nH}$. In fact, this example generalises to the following:

**Theorem 11.** Any hypergraph H of degree 2 is isomorphic to $H^p_{nH}$ with $T_0$ being an ontology of depth 1.

We now show that answering $q_f$ over $T_0$ and certain single-individual data instances amounts to computing the Boolean function $f_H$. Let $H = (V, E)$ be a hypergraph of degree 2 with $V = \{v_1, \ldots, v_n\}$ and $E = \{e_1, \ldots, e_m\}$. We denote by $\alpha(v)$ the i-th component of $\alpha \in \{0, 1\}^n$, by $\beta(e)$ the j-th component of $\beta \in \{0, 1\}^m$, and set

$$A_{\alpha, \beta} = \{ R_v(a, a) \mid \alpha(v) = 1 \} \cup \{ A_{e_j}(a) \mid \beta(e) = 1 \}.$$

**Theorem 12.** Let $H = (V, E)$ be a hypergraph of degree 2. Then $T_0, A_{\alpha, \beta} \models q_f$ if $f_H(\alpha, \beta) = 1$, for any $\alpha \in \{0, 1\}^{|V|}$ and $\beta \in \{0, 1\}^{|E|}$.

Proof. ($\Rightarrow$) Let $X$ be an independent subset of $E$ such that $\bigvee_{e \in X} p_e \land \bigwedge_{e \in X} p_e$ is true on $\alpha$ (for the $p_e$) and $\beta$ (for the $p_e$). Define $h: q_f \rightarrow C_{T_0, A_{\alpha, \beta}}$ by taking $h(z_e) = a$ if $e \notin X$ and $h(z_e) = w_e$, otherwise, where $w_e$ is the labelled null in the canonical model $C_{T_0, A_{\alpha, \beta}}$ introduced to witness the existential quantifier in (6). One can check that $h$ is a homomorphism, and so $T_0, A_{\alpha, \beta} \models q_f$.

($\Leftarrow$) Suppose $h: q_f \rightarrow C_{T_0, A_{\alpha, \beta}}$ is a homomorphism. We show that the set $X = \{ e \in E \mid h(z_e) \neq a \}$ is independent. Indeed, if $e, e' \in X$ and $e \cap e' \neq \emptyset$, then $h$ sends one variable of the $R_v$-atom to the labelled null $w_e$ and the other end to $w_e$, which is impossible. We claim that $f_H(\alpha, \beta) = 1$. Indeed, for each $e \in V \setminus V_X$, $h$ sends both ends of the $R_v$-atom to $a$, and so $\alpha(v) = 1$. For each $e \in E$, we must have $h(z_e) = w_e$ because $h(z_e) \neq a$, and so $\beta(e) = 1$. It follows that $f_H(\alpha, \beta) = 1$.

We are fully equipped now to show that there exist CQs and ontologies of depth 1 without polynomial-size PE-rewritings:

**Theorem 13.** There is a sequence of CQs $q_n$ and ontologies $T_n$ of depth 1, both of polynomial size in $n$, such that any PE-rewriting of $q_n$ and $T_n$ of size $n^\Omega(\log n)$.

Proof. As shown in [20], there is a sequence $f_n$ of monotone Boolean functions that are computable by polynomial-size monotone NBPs, but any monotone Boolean formulas computing $f_n$ are of size $n^\Omega(\log n)$. In fact, $f_n$ checks whether two given vertices are connected by a path in a given undirected graph. By Theorem 6, there is a sequence of polynomial-size monotone HGPS $H_n$ of degree 2 computing $f_n$. By applying Theorem 11 to the hypergraph $H_n$ of $H_n$, we obtain a sequence of $q_n$ and $T_n$ such that $H_n \cong H_{n, H_n}$. We show now that any PE-rewriting $q'_n$ of $q_n$, we can be transformed into a monotone Boolean formula computing $f_n$ and having size $\leq |q'_n|$.

To define it, we eliminate the quantifiers in $q'_n$ in the following way: take a constant $a$ and replace every subformula of the form $\exists x \psi(x)$ in $q'_n$ with $\psi(a)$, repeating this operation as many times as possible. The resulting formula $q''_n$ is built from atoms of the form $A(a)$, $R_v(a, a)$ and $S_v(a, a)$ using $\land$ and $\lor$. For every data instance $A$ with a single individual $a$, we have $T_n, A \models q_n$, if $A \models q''_n$. Let $\chi_n$ be the result of replacing $S_v(a, a)$ in $q''_n$, with $\bot$. $A_{\alpha}(a)$ with $p_e$ and $R_v(a, a)$ with $p_e$. Clearly, $|\chi_n| \leq |q''_n|$. By the definition of $A_{\alpha, \beta}$ and Theorem 12, we have

$$\chi_n(\alpha, \beta) = 1 \iff A_{\alpha, \beta} \models q''_n \iff T_n, A_{\alpha, \beta} \models q_n \iff f_{H_n}(\alpha, \beta) = 1.$$
Theorem 15. The HGP $H_{n,k}$ computes $\text{CLIQUE}_{n,k}$.

Proof. We show that, for each $\overline{c} \in \{0,1\}^{n(n-1)/2}$, there is an independent set $X$ of hyperedges covering all zeros in $H_{n,k}$ iff $\text{CLIQUE}_{n,k}(\overline{c}) = 1$.

($\Leftarrow$) Let function $\lambda: \{1,\ldots,k\} \to \{1,\ldots,n\}$ be such that $C = \{\lambda(i) \mid 1 \leq i \leq k\}$ is a $k$-clique in the graph given by $\overline{c}$. Then

$$X = \{f^{\lambda(i)} \mid 1 \leq i \leq k\} \cup \{h^{j,j'} \mid j \notin C, j' \in C\} \cup \{h^{j,j'} \mid j, j' \notin C \text{ and } j < j'\}$$

is independent and covers all zeros in $H_{n,k}$ under $\overline{c}$. Indeed, $X$ is independent because, in every $h^{j,j'} \in X$, the index $j$ does not belong to $C$. By definition, each $f^{\lambda(i)} \in X$ covers $v_i$, for $1 \leq i \leq k$. Thus, it remains to show that any $w_{jj'}$ with $e_{jj'} = 0$ (that is, the edge $(j, j')$ belongs to the complement of $C$) is covered by some hyperedge. All edges of the complement of $G$ can be divided into two groups: those that are adjacent to $C$, and those that are not. The $w_{jj'}$ that correspond to the edges of the former group are covered by the $h^{j,j'}$ from the middle disjunct of $X$, where $j$ corresponds to the end of the edge $(j, j')$ that is not $C$. To cover $w_{jj'}$ of the latter group, take $h^{j,j'}$ from the last disjunct of $X$.

($\Rightarrow$) Suppose that $X$ is an independent set which covers all zeros labelling the vertices of $H_{n,k}$, for an input $\overline{c}$. The vertex $v_i$ is labelled with $0$, and so there is $\lambda(i)$ such that $f^{\lambda(i)} \in X$. We claim that $C = \{\lambda(i) \mid 1 \leq i \leq k\}$ is a $k$-clique in the graph given by $\overline{c}$. Indeed, suppose that the graph has no edge between some vertices $j, j' \in C$, that is, $e_{jj'} = 0$ for $j < j'$. Since $w_{jj'}$ is labelled with $0$, it must be covered by a hyperedge in $X$, which can only be either $h^{j,j'}$ or $h^{j,j'}$ (see the picture above). But $h^{j,j'}$ intersects $f^{\lambda(j)}$ and $h^{j,j'}$ intersects $f^{\lambda(j')}$, which is a contradiction. $\Box$

We are now in a position to define $\mathcal{T}_{n,k}$ of depth 2 and $q_{n,k}$, both of polynomial size in $n$, that can compute $\text{CLIQUE}_{n,k}$. Let $q_{n,k}$ contain the following formulas (all variables are quantified):

$$\begin{align*}
T_{ij}(v_i, z_{ij}) & \quad \text{for } 1 \leq i \leq k, 1 \leq j \leq n, \\
P_{jj'}(w_{jj'}, x_{jj'}) & \quad \text{for } 1 \leq j < j' \leq n, \\
P_{jj'}(w_{jj'}, x_{jj'}) & \quad \text{for } 1 \leq j < j' \leq n, \\
Q(u_{jj'}, x_{jj'}), U(u_{jj'}, z_{ij}) & \quad \text{for } 1 \leq j \neq j' \leq n \quad \text{and } 1 \leq i \leq k.
\end{align*}$$

The picture below illustrates the fragments of $q_{n,k}$ centred in each variable of the form $z_{ij}$ and $x_{jj'}$ (the fragment centred in $x_{jj'}$ is similar to that of $x_{jj'}$ except the index of the $w_{jj'}$):

The ontology $\mathcal{T}_{n,k}$ mimics the arrangement of atoms in the layers depicted above and contains the following tgds, where $1 \leq i \leq k$ and $1 \leq j \neq j' \leq n$,

$$\begin{align*}
A_{ij}(x) & \mapsto \exists y \left[ \bigwedge_{j' \neq j} T_{ij'}(y, x) \land U(y, x) \land Q(y, x) \land A'_{ij}(y) \right], \\
A'_{ij}(x) & \mapsto \exists y \left[ T_{ij}(x, y) \land U(x, y) \right], \\
P_{jj'}(y, x) & \mapsto \exists y \left[ P_{jj'}(y, x) \land U(y, x) \land B'_{jj'}(y) \right], \\
B'_{jj'}(x) & \mapsto \exists y \left[ B_{jj'}(x, y) \land Q(x, y) \right].
\end{align*}$$

The canonical models $C_{\mathcal{T}_{n,k}}^{A_{ij}(a)}$ and $C_{\mathcal{T}_{n,k}}^{B_{jj'}(a)}$ are also illustrated in picture above with the horizontal dashed lines showing possible ways of embedding the fragments of $q_{n,k}$ into them. These embeddings give rise to the following tree witnesses:

$$\begin{align*}
t^{ij} & = (t^{ij}, t^{ij}) \text{ generated by } A_{ij}(x), \text{ for } 1 \leq i \leq k \text{ and } 1 \leq j \leq n, \text{ where} \\
t^{ij} & = (\{z_{ij} \mid 1 \leq i \leq n, j \neq j'\} \cup \{z_{ij} \mid 1 \leq i \leq k, i \neq i'\}), \\
t^{ij} & = \{v_i, z_{ij}\} \cup \{u_{jj}, 1 \leq j \neq n, j \neq j'\}; \\
g^{ij} & = (g^{ij}, g^{ij}) \text{ and } g^{ij} = (g^{ij}, g^{ij}), \text{ generated by } B_{jj'}(x) \text{ and } B_{jj'}(x), \text{ respectively, for } 1 \leq j < j' \leq n, \text{ where} \\
g^{ij} & = \{x_{ij} \cup \{z_{ij} \mid 1 \leq i \leq k\}, \\
g^{ij} & = \{w_{jj'}, u_{jj'}, x_{jj'}\}, \\
g^{ij} & = \{x_{ij} \cup \{z_{ij} \mid 1 \leq i \leq k\}, \\
g^{ij} & = \{w_{jj'}, u_{jj'}, x_{jj'}\}.
\end{align*}$$

The tree witnesses $t^{ij}, g^{ij}$ and $g^{ij}$ are uniquely determined by their most remote (from the root) variables, $z_{ij}, x_{ij}$ and $x_{ij}$, respectively, and correspond to the hyperedges $f^{ij}, h^{ij}, h^{ij}$ of $H_{n,k}$; their internal variables of the form $v_i, w_{jj'}$ and $u_{jj'}$ correspond to the vertices in the respective hyperedge.

For a vector $\overline{c}$ encoding a graph with $n$ vertices, let $\mathcal{A}_c$ be a data instance with one individual $a$ and the following atoms:

$$Q(a, a), U(a, a), A_{ij}(a), \text{ for } 1 \leq i \leq k \text{ and } 1 \leq j \leq n, \text{ and } P_{jj'}(a, a) \text{ and } P_{jj'}(a, a), \text{ for } 1 \leq j < j' \leq n \text{ with } e_{jj'} = 1.$$

Lemma 16. $\mathcal{T}_{n,k}, \mathcal{A}_c \models q_{n,k}$ iff $\text{CLIQUE}_{n,k}(\overline{c}) = 1$.

Proof. ($\Rightarrow$) Suppose $\mathcal{T}_{n,k}, \mathcal{A}_c \models q_{n,k}$. Then there is a homomorphism $g$ from $q_{n,k}$ to the canonical model $\mathcal{C}$ of $(\mathcal{T}_{n,k}, \mathcal{A}_c)$. Since the only points of $\mathcal{C}$ that belong to $\exists y T_{ij}(x, y)$ are of the form $e_{ij}$ (in the picture above) and $q_{n,k}$ contains atoms of the form $T_{ij}(v_i, z_{ij})$, there is $\lambda: \{1, \ldots, k\} \to \{1, \ldots, n\}$ such that $g(v_i) = c_{\lambda(i)}$. We claim that $C = \{\lambda(i) \mid 1 \leq i \leq k\}$ is a $k$-clique in the graph given by $\overline{c}$.\[\]
We first show that $\lambda$ is injective. Suppose to the contrary that $\lambda(i) = \lambda(i') = j$, for $i \neq i'$. Since $q_{n,k}^i$ contains $T_{ij}(v_i, z_{ij})$ and $T_{ij}'(v_i, z_{ij})$, we have $g(z_{ij}) = c_{ij}$ and $g(z_{ij}) = c_{ij}'$. Take $j' \neq j$. Since $U((uj_{ij}', z_{ij}), (uj_{ij}', z_{ij}')) \in q_{n,k}$, we obtain $g(u_{ij}) = c_{ij}$ and $g(u_{ij}) = c_{ij}'$, contrary to $j \neq j'$.

Next, we show that $e_{ij} = 1$, for all $j, j' \in C$ with $j < j'$. Since $U((uj_{ij}', z_{ij}), z_{ij})$ is in $q_{n,k}$, we have $g(u_{ij}) = c_{ij}$, and so $g(x_{ij}) = a$. Similarly, we also have $g(u_{ij}') = c_{ij}'$ and $g(x_{ij}') = a$. Then, since $q_{n,k}^i$ contains both $P_{ij}'(u_{ij}', x_{ij}')$ and $P_{ij}(u_{ij}, x_{ij})$ and $C$ contains no pair of points in both $P_{ij}'$ and $P_{ij}$ apart from $(a, a)$, we obtain $e_{ij} = 1$ whenever $g(x_{ij}') = g(x_{ij}) = a$, as shown in the picture below:

$$v_i \quad u_{ij} \quad v_{ij} \quad u_{ij}' \quad v_{ij}' \quad T_{ij} \quad T_{ij}' \quad U \quad Q \quad x_{ij} \quad x_{ij}'$$

($\Leftarrow$) Suppose $\lambda: \{1, \ldots, k\} \rightarrow \{1, \ldots, n\}$ is a $k$-clique. We construct a homomorphism $g$ from $q_{n,k}^i$ to the canonical model of $(\mathcal{T}_{n,k}, \mathcal{A}_C)$ relying upon the cover $X$ constructed for $H_{n,k}$ in the proof of Theorem 15, ($\Leftarrow$). The internal variables of the tree witnesses from $X$ are sent to labelled nulls, and all other points are sent to $a$. It follows that $\mathcal{T}_{n,k}, \mathcal{A}_C \models q_{n,k}$.

**Theorem 17.** There exists a sequence of CQs $q_n$, and ontologies $\mathcal{T}_n$ of depth 2 any PE- or NDL-rewritings of which are of exponential size, while any FO-rewriting is of superpolynomial size unless NP $\subseteq$ P/poly.

**Proof.** Given a PE-, FO- or NDL-rewriting $q_{n,k}^i$ of $q_{n,k}$ and $\mathcal{T}_{n,k}$, we show how to construct, respectively, a monotone Boolean formula, a Boolean formula or a monotone Boolean circuit for the function $\text{CLIQUE}_{n,k}^i$ of size $|q_{n,k}^i|$.

Suppose $q_{n,k}^i$ is a PE-rewriting of $q_{n,k}$ and $\mathcal{T}_{n,k}$. We eliminate the quantifiers in $q_{n,k}^i$ by replacing any $\exists x \psi(x)$ in $q_{n,k}^i$ with $\psi(a)$, any $P_{ij}'(a, a)$ and $P_{ij}(a, a)$ with $e_{ij}$, any $T_{ij}(a, a)$, $A_{ij}(a)$ and $B_{ij}(a)$ with 0, and $U(a, a)$, $Q(a, a)$, $A_{ij}(a)$ and $B_{ij}(a)$ with 1. One can check that the resulting monotone Boolean formula computes $\text{CLIQUE}_{n,k}^i$. If $q_{n,k}^i$ is an FO-rewriting, then we also replace $\forall x \psi(x)$ with $\psi(a)$.

If $(\Pi_q, q_{n,k})$ is an NDL-rewriting of $q_{n,k}$, we replace all the individual variables in $\Pi$ with II and then perform the replacement described above. Denote the resulting propositional NDL-program by $\Pi'$. The program $\Pi'$ can now be transformed into a monotone Boolean circuit computing $\text{CLIQUE}_{n,k}^i$: for every (propositional) variable $p$ occurring in the head of a clause in $\Pi'$, we introduce an $\lor$-gate whose output is $p$ and inputs are the bodies of the clauses with the head $p$; and for each such body, we introduce an $\land$-gate whose inputs are the propositional variables in the body.

Now Theorem 17 follows from the lower bounds for monotone Boolean circuits and formulas computing $\text{CLIQUE}_{n,k}^i$ given at the beginning of this section.

As the function $\text{CLIQUE}_{n,k}^i$ is known to be NP/poly-complete with respect to NC$^1$-reductions, we also obtain:

**Theorem 18.** There exist polynomial-size FO-rewritings for all CQs and ontologies of depth 2 with polynomially-many tree witnesses if all functions in NP/poly are computed by polynomial-size formulas, that is, if NP/poly $\subseteq$ NC$^1$.

### 7. Rewritings of Tree-Shaped CQs

A CQ is said to be tree-shaped if its Gaifman graph is a tree. It is well known [13, 37] that tree-shaped CQs (or, more generally, CQs of bounded treewidth) can be evaluated over plain data instances in polynomial time. In contrast, the evaluation of arbitrary CQs is NP-complete for combined complexity and W[1]-complete for parameterised complexity. In this section, we consider tree-shaped CQs over ontologies.

At first sight, we do not gain much by focusing on tree-shaped CQs: answering such CQs over ontologies is NP-complete for combined complexity [23], while their PE- and NDL-rewritings can suffer an exponential blowup [21]. However, by examining the tree-witness rewriting (4), we see that the twos formula (3) defines a predicate over the data that can be computed in linear time. It follows that, for a tree-shaped $q$, every disjunct of (4) can also be regarded as a tree-shaped CQ of size $|q|$. So, bearing in mind that $|q_j| \leq 3^{|q|}$, we obtain the following:

**Theorem 19.** Given a tree-shaped CQ $q(\bar{x})$, an ontology $\mathcal{T}$, a data instance $A$ and a tuple $\bar{a}$ from $\text{ind}(A)$, the problem of deciding whether $\mathcal{T}, A \models q(\bar{a})$ is fixed-parameter tractable, with parameter $|q|$.

Furthermore, if each variable in a tree-shaped CQ is covered by a ‘small’ number of tree witnesses then we can obtain polynomial-size PE- or NDL-rewritings.

**Example 20.** Consider the following ontology:

$$\mathcal{T} = \{ A_i(x) \rightarrow \exists y (R_i(x, y) \land R_{i+1}(y, x)) \mid 1 \leq i \leq 3 \}$$

and the following CQ:

$$q = \exists y_1 \ldots y_5 \bigwedge_{1 \leq i \leq 4} R_i(y_i, y_{i+1})$$

illustrated in the picture below:

[Diagram of a tree-shaped CQ with nodes $q_1, q_2, R_1, R_2, R_3, R_4$ and variables $y_1, y_2, y_3, y_4$]

We construct a PE-rewriting $q'$ of $q$ and $\mathcal{T}$ recursively by splitting $q$ into smaller subqueries. Suppose $(\mathcal{T}, A) \models q$, for some $A$. Then there is a homomorphism $h: q \rightarrow C_{\mathcal{T}, A}$. Consider the 'central' variable $y_3$, dividing $q$ in half. If $h(y_3)$ is in the data part of $C_{\mathcal{T}, A}$ then $y_3$ behaves like a free variable in $q$. Since $q$ is tree-shaped, we can then proceed by constructing PE-rewritings, $q'_1(y_3)$ and $q'_2(y_3)$, for the subqueries

$$q_1(y_3) = \exists y_1 y_2 (R_1(y_1, y_2) \land R_2(y_2, y_3)),
q_2(y_3) = \exists y_1 y_2 (R_3(y_3, y_4) \land R_4(y_4, y_5)).$$

If $h(y_3)$ is a labelled null, then $y_3$ must be an internal point of some tree witness for $q$ and $\mathcal{T}$. We have only one such tree witness, $t = (t_1, t_2)$, generated by $A_3(x)$ with $t_1 = \{y_2, y_3\}$, $t_2 = \{y_4\}$ and $q_1 = \{R_1(y_2, y_3), R_3(y_3, y_4)\}$ (shaded in the picture above). But then $h(y_3) = h(y_4)$ and this element is in the data part of $C_{\mathcal{T}, A}$. So, we need PE-rewritings, $q'_1(y_2)$ and $q'_2(y_4)$, of the remaining fragments of $q$:

$$q'_1(y_2) = \exists y_1 R_1(y_1, y_2),
q'_2(y_4) = \exists y_5 R_4(y_4, y_5).$$
If the required rewritings \( q^i \), \( 1 \leq i \leq 4 \), are constructed then we obtain a PE-rewriting \( q^1 \) of \( q \) and \( T \) by taking

\[
q^1 = \exists y_3 ( q^1_1(y_3) \land q^1_2(y_3)) \lor
\exists y_3(y_3) \land (y_2 = y_1) \land q^1_3(y_2) \land q^1_4(y_1)).
\]

We analyse \( q_1, q_2, q_3 \), and \( q_4 \) in the same way and obtain

\[
q^1_1(y_3) = \exists y_3(q_1^1(y_3) \land R_{A_1}(y_3, y_3)) \lor
\exists y_3(A_1(y_3) \land (y_3 = y_1)),
\]

\[
q^1_2(y_3) = \exists y_3(q_3^1(y_3, y_3)) \lor
\exists y_3(A_3(y_3) \land (y_3 = y_1)),
\]

\[
q^1_3(y_2) \text{ and } q^1_4(y_1) \text{ equal to } q_3(y_2) \text{ and } q_4(y_1), \text{ respectively.}
\]

We now give a general definition of a PE-rewriting obtained by the strategy ‘divide and rewrite’ and applicable to any (not necessarily tree-shaped) CQ. Let \( q(x) = \exists y \varphi(x, \bar{y}) \) and an ontology \( T \) be given. We recursively define a PE-query \( q^1(x) \) as follows. Take the finest partition of \( \exists y \varphi(x, \bar{y}) \) into a conjunction \( \bigwedge y \exists y \varphi_j(x, \bar{y}_j) \) such that every atom containing some \( \bar{y} \) belongs to the same conjunct \( \varphi_j \). (Informally, the Gaifman graph of \( \varphi \) is cut along the answer variables \( x \).) By definition, the set of tree witnesses for \( \exists y \varphi(x, \bar{y}) \) and \( T \) is the disjoint union of the sets of tree witnesses for \( \exists y \varphi_j(x, \bar{y}_j) \) and \( T \). Then we set

\[
(\exists y \varphi(x, \bar{y}))^1 = \bigwedge y \varphi_j \land tw_j,\]

where \( \varphi_j(x, \bar{y}_j) \) in case \( \bar{y}_j \) is empty; otherwise, we choose a variable \( z \) in \( \bar{y}_j \) and define \( \varphi_j \) to be the formula

\[
\exists z (\exists [\bar{y}_j \setminus \{z\}] \varphi_j(x, \bar{y}_j)) \land (\exists y \varphi_j(x, \bar{y}_j)) \land tw_j,\]

where \( \varphi_j \) is a PE-variable in \( \varphi_j(x, \bar{y}_j) \) and \( t \) is a tree witness for \( \exists y \varphi_j(x, \bar{y}_j) \) and \( T \) such that \( t \in \{t_k, t_l\} \) and \( z \in t_1 \).

Theorem 21. For any CQ \( q(x) \) and ontology \( T \), \( q^1(x) \) is a PE-rewriting of \( q \) and \( T \) (over complete data).

The exact form of the rewriting \( q^1 \) depends on the choice of the variables \( z \). We now consider two strategies for choosing these variables in the case of tree-shaped CQs. Let

\[
d_T^q = 1 + \max_{z \in \bar{g}} \{|t = (t_k, t_l) \in \Theta_T^q \mid z \in t_1\}\..
\]

We call \( d_T^q \) the tree-witness degree of \( q \) and \( T \). For example, the tree-witness degree of any CQ and ontology of depth 1 is at most 2, as observed in the proof of Theorem 8. In general, however, it can only be bounded by \( 1 + |\Theta_T^q| \).

Given a tree-shaped CQ \( q(x) = \exists y \varphi(x, \bar{y}) \), we pick some variable as its root and define a partial order \( \prec \) on the variables of \( q \) by taking \( z \prec z' \) if \( z' \) occurs in the subtree of \( q \) rooted in \( z \). The strategy used in [8] chooses the smallest \( z \) with respect to \( \preceq \).

Corollary 22. (8). Any tree-shaped CQ and any ontology with polynomially-many tree-witnesses have a polynomial-size NDL-rewriting.

As the depth of recursion in the rewriting process with the above strategy is bounded by \(|q|\), we can only obtain a PE-rewriting of exponential size in \(|q|\). However, if we adopt the strategy of choosing \( z \) that splits the graph of each \( \varphi \) in half, then the depth of recursion does not exceed \( \log |q| \), and so the resulting PE-rewriting is of polynomial size for \( q \) and \( T \) of bounded tree-witness degree. This strategy is based on the following fact:

Proposition 23. Any tree \( T = (V, E) \) contains a vertex \( v \in V \) such that each connected component obtained by removing \( v \) from \( T \) has at most \(|V|/2\) vertices.

As a consequence, we obtain:

Theorem 24. For any tree-shaped CQ \( q \) and any ontology \( T \), there is a PE-rewriting of size \(|T| \cdot |q|^{1+\log d_T^q} \) (over complete data).

Proof. Denote by \( F(n) \) the maximal size of \( p^j \) for a subquery \( p \) of the CQ \( q \) with at most \( n \) atoms. We show by induction on \( n \) that \( F(n) \leq |T| \cdot n^{1+\log d_T^q} \). By definition, for each component \( p_j \) of the finest partition of \( p \), the length of its contribution to \( p^j \) does not exceed

\[
F(n_j) + \sum_{i=1}^{d_T^q} (n_j - m_{ji}) + |T| \cdot m_{ji},
\]

where \( n_j \) is the number of atoms in \( p_j \), and \( m_{ji} \) is the number of atoms in the \( i \)th tree witness with \( z \in t_i \), \( 1 \leq m_{ji} \leq n_j \). By the induction hypothesis, the length of the contribution of \( p_j \) does not exceed

\[
|T| \cdot n_j^{1+\log d_T^q} + |T| \cdot \sum_{i=1}^{d_T^q} ((n_j - m_{ji})^{1+\log d_T^q} + m_{ji}) \leq
|T| \cdot (n_j^{1+\log d_T^q} + (d_T^q - 1) \cdot n_j^{1+\log d_T^q}) = |T| \cdot d \cdot n_j^{1+\log d_T^q}.
\]

By Proposition 23, we can choose \( z \) (at the preceding step) so that \( p \) with \( n \) atoms is split into components \( p_1, \ldots, p_k \) each of which has \( n_j \leq n/2 \) atoms (by definition, \( \sum j=1^k n_j = n \)). This gives

\[
F(n) \leq |T| \cdot d \cdot \sum_{j=1}^{k} (n/2)^{1+\log d_T^q} \leq |T| \cdot n^{1+\log d_T^q}
\]

as required.

Corollary 25. Any tree-shaped CQ \( q \) and ontology \( T \) of depth 1 have a PE-rewriting of size \(|T| \cdot |q|^2 \) (over complete data).

8. Conclusions

We have established a fundamental link between FO-rewritings of CQs over OWL 2 QL ontologies of depth 1 and 2 and—via the hypergraph functions and programs—classical computational models for Boolean functions. This link allowed us to apply the Boolean complexity theory and obtain both polynomial upper and exponential (or superpolynomial) lower bounds for the size of rewritings. It is to be noted that the high lower bounds were proved for CQs and ontologies with polynomially-many tree-witnesses and polynomial-size chases.

A few challenging important questions remain open:

(i) Are all hypergraphs representable as subgraphs of some tree-witness hypergraphs?
(ii) Do all tree-shaped CQs have polynomial-size rewritings over ontologies of depth 2 (more generally, of bounded depth)?

(iii) What is the size of CQ rewritings over a fixed ontology in the worst case?

(The last question is related to the non-uniform approach to the complexity of query answering in OBDA on the level of individual ontologies [27].)

Acknowledgments

This work was supported by the UK EPSRC project ‘ExODA: Integrating Description Logics and Database Technologies for Expressive Ontology-Based Data Access’ (EP/H05099X).

References


