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 $\subseteq$ 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 I.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*[1] 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

$$\forall \vec{x}\, \big(\varphi(\vec{x}) \to \exists \vec{y}\, \psi(\vec{x}, \vec{y})\big) \quad \text{or} \quad \forall \vec{x}\, \big(\varphi(\vec{x}) \land \varphi'(\vec{x}) \to \bot\big) \qquad (1)$$

where φ, φ' and ψ are unary or binary predicates (such sentences are known as linear tuple-generating dependencies—or tgds—of 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 $\mathcal{T}$ and a CQ $\boldsymbol{q}(\vec{x})$, one can construct a first-order (FO) formula $\boldsymbol{q}'(\vec{x})$ in the signature of $\boldsymbol{q}$ and $\mathcal{T}$ such that $\mathcal{T}, \mathcal{A} \models \boldsymbol{q}(\vec{a})$ iff $\mathcal{A} \models \boldsymbol{q}'(\vec{a})$, for any set $\mathcal{A}$ of ground atoms (data) and any tuple $\vec{a}$ of constants in $\mathcal{A}$. Thus, to find certain answers to $\boldsymbol{q}(\vec{x})$ over $(\mathcal{T}, \mathcal{A})$, we can first compute an *FO-rewriting* $\boldsymbol{q}'(\vec{x})$ of $\boldsymbol{q}$ and $\mathcal{T}$, and then evaluate it over any given data $\mathcal{A}$ using, for example, a relational database management system. The ontology $\mathcal{T}$ 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 tgds [7, 11].

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)? First answers to these questions were given in [21] which constructed a sequence of CQs $\boldsymbol{q}_n$ and ontologies $\mathcal{T}_n$, for $n = 1, 2, \dots$, with only exponential positive existential (PE) and nonrecursive datalog (NDL) rewritings, and superpolynomial FO-rewritings unless NP $\subseteq$ 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 chases (canonical models) for $\mathcal{T}_n$ contain full binary trees of depth n and give rise to *exponentially-many* homomorphisms from $\boldsymbol{q}_n$ to the trees of labelled nulls of the chases, all of which have to be reflected in the rewritings of $\boldsymbol{q}_n$ and $\mathcal{T}_n$.

In this paper, we investigate succinctness of CQ rewritings over 'shallow' ontologies whose (polynomial-size) chases 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 chases 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

[1] `www.w3.org/TR/owl2-profiles`

[2] See, e.g., QuOnto [30], Presto/Prexto [35, 36], Rapid [14], Ontop [34], Requiem/Blackout [28, 29], Nyaya [17], Clipper [15] and PURE [25].

CSL-LICS 2014, July 14–18, 2014, Vienna, Austria.
Copyright © 2014 ACM 978-1-4503-2886-9…$15.00.
http://dx.doi.org/10.1145/2603088.2603131

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 NDL-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 '$\mathrm{NL/poly} \subseteq \mathrm{NC}^1$?' (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 $\mathrm{NP} \subseteq \mathrm{P/poly}$ (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 '$\mathrm{NP/poly} \subseteq \mathrm{NC}^1$?' (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 chases for the given ontology. In particular, hypergraphs H of degree 2 (every vertex in which belongs to 2 hyperedges) correspond to CQs $\boldsymbol{q}$ and ontologies $\mathcal{T}$ of depth 1 such that answering $\boldsymbol{q}$ over $\mathcal{T}$ 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 $\textsc{Clique}_{n,k}(\vec{e})$ (graph $\vec{e}$ with n vertices has a k-clique) as CQs over ontologies of depth 2. Finally, we show that any tree-shaped CQ $\boldsymbol{q}$ and ontology $\mathcal{T}$ have a PE-rewriting of size $O(|\mathcal{T}|^2 \cdot |\boldsymbol{q}|^{1+\log d})$, 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 '$\mathcal{T}, \mathcal{A} \models \boldsymbol{q}$?', for tree-shaped Boolean CQs and any $\mathcal{T}$, is fixed-parameter tractable, with parameter $|\boldsymbol{q}|$ (recall that the problem '$\mathcal{A} \models \boldsymbol{q}$?', for tree-shaped $\boldsymbol{q}$, is known to be tractable [37], while '$\mathcal{T}, \mathcal{A} \models \boldsymbol{q}$?' 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 $\boldsymbol{q}$, $\mathcal{T}$ and $\mathcal{A}$, the extra quantifiers guess a homomorphism from $\boldsymbol{q}$ to the chase for $(\mathcal{T}, \mathcal{A})$, 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*, $\mathcal{T}$, is a finite set of *tuple-generating dependencies* (*tgds*) of the form

$$\forall \vec{x} \, \big(\varphi(\vec{x}) \to \exists \vec{y} \bigwedge \psi_i(\vec{x}, \vec{y})\big), \qquad (2)$$

where φ and the ψ_i are unary or binary atoms without constants and $|\vec{x} \cup \vec{y}| \leq 2$. These tgds are expressible 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, $|\mathcal{T}|$, of $\mathcal{T}$ is the number of predicate occurrences in $\mathcal{T}$. A *data instance*, $\mathcal{A}$, is a finite set of ground atoms. The set of individual constants in $\mathcal{A}$ is denoted by $\mathsf{ind}(\mathcal{A})$. Taken together, $\mathcal{T}$ and $\mathcal{A}$ form the *knowledge base* (KB) $(\mathcal{T}, \mathcal{A})$. 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 $\mathsf{ind}(\mathcal{A})$, if $\mathcal{T}, \mathcal{A} \models S(\vec{a})$ then $S(\vec{a}) \in \mathcal{A}$ (see Remark 1 below).

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

Given a CQ $\boldsymbol{q}(\vec{x})$ and an ontology $\mathcal{T}$, an FO-formula $\boldsymbol{q}'(\vec{x})$ *without constants* is called an *FO-rewriting of* $\boldsymbol{q}(\vec{x})$ *and* $\mathcal{T}$ if, for any (complete) data instance $\mathcal{A}$ and any $\vec{a}$ from $\mathsf{ind}(\mathcal{A})$, we have $(\mathcal{T}, \mathcal{A}) \models \boldsymbol{q}(\vec{a})$ iff $\mathcal{A} \models \boldsymbol{q}'(\vec{a})$.[3] If $\boldsymbol{q}'$ is a positive existential formula, we call it a *PE-rewriting of* $\boldsymbol{q}$ *and* $\mathcal{T}$. 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 $\forall \vec{x}\, (\gamma_1 \wedge \cdots \wedge \gamma_m \to \gamma_0)$, where each γ_i is an atom of the form $P(x_1, \ldots, x_l)$ with $x_i \in \vec{x}$. The atom γ_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 $\boldsymbol{q}'(\vec{x})$, $(\Pi, \boldsymbol{q}')$ is called an *NDL-rewriting of* $\boldsymbol{q}(\vec{x})$ *and* $\mathcal{T}$ in case $\mathcal{T}, \mathcal{A} \models \boldsymbol{q}(\vec{a})$ iff $\Pi, \mathcal{A} \models \boldsymbol{q}'(\vec{a})$, for any (complete) $\mathcal{A}$ and tuple $\vec{a}$ from $\mathsf{ind}(\mathcal{A})$. 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 $(\Pi, \boldsymbol{q}')$ of $\boldsymbol{q}$ and $\mathcal{T}$ over complete data, there is an NDL-rewriting $(\Pi', \boldsymbol{q}')$ over arbitrary data with $|\Pi'| \leq |\Pi| + O(|\mathcal{T}|)$. Similarly, for a PE-rewriting $\boldsymbol{q}'$ of $\boldsymbol{q}$ and $\mathcal{T}$ over complete data, there is a PE-rewriting $\boldsymbol{q}''$ over arbitrary data with $|\boldsymbol{q}''| \leq O(|\boldsymbol{q}'| \cdot |\mathcal{T}|)$.

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 $(\mathcal{T}, \mathcal{A})$, there is a *canonical model* (or *chase*) $\mathcal{C}_{\mathcal{T},\mathcal{A}}$ such that $\mathcal{T}, \mathcal{A} \models \boldsymbol{q}(\vec{a})$ iff

[3] Thus, we do not allow the rewriting from [18] as it contains constants.

$\mathcal{C}_{\mathcal{T},\mathcal{A}} \models \boldsymbol{q}(\vec{a})$, for all CQs $\boldsymbol{q}(\vec{x})$ and $\vec{a}$ from $\mathsf{ind}(\mathcal{A})$. The domain of $\mathcal{C}_{\mathcal{T},\mathcal{A}}$ consists of $\mathsf{ind}(\mathcal{A})$ and the witnesses, or *labelled nulls*, introduced by the existential quantifiers in $\mathcal{T}$.

For any formula $\varrho(x)$ of the form $S(x)$, $S(x,x)$, $\exists y\, S(x,y)$ or $\exists y\, S(y,x)$, where S is a predicate in $\mathcal{T}$, we denote by $\mathcal{C}_{\mathcal{T}}^{\varrho(a)}$ the canonical model of the KB $(\mathcal{T} \cup \{A(x) \to \varrho(x)\}, \{A(a)\})$, where A is a fresh unary predicate and a a fresh constant (note that such a canonical model is independent of the data instance $\mathcal{A}$). We say that $\mathcal{T}$ is *of depth* k, $1 \le k < \omega$, if at least one of the $\mathcal{C}_{\mathcal{T}}^{\varrho(a)}$ contains a chain of the form $R_0(w_0,w_1)\dots R_{k-1}(w_{k-1},w_k)$, with not necessarily distinct w_i, but none of the $\mathcal{C}_{\mathcal{T}}^{\varrho(a)}$ has such a chain of greater length. For example, the ontology $\mathcal{T} = \{A(x) \to \exists y\, P(x,y)\}$ is of depth 1, the ontology $\mathcal{T} \cup \{P(x,y) \to \exists z\, S(y,z)\}$ is of depth 2, whereas $\mathcal{T}' = \mathcal{T} \cup \{P(x,y) \to \exists z\, P(y,z)\}$ is of infinite depth because $\mathcal{C}_{\mathcal{T}'}^{\exists y P(a,y)}$ contains an infinite chain of the form $P(a,w_1)P(w_1,w_2)\dots$.

Suppose we are given a CQ $\boldsymbol{q}(\vec{x}) = \exists \vec{y}\, \varphi(\vec{x},\vec{y})$ and an ontology $\mathcal{T}$. For a pair $\mathfrak{t} = (\mathfrak{t}_\mathsf{r}, \mathfrak{t}_\mathsf{i})$ of disjoint sets of variables in $\boldsymbol{q}$, with $\mathfrak{t}_\mathsf{i} \subseteq \vec{y}$ and $\mathfrak{t}_\mathsf{i} \neq \emptyset$ ($\mathfrak{t}_\mathsf{r}$ can be empty), set

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

We call $\mathfrak{t} = (\mathfrak{t}_\mathsf{r}, \mathfrak{t}_\mathsf{i})$ a *tree witness for* $\boldsymbol{q}$ *and* $\mathcal{T}$ *generated by* ϱ if $\boldsymbol{q}_\mathfrak{t}$ is a minimal subset of $\boldsymbol{q}$ for which there is a homomorphism $h\colon \boldsymbol{q}_\mathfrak{t} \to \mathcal{C}_{\mathcal{T}}^{\varrho(a)}$ such that $\mathfrak{t}_\mathsf{r} = h^{-1}(a)$ and $\boldsymbol{q}_\mathfrak{t}$ contains all atoms of $\boldsymbol{q}$ with at least one variable from $\mathfrak{t}_\mathsf{i}$. Note that the same tree witness $\mathfrak{t} = (\mathfrak{t}_\mathsf{r}, \mathfrak{t}_\mathsf{i})$ can be generated by different ϱ. Now, we set

$$\mathsf{tw}_\mathfrak{t}(\mathfrak{t}_\mathsf{r}) \;=\; \bigvee_{\mathfrak{t} \text{ generated by } \varrho} \exists z\, \big(\varrho(z) \;\wedge\; \bigwedge_{x \in \mathfrak{t}_\mathsf{r}} (x = z)\big). \tag{3}$$

The variables in $\mathfrak{t}_\mathsf{i}$ do not occur in $\mathsf{tw}_\mathfrak{t}$ and are called *internal*. The length, $|\mathsf{tw}_\mathfrak{t}|$, of $\mathsf{tw}_\mathfrak{t}$ is $O(|\boldsymbol{q}| \cdot |\mathcal{T}|)$. Tree witnesses $\mathfrak{t}$ and $\mathfrak{t}'$ are *conflicting* if $\boldsymbol{q}_\mathfrak{t} \cap \boldsymbol{q}_{\mathfrak{t}'} \neq \emptyset$. Denote by $\Theta_{\mathcal{T}}^{\boldsymbol{q}}$ the set of tree witnesses for $\boldsymbol{q}$ and $\mathcal{T}$. A subset $\Theta \subseteq \Theta_{\mathcal{T}}^{\boldsymbol{q}}$ is *independent* if no pair of distinct tree witnesses in it is conflicting. Let $\boldsymbol{q}_\Theta = \bigcup_{\mathfrak{t} \in \Theta} \boldsymbol{q}_\mathfrak{t}$. The following PE-formula $\boldsymbol{q}_\mathsf{tw}$ is called the *tree-witness rewriting of* $\boldsymbol{q}$ *and* $\mathcal{T}$:

$$\boldsymbol{q}_\mathsf{tw}(\vec{x}) \;=\; \bigvee_{\Theta \subseteq \Theta_{\mathcal{T}}^{\boldsymbol{q}} \text{ independent}} \exists \vec{y}\, \big(\bigwedge_{S(\vec{z}) \in \boldsymbol{q} \setminus \boldsymbol{q}_\Theta} S(\vec{z}) \;\wedge\; \bigwedge_{\mathfrak{t} \in \Theta} \mathsf{tw}_\mathfrak{t}(\mathfrak{t}_\mathsf{r}) \big). \tag{4}$$

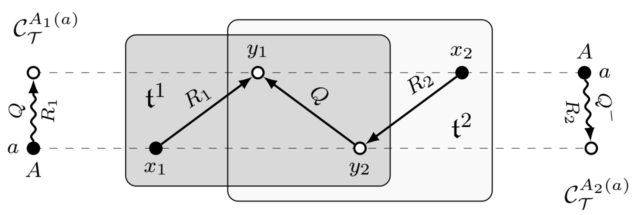
Example 2. Consider an ontology $\mathcal{T}$ with the tgds

$$A_1(x) \to \exists y\, \big(R_1(x,y) \wedge Q(x,y)\big),$$
$$A_2(x) \to \exists y\, \big(R_2(x,y) \wedge Q(y,x)\big)$$

and the CQ

$$\boldsymbol{q}(x_1,x_2) = \exists y_1 y_2\, \big(R_1(x_1,y_1) \wedge Q(y_2,y_1) \wedge R_2(x_2,y_2)\big).$$

The CQ $\boldsymbol{q}$ is shown below alongside $\mathcal{C}_{\mathcal{T}}^{A_1(a)}$ and $\mathcal{C}_{\mathcal{T}}^{A_2(a)}$:



There are two tree witnesses, $\mathfrak{t}^1$ and $\mathfrak{t}^2$, for $\boldsymbol{q}$ and $\mathcal{T}$ with

$$\boldsymbol{q}_{\mathfrak{t}^1} = \{R_1(x_1,y_1), Q(y_2,y_1)\}, \quad \boldsymbol{q}_{\mathfrak{t}^2} = \{Q(y_2,y_1), R_2(x_2,y_2)\}$$

(shown above by the dark and light shading, respectively). The tree witness $\mathfrak{t}^1 = (\mathfrak{t}^1_\mathsf{r}, \mathfrak{t}^1_\mathsf{i})$ with $\mathfrak{t}^1_\mathsf{r} = \{x_1, y_2\}$ and $\mathfrak{t}^1_\mathsf{i} = \{y_1\}$ is generated by $A_1(x)$, which gives

$$\mathsf{tw}_{\mathfrak{t}^1}(x_1,y_2) = \exists z\, \big(A_1(z) \wedge (x_1 = z) \wedge (y_2 = z)\big).$$

Symmetrically, the tree witness $\mathfrak{t}^2$ gives

$$\mathsf{tw}_{\mathfrak{t}^2}(x_2,y_1) = \exists z\, \big(A_2(z) \wedge (x_2 = z) \wedge (y_1 = z)\big).$$

As $\mathfrak{t}^1$ and $\mathfrak{t}^2$ are conflicting, $\Theta_{\mathcal{T}}^{\boldsymbol{q}}$ contains three independent subsets: $\emptyset$, $\{\mathfrak{t}^1\}$ and $\{\mathfrak{t}^2\}$. Thus, we obtain the following rewriting:

$$\exists y_1 y_2\, \big[(R_1(x_1,y_1) \wedge Q(y_2,y_1) \wedge R_2(x_2,y_2)) \vee (R_2(x_2,y_2) \wedge \mathsf{tw}_{\mathfrak{t}^1}) \vee (R_1(x_1,y_1) \wedge \mathsf{tw}_{\mathfrak{t}^2})\big].$$

Theorem 3 ([24])**.** *For any complete data instance $\mathcal{A}$ and any tuple $\vec{a}$ from $\mathsf{ind}(\mathcal{A})$, we have $\mathcal{T},\mathcal{A} \models \boldsymbol{q}(\vec{a})$ iff $\mathcal{A} \models \boldsymbol{q}_\mathsf{tw}(\vec{a})$.*

The number of tree witnesses, $|\Theta_{\mathcal{T}}^{\boldsymbol{q}}|$, is bounded by $3^{|\boldsymbol{q}|}$. On the other hand, there is a sequence of queries $\boldsymbol{q}_n$ and ontologies $\mathcal{T}_n$ with exponentially many (in $|\boldsymbol{q}_n|$) tree witnesses [24]. The length of $\boldsymbol{q}_\mathsf{tw}$ is $O(2^{|\Theta_{\mathcal{T}}^{\boldsymbol{q}}|} \cdot |\boldsymbol{q}| \cdot |\mathcal{T}|)$. If any two tree-witnesses for $\boldsymbol{q}$ and $\mathcal{T}$ are *compatible*—that is, they are either non-conflicting or one is included in the other—then $\boldsymbol{q}_\mathsf{tw}$ can be equivalently transformed into the PE-rewriting

$$\boldsymbol{q}'_\mathsf{tw}(\vec{x}) \;=\; \exists \vec{y} \bigwedge_{S(\vec{z}) \in \boldsymbol{q}} \big(S(\vec{z}) \;\vee \bigvee_{\mathfrak{t} \in \Theta_{\mathcal{T}}^{\boldsymbol{q}} \text{ with } S(\vec{z}) \in \boldsymbol{q}_\mathfrak{t}} \mathsf{tw}_\mathfrak{t}(\mathfrak{t}_\mathsf{r}) \big)$$

of size $O(|\Theta_{\mathcal{T}}^{\boldsymbol{q}}| \cdot |\boldsymbol{q}|^2 \cdot |\mathcal{T}|)$. Our aim now is to investigate transformations of this kind in the more abstract setting of Boolean functions. In Section 5, we shall see an example of $\boldsymbol{q}$ and $\mathcal{T}$ with only $|\boldsymbol{q}|$-many tree witnesses any PE-rewriting of which is of superpolynomial size because of multiple combinations of incompatible tree witnesses.

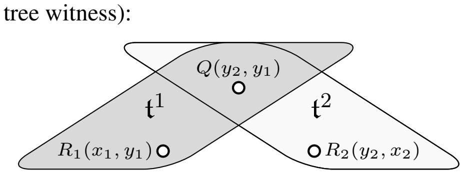
3. Hypergraph Functions

The tree-witness rewriting $\boldsymbol{q}_\mathsf{tw}$ gives rise to monotone Boolean functions we call hypergraph functions. Let $H = (V,E)$ be a hypergraph with *vertices* $v \in V$ and *hyperedges* $e \in E$, $E \subseteq 2^V$. A subset $X \subseteq E$ is *independent* if $e \cap e' = \emptyset$, for any distinct $e, e' \in X$. Denote by V_X the set of vertices occurring in the hyperedges of X. With each $v \in V$ and $e \in E$ we associate propositional variables p_v and p_e, respectively. The *hypergraph function* f_H for H is given by the Boolean formula

$$f_H \;=\; \bigvee_{X \subseteq E \text{ independent}} \Big(\bigwedge_{v \in V \setminus V_X} p_v \;\wedge\; \bigwedge_{e \in X} p_e \Big). \tag{5}$$

By the definition of $\boldsymbol{q}_\mathsf{tw}$, every pair $\boldsymbol{q}$ and $\mathcal{T}$ gives rise to a hypergraph $H_{\mathcal{T}}^{\boldsymbol{q}}$ whose vertices are the atoms of $\boldsymbol{q}$ and hyperedges are the sets $\boldsymbol{q}_\mathfrak{t}$, for $\mathfrak{t} \in \Theta_{\mathcal{T}}^{\boldsymbol{q}}$: formula (5) for $H_{\mathcal{T}}^{\boldsymbol{q}}$ is the same as rewriting (4) for $\boldsymbol{q}$ and $\mathcal{T}$ with the atoms $S(\vec{z}) \in \boldsymbol{q}$ and the tree witness formulas $\mathsf{tw}_\mathfrak{t}$ treated as propositional variables, $p_{S(\vec{z})}$ and $p_\mathfrak{t}$, respectively.

Example 4. For $\boldsymbol{q}$ and $\mathcal{T}$ from Example 2, the hypergraph $H_{\mathcal{T}}^{\boldsymbol{q}}$ has 3 vertices (one for each atom in the query) and 2 hyperedges (one for each tree witness):



The hypergraph function of $H_{\mathcal{T}}^{\boldsymbol{q}}$ is as follows:

$$f_{H_{\mathcal{T}}^{\boldsymbol{q}}} = (p_{R_1(x_1,y_1)} \wedge p_{Q(y_2,y_1)} \wedge p_{R_2(x_2,y_2)}) \vee (p_{R_2(x_2,y_2)} \wedge p_{\mathfrak{t}^1}) \vee (p_{R_1(x_1,y_1)} \wedge p_{\mathfrak{t}^2}).$$

Suppose now that the function $f_{H_{\mathcal{T}}^{\boldsymbol{q}}}$ is computed by a Boolean formula χ. By comparing (5) and (4), it is readily seen that the result of prefixing $\exists \vec{y}$ to χ and replacing each $p_{S(\vec{z})}$ in it with $S(\vec{z})$ and each $p_\mathfrak{t}$ with the formula $\mathsf{tw}_\mathfrak{t}(\mathfrak{t}_\mathsf{r})$ of length $O(|\boldsymbol{q}| \cdot |\mathcal{T}|)$ is a rewriting of $\boldsymbol{q}$ and $\mathcal{T}$. 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 $f_{H^{\boldsymbol{q}}_{\mathcal{T}}}$ is computed by some (monotone) Boolean formula χ then there exists a (PE-) FO-rewriting of $\boldsymbol{q}$ and $\mathcal{T}$ of size $O(|\chi| \cdot |\boldsymbol{q}| \cdot |\mathcal{T}|)$.*

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

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 (NBPs) and nondeterministic Boolean circuits [3, 19].

4. Hypergraphs, NBPs and Boolean Circuits

Let $p_1, \dots, p_n$ 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) 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 $\mathrm{HGP}(f)$ ($\mathrm{HGP}^n(f)$) the minimal size of hypergraph programs (of degree $\leq n$) computing f; $\mathrm{HGP}_+(f)$ and $\mathrm{HGP}^n_+(f)$ are used for the size of monotone programs.

Our first result in this section establishes a link between hypergraph programs of degree ≤ 2 and NBPs. Note [22] that any (monotone) hypergraph program H of degree ≤ 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 NBPs may have multiple parallel arcs (with distinct labels) connecting two nodes. We write $v \to_{\vec{\alpha}} v'$ 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 $s \to_{\vec{\alpha}} t$. 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 $\mathrm{NBP}(f)$ (respectively, $\mathrm{NBP}_+(f)$) the minimal size of (monotone) NBPs computing f. As usual, f^* is the Boolean function dual to f.

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

(*ii*) *For any monotone Boolean function f, $\mathrm{HGP}^2_+(f)$ and $\mathrm{NBP}_+(f^*)$ 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 ≤ 2 as follows. For each arc e in G, H has two vertices e^0 and e^1, which represent the beginning and the end of e. The vertex e^0 is labelled with the *negated* label of e in G and e^1 with 1. We also add to H a vertex t labelled with 0. For each arc e in G, H has an *e-hyperedge* $\{e^0, e^1\}$. For each vertex v in G but s and t, H has a *v-hyperedge* that consists of all vertices e^1, for the arcs e leading to v, and all vertices e^0, 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 $s \not\to_{\vec{\alpha}} t$ 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 $s \not\to_{\vec{\alpha}} v$. Conversely, if $s \to_{\vec{\alpha}} t$ 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 $e_1, \dots, e_k$. 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_{e_i} and with an input $\vec{\alpha}$ we associate the following set $\Phi_{\vec{\alpha}}$ of binary clauses:

- $\neg p_{e_i} \lor \neg p_{e_j}$, if $e_i \cap e_j \neq \emptyset$ (informally: intersecting hyperedges cannot be chosen at the same time),
- $p_{e_i} \lor p_{e_j}$, 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 $X = \{e_i \mid \vec{\beta}(p_{e_i}) = 1\}$, for some assignment $\vec{\beta}$ satisfying $\Phi_{\vec{\alpha}}$. Let $B_{\vec{\alpha}}$ be a directed graph $(V, E_{\vec{\alpha}})$ with

$$\begin{aligned} V &= \{e_i^+, e_i^- \mid 1 \leq i \leq k\}, \\ E_{\vec{\alpha}} &= \{(e_i^+, e_j^-) \mid e_i \cap e_j \neq \emptyset\} \cup \\ &\qquad \{(e_i^-, e_j^+) \mid v \in e_i \cap e_j \text{ and } \vec{\alpha}(v) = 0\}. \end{aligned}$$

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 from (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 $e_i^- \to_{\vec{\alpha}} e_i^+$ and $e_i^+ \to_{\vec{\alpha}} e_i^-$ 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 with arcs from 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 $s \to_{\vec{\alpha}} t$ iff there is e_i such that $B_{\vec{\alpha}}$ contains a cycle going through e_i^+ and e_i^-. ❑

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

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

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

(*ii*) *For any monotone Boolean function* f, $\mathrm{HGP}_+(f)$, $\mathrm{HGP}_+^3(f)$ *and* $\mathrm{NBC}_+(f)$ *are polynomially related.*

Proof. It is easy to see that any function f computable by a (monotone) HGP H can also be computed by a (monotone) nondeterministic circuit of size $\mathrm{poly}(|H|)$. Conversely, suppose f is computed by a nondeterministic circuit $\mathbf{C}(\vec{x},\vec{y})$. Let $g_1,\dots,g_n$ be the nodes of $\mathbf{C}$ (including the inputs $\vec{x}$ and $\vec{y}$). We construct an HGP of degree ≤ 3 computing f by taking, for each i, a vertex g_i labelled with 0 and a pair of hyperedges $\bar{e}_{g_i}$ and e_{g_i}, both containing g_i. No other edge contains g_i, and so either $\bar{e}_{g_i}$ 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, $\bar{e}_{g_i}$ is there.) To ensure this property, for each input variable x_i, we add a vertex labelled with $\neg x_i$ to e_{x_i} and a fresh vertex labelled with x_i to $\bar{e}_{x_i}$. 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 $\bar{e}_{g_j}$, and a vertex labelled with 1 to $\bar{e}_{g_i}$ and e_{g_j}.
– If $g_i = g_j \lor g_{j'}$ then we add a vertex labelled with 1 to e_{g_j} and $\bar{e}_{g_i}$, add a vertex labelled with 1 to $e_{g_{j'}}$ and $\bar{e}_{g_i}$; then, we add vertices h_j and $h_{j'}$ labelled with 1 to $\bar{e}_{g_j}$ and $\bar{e}_{g_{j'}}$, respectively, and a vertex u_i labeled with 0 to $\bar{e}_{g_i}$; finally, we add hyperedges $\{h_j, u_i\}$ 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 iff it contains $\bar{e}_{g_j}$ in the first case, and iff it contains at least one of e_{g_j} and $e_{g_{j'}}$ in the second case. Finally, we add a vertex labelled with 0 to e_g for the output gate g of $\mathbf{C}$. By induction on the structure of $\mathbf{C}$ one can show that, for each $\vec{\alpha}$, there is $\vec{\beta}$ such that $\mathbf{C}(\vec{\alpha},\vec{\beta}) = 1$ iff the constructed hypergraph program returns 1 on $\vec{\alpha}$.

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

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 $\mathcal{T}$ of depth 1 and any CQ $\boldsymbol{q}$, the hypergraph $H^{\boldsymbol{q}}_{\mathcal{T}}$ is of degree ≤ 2 and $|\Theta^{\boldsymbol{q}}_{\mathcal{T}}| \leq |\boldsymbol{q}|$.*

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

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

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

We show next that any hypergraph H of degree 2 is representable by means of a CQ $\boldsymbol{q}_H$ and an ontology $\mathcal{T}_H$ of depth 1 in the sense that H is isomorphic to $H^{\boldsymbol{q}_H}_{\mathcal{T}_H}$ ($H \cong H^{\boldsymbol{q}_H}_{\mathcal{T}_H}$, in symbols). We can assume that $H = (V,E)$ comes with two fixed maps $i_1, i_2\colon V \to E$ such that $i_1(v) \neq i_2(v)$, $v \in i_1(v)$ and $v \in i_2(v)$, for any $v \in V$. For each hyperedge $e \in E$, we take an individual variable z_e and denote by $\vec{z}$ the vector of all such variables. For every vertex $v \in V$, we take a binary predicate R_v and set

$$\boldsymbol{q}_H \;=\; \exists \vec{z} \bigwedge_{v \in V} R_v(z_{i_1(v)}, z_{i_2(v)}).$$

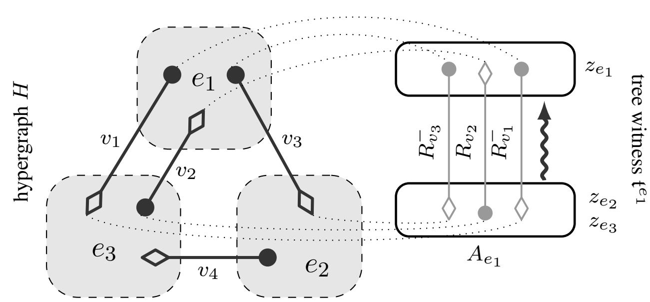
Let $\mathcal{T}_H$ be an ontology with the following tgds, for $e \in E$:

$$A_e(x) \;\to\; \exists y \Big[\bigwedge_{\substack{v \in V\\ i_1(v) = e}} R_v(y,x) \;\land\; \bigwedge_{\substack{v \in V\\ i_2(v) = e}} R_v(x,y) \Big]. \qquad (6)$$

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 $e_1 = \{v_1,v_2,v_3\}$, $e_2 = \{v_3,v_4\}$, $e_3 = \{v_1,v_2,v_4\}$, and assume that

$$i_1\colon v_1 \mapsto e_1,\;\; v_2 \mapsto e_3,\;\; v_3 \mapsto e_1,\;\; v_4 \mapsto e_2,$$
$$i_2\colon v_1 \mapsto e_3,\;\; v_2 \mapsto e_1,\;\; v_3 \mapsto e_2,\;\; v_4 \mapsto e_3.$$

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



In this case,

$$\boldsymbol{q}_H = \exists z_{e_1} z_{e_2} z_{e_3} \big(R_{v_1}(z_{e_1}, z_{e_3}) \land R_{v_2}(z_{e_3}, z_{e_1}) \land R_{v_3}(z_{e_1}, z_{e_2}) \land R_{v_4}(z_{e_2}, z_{e_3}) \big)$$

and the ontology $\mathcal{T}_H$ consists of the following tgds:

$$A_{e_1}(x) \to \exists y \big[R_{v_1}(y,x) \land R_{v_2}(x,y) \land R_{v_3}(y,x) \big],$$
$$A_{e_2}(x) \to \exists y \big[R_{v_3}(x,y) \land R_{v_4}(y,x) \big],$$
$$A_{e_3}(x) \to \exists y \big[R_{v_1}(x,y) \land R_{v_2}(y,x) \land R_{v_4}(x,y) \big].$$

The model $\mathcal{C}^{A_{e_1}(a)}_{\mathcal{T}_H}$ is shown on the right-hand side of the picture above. Note that each z_e determines the tree witness $\mathfrak{t}^e$ in which $\boldsymbol{q}_{\mathfrak{t}^e} = \{ R_v(z_{i_1(v)}, z_{i_2(v)}) \mid v \in e\}$; $\mathfrak{t}^e$ and $\mathfrak{t}^{e'}$ are conflicting iff

$e \cap e' \neq \emptyset$. It follows that H is isomorphic to $H^{\boldsymbol{q}_H}_{\mathcal{T}_H}$. In fact, this example generalises to the following:

Theorem 11. *Any hypergraph H of degree 2 is isomorphic to $H^{\boldsymbol{q}_H}_{\mathcal{T}_H}$, with $\mathcal{T}_H$ being an ontology of depth 1.*

We now show that answering $\boldsymbol{q}_H$ over $\mathcal{T}_H$ 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 $V = \{v_1, \dots, v_n\}$ and $E = \{e_1, \dots, e_m\}$. We denote by $\vec{\alpha}(v_i)$ the i-th component of $\vec{\alpha} \in \{0,1\}^n$, by $\vec{\beta}(e_j)$ the j-th component of $\vec{\beta} \in \{0,1\}^m$, and set

$$\mathcal{A}_{\vec{\alpha},\vec{\beta}} \;=\; \{\, R_{v_i}(a,a) \mid \vec{\alpha}(v_i) = 1 \,\} \;\cup\; \{\, A_{e_j}(a) \mid \vec{\beta}(e_j) = 1 \,\}.$$

Theorem 12. *Let $H = (V, E)$ be a hypergraph of degree 2. Then $\mathcal{T}_H, \mathcal{A}_{\vec{\alpha},\vec{\beta}} \models \boldsymbol{q}_H$ iff $f_H(\vec{\alpha}, \vec{\beta}) = 1$, for any $\vec{\alpha} \in \{0,1\}^{|V|}$ and $\vec{\beta} \in \{0,1\}^{|E|}$.*

Proof. ($\Leftarrow$) Let X be an independent subset of E such that $\bigwedge_{v \in V \setminus V_X} p_v \wedge \bigwedge_{e \in X} p_e$ is true on $\vec{\alpha}$ (for the p_v) and $\vec{\beta}$ (for the p_e). Define $h \colon \boldsymbol{q}_H \to \mathcal{C}_{\mathcal{T}_H, \mathcal{A}_{\vec{\alpha},\vec{\beta}}}$ 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 $\mathcal{C}_{\mathcal{T}_H, \mathcal{A}_{\vec{\alpha},\vec{\beta}}}$ introduced to witness the existential quantifier in (6). One can check that h is a homomorphism, and so $\mathcal{T}_H, \mathcal{A}_{\vec{\alpha},\vec{\beta}} \models \boldsymbol{q}_H$.

($\Rightarrow$) Suppose $h \colon \boldsymbol{q}_H \to \mathcal{C}_{\mathcal{T}_H, \mathcal{A}_{\vec{\alpha},\vec{\beta}}}$ 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 $v \in e \cap e'$, 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 $f_H(\vec{\alpha}, \vec{\beta}) = 1$. Indeed, for each $v \in V \setminus V_X$, h sends both ends of the R_v-atom to a, and so $\vec{\alpha}(v) = 1$. For each $e \in X$, we must have $h(z_e) = w_e$ because $h(z_e) \neq a$, and so $\vec{\beta}(e) = 1$. It follows that $f_H(\vec{\alpha}, \vec{\beta}) = 1$. ❑

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 $\boldsymbol{q}_n$ and ontologies $\mathcal{T}_n$ of depth 1, both of polynomial size in n, such that any PE-rewriting of $\boldsymbol{q}_n$ and $\mathcal{T}_n$ is 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 $\boldsymbol{q}_n$ and $\mathcal{T}_n$ such that $H_n \cong H^{\boldsymbol{q}_n}_{\mathcal{T}_n}$. We show now that any PE-rewriting $\boldsymbol{q}'_n$ of $\boldsymbol{q}_n$ and $\mathcal{T}_n$ can be transformed to a monotone Boolean formula computing f_n and having size $\leq |\boldsymbol{q}'_n|$.

To define it, we eliminate the quantifiers in $\boldsymbol{q}'_n$ in the following way: take a constant a and replace every subformula of the form $\exists x\, \psi(x)$ in $\boldsymbol{q}'_n$ with $\psi(a)$, repeating this operation as many times as possible. The resulting formula $\boldsymbol{q}''_n$ is built from atoms of the form $A_e(a)$, $R_v(a,a)$ and $S_e(a,a)$ using $\wedge$ and $\vee$. For every data instance $\mathcal{A}$ with a single individual a, we have $\mathcal{T}_n, \mathcal{A} \models \boldsymbol{q}_n$ iff $\mathcal{A} \models \boldsymbol{q}''_n$. Let χ_n be the result of replacing $S_e(a,a)$ in $\boldsymbol{q}''_n$ with $\bot$, $A_e(a)$ with p_e and $R_v(a,a)$ with p_v. Clearly, $|\chi_n| \leq |\boldsymbol{q}''_n|$. By the definition of $\mathcal{A}_{\vec{\alpha},\vec{\beta}}$ and Theorem 12, we have

$$\chi_n(\vec{\alpha}, \vec{\beta}) = 1 \quad \text{iff} \quad \mathcal{A}_{\vec{\alpha},\vec{\beta}} \models \boldsymbol{q}''_n \quad \text{iff} \quad \mathcal{T}_n, \mathcal{A}_{\vec{\alpha},\vec{\beta}} \models \boldsymbol{q}_n \quad \text{iff} \quad f_{H_n}(\vec{\alpha}, \vec{\beta}) = 1.$$

As H'_n computes f^*_n, we can obtain f^*_n from f_{H_n} by replacing each p_e with 1 and each p_v with the label of v in H'_n. The same substitution in χ_n (with $\top$ and $\bot$ in place of 1 and 0) gives a monotone formula that computes f^*_n. By swapping $\vee$ and $\wedge$ in it, we obtain a monotone formula χ'_n computing f_n. It remains to recall that $|\boldsymbol{q}'_n| \geq |\chi'_n| = n^{\Omega(\log n)}$. ❑

It may be of interest to note that the function f_n in the proof above is in the complexity class L. The algorithm computing f_n by querying the NDL-rewriting of Theorem 9 over single-individual data instances runs in polynomial time; the algorithm querying any PE-rewriting to compute f_n requires, by Theorem 13, superpolynomial time.

As reachability in directed graphs is NL/poly-complete under NC^1-reductions and NL $=$ coNL, the argument in the proof of Theorem 13 shows that the existence of short FO-rewritings of CQs and ontologies of depth 1 is equivalent to a well-known open problem in computational complexity:

Theorem 14. *There exist polynomial-size FO-rewritings for all CQs and ontologies of depth 1 iff all functions in the class* NL/*poly are computed by polynomial-size Boolean formulas, that is, iff* NL/*poly* $\subseteq \mathrm{NC}^1$.

As we shall see in Section 7, *tree-shaped* CQs and ontologies of depth 1 always have polynomial-size PE-rewritings.

6. Rewritings over Ontologies of Depth 2

Our next aim is to show that CQs and ontologies of depth 2 can compute the NP-complete function checking whether a graph with n vertices has a k-clique. We remind the reader (see, e.g., [3] for details) that the monotone Boolean function $\textsc{Clique}_{n,k}(\vec{e})$ of $n(n-1)/2$ variables $e_{jj'}$, $1 \leq j < j' \leq n$, returns 1 iff the graph with vertices $\{1, \dots, n\}$ and edges $\{\{j, j'\} \mid e_{jj'} = 1\}$ contains a k-clique. A series of papers, started by Razborov's [32], gave an exponential lower bound for the size of monotone circuits computing $\textsc{Clique}_{n,k}$: $2^{\Omega(\sqrt{k})}$ for $k \leq \frac{1}{4}(n/\log n)^{2/3}$ [2]. For monotone formulas, an even better lower bound is known: $2^{\Omega(k)}$ for $k = 2n/3$ [31].

We first construct a monotone HGP computing $\textsc{Clique}_{n,k}$ and then use the intuition behind the construction to encode $\textsc{Clique}_{n,k}$ via a Boolean CQ $\boldsymbol{q}_{n,k}$ and an ontology $\mathcal{T}_{n,k}$ of depth 2 and polynomial size. As a consequence, any PE- or NDL-rewriting of $\boldsymbol{q}_{n,k}$ and $\mathcal{T}_{n,k}$ is of exponential size, while any FO-rewriting is superpolynomial unless NP $\subseteq$ P/poly.

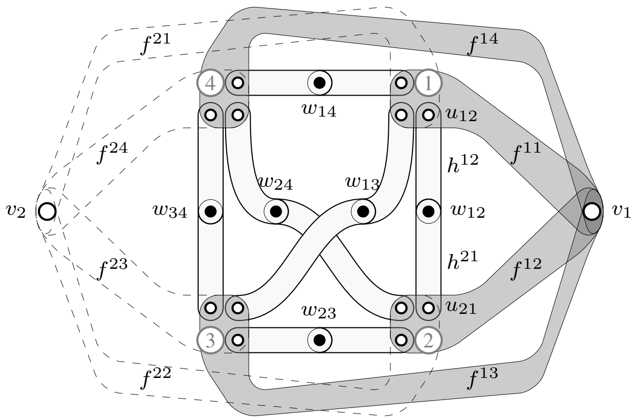
Given n and k, let $H_{n,k}$ be a monotone HGP with vertices

– v_i labelled with 0, for $1 \leq i \leq k$,
– $w_{jj'}$ labelled with $e_{jj'}$, for $1 \leq j < j' \leq n$,
– $u_{jj'}$ and $u_{j'j}$ labelled with 1, for $1 \leq j < j' \leq n$,

and hyperedges

$$f^{ij} = \{v_i\} \cup \{u_{jj'} \mid j' \neq j\} \;\; (1 \leq i \leq k \text{ and } 1 \leq j \leq n),$$
$$h^{jj'} = \{w_{jj'}, u_{jj'}\} \text{ and } h^{j'j} = \{w_{jj'}, u_{j'j}\} \;\; (1 \leq j < j' \leq n).$$

Informally, the $w_{jj'}$ represent the edges of the complete graph with n vertices; they can be turned 'on' or 'off' by means of the variables $e_{jj'}$. The vertex $u_{jj'}$ together with the hyperedge $h^{jj'}$ represent the 'half' of the edge connecting j and j' that is adjacent to j; the other 'half' is represented by $u_{j'j}$ and $h^{j'j}$. The vertices v_i represent a k-clique and the edge f^{ij} corresponds to the choice of the vertex j of the graph as the ith element of the clique. The hypergraph $H_{4,2}$ is shown below:



Theorem 15. *The HGP $H_{n,k}$ computes* $\text{CLIQUE}_{n,k}$.

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

($\Leftarrow$) Let function $\lambda \colon \{1,\dots,k\} \to \{1,\dots,n\}$ be such that $C = \{\lambda(i) \mid 1 \le i \le k\}$ is a k-clique in the graph given by $\vec{e}$. Then

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

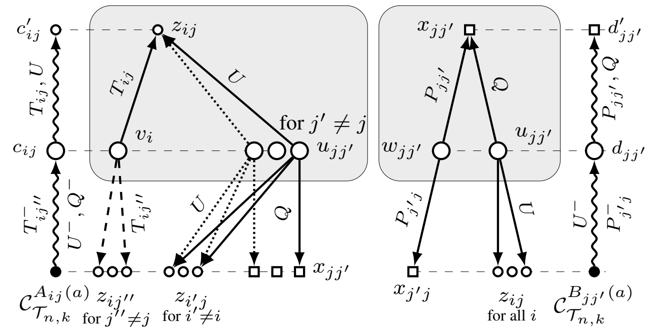
is independent and covers all zeros in $H_{n,k}$ under $\vec{e}$. Indeed, X is independent because, in every $h^{jj'} \in X$, the index j does not belong to C. By definition, each $f^{i\lambda(i)}$ covers v_i, for $1 \le i \le k$. Thus, it remains to show that any $w_{jj'}$ with $e_{jj'} = 0$ (that is, the edge $\{j, j'\}$ belongs to the complement of G) 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 $h^{jj'}$ 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 $h^{jj'}$ 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 $\vec{e}$. The vertex v_i is labelled with 0, and so there is $\lambda(i)$ such that $f^{i\lambda(i)} \in X$. We claim that $C = \{\lambda(i) \mid 1 \le i \le k\}$ is a k-clique in the graph given by $\vec{e}$. 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 $h^{jj'}$ or $h^{j'j}$ (see the picture above). But $h^{jj'}$ intersects $f^{\lambda^{-1}(j)j}$ and $h^{j'j}$ intersects $f^{\lambda^{-1}(j')j'}$, which is a contradiction. ❑

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

$T_{ij}(v_i, z_{ij})$	for $1 \le i \le k, 1 \le j \le n$,
$P_{jj'}(w_{jj'}, x_{jj'})$, $P_{j'j}(w_{jj'}, x_{j'j})$	for $1 \le j < j' \le n$,
$Q(u_{jj'}, x_{jj'})$, $U(u_{jj'}, z_{ij})$	for $1 \le j \ne j' \le n$ and $1 \le i \le k$.

The picture below illustrates the fragments of $\boldsymbol{q}_{n,k}$ centred in each variable of the form z_{ij} and $x_{jj'}$ (the fragment centred in $x_{j'j}$ 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 $1 \le i \le k$ and $1 \le j \ne j' \le n$,

$$A_{ij}(x) \to \exists y \, \big[\bigwedge_{j'' \ne j} T_{ij''}(y, x) \land U(y, x) \land Q(y, x) \land A'_{ij}(y) \big],$$

$$A'_{ij}(x) \to \exists y \, \big[T_{ij}(x, y) \land U(x, y) \big],$$

$$B_{jj'}(x) \to \exists y \, \big[P_{j'j}(y, x) \land U(y, x) \land B'_{jj'}(y) \big],$$

$$B'_{jj'}(x) \to \exists y \, \big[P_{jj'}(x, y) \land Q(x, y) \big].$$

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

– $\mathfrak{t}^{ij} = (\mathfrak{t}^{ij}_{\mathsf{r}}, \mathfrak{t}^{ij}_{\mathsf{i}})$ generated by $A_{ij}(x)$, for $1 \le i \le k$ and $1 \le j \le n$, where

$$\mathfrak{t}^{ij}_{\mathsf{r}} = \{z_{ij'}, x_{jj'} \mid 1 \le j' \le n, \ j' \ne j\} \cup \{z_{i'j} \mid 1 \le i' \le k, \ i \ne i'\},$$

$$\mathfrak{t}^{ij}_{\mathsf{i}} = \{v_i, z_{ij}\} \cup \{u_{jj'} \mid 1 \le j' \le n, \ j' \ne j\};$$

– $\mathfrak{s}^{jj'} = (\mathfrak{s}^{jj'}_{\mathsf{r}}, \mathfrak{s}^{jj'}_{\mathsf{i}})$ and $\mathfrak{s}^{j'j} = (\mathfrak{s}^{j'j}_{\mathsf{r}}, \mathfrak{s}^{j'j}_{\mathsf{i}})$, generated by $B_{jj'}(x)$ and $B_{j'j}(x)$, respectively, for $1 \le j < j' \le n$, where

$$\mathfrak{s}^{jj'}_{\mathsf{r}} = \{x_{j'j}\} \cup \{z_{ij} \mid 1 \le i \le k\}, \quad \mathfrak{s}^{jj'}_{\mathsf{i}} = \{w_{jj'}, u_{jj'}, x_{jj'}\},$$

$$\mathfrak{s}^{j'j}_{\mathsf{r}} = \{x_{jj'}\} \cup \{z_{ij'} \mid 1 \le i \le k\}, \quad \mathfrak{s}^{j'j}_{\mathsf{i}} = \{w_{jj'}, u_{j'j}, x_{j'j}\}.$$

The tree witnesses $\mathfrak{t}^{ij}$, $\mathfrak{s}^{jj'}$ and $\mathfrak{s}^{j'j}$ are uniquely determined by their most remote (from the root) variables, z_{ij}, $x_{jj'}$ and $x_{j'j}$, respectively, and correspond to the hyperedges f^{ij}, $h^{jj'}$, $h^{j'j}$ 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 $\vec{e}$ encoding a graph with n vertices, let $\mathcal{A}_{\vec{e}}$ be a data instance with one individual a and the following atoms:

$$Q(a,a), \quad U(a,a), \quad A_{ij}(a), \quad \text{for } 1 \le i \le k \text{ and } 1 \le j \le n,$$

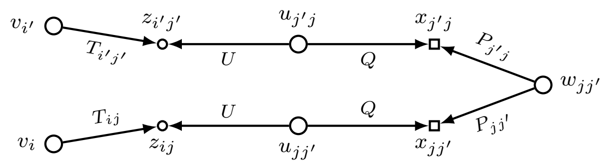
$$P_{jj'}(a,a) \text{ and } P_{j'j}(a,a), \quad \text{for } 1 \le j < j' \le n \text{ with } e_{jj'} = 1.$$

Lemma 16. $\mathcal{T}_{n,k}, \mathcal{A}_{\vec{e}} \models \boldsymbol{q}_{n,k}$ *iff* $\text{CLIQUE}_{n,k}(\vec{e}) = 1$.

Proof. ($\Rightarrow$) Suppose $\mathcal{T}_{n,k}, \mathcal{A}_{\vec{e}} \models \boldsymbol{q}_{n,k}$. Then there is a homomorphism g from $\boldsymbol{q}_{n,k}$ to the canonical model $\mathcal{C}$ of $(\mathcal{T}_{n,k}, \mathcal{A}_{\vec{e}})$. Since the only points of $\mathcal{C}$ that belong to $\exists y \, T_{ij}(x, y)$ are of the form c_{ij} (in the picture above) and $\boldsymbol{q}_{n,k}$ contains atoms of the form $T_{ij}(v_i, z_{ij})$, there is $\lambda \colon \{1,\dots,k\} \to \{1,\dots,n\}$ such that $g(v_i) = c_{i\lambda(i)}$. We claim that $C = \{\lambda(i) \mid 1 \le i \le k\}$ is a k-clique in the graph given by $\vec{e}$.

We first show that λ is injective. Suppose to the contrary that $\lambda(i) = \lambda(i') = j$, for $i \neq i'$. Since $\boldsymbol{q}_{n,k}$ contains $T_{ij}(v_i, z_{ij})$ and $T_{i'j}(v_{i'}, z_{i'j})$, we have $g(z_{ij}) = c'_{ij}$ and $g(z_{i'j}) = c'_{i'j}$. Take $j' \neq j$. Since $U(u_{jj'}, z_{ij}), U(u_{jj'}, z_{i'j}) \in \boldsymbol{q}_{n,k}$, we obtain $g(u_{jj'}) = c_{ij}$ and $g(u_{jj'}) = c_{i'j}$, contrary to $i \neq i'$.

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



($\Leftarrow$) Suppose $\lambda\colon \{1, \dots, k\} \to \{1, \dots, n\}$ is a k-clique. We construct a homomorphism g from $\boldsymbol{q}_{n,k}$ to the canonical model of $(\mathcal{T}_{n,k}, \mathcal{A}_{\vec{e}})$ 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 $\mathcal{T}_{n,k}, \mathcal{A}_{\vec{e}} \models \boldsymbol{q}_{n,k}$. ❑

Theorem 17. *There exists a sequence of CQs $\boldsymbol{q}_n$ and ontologies $\mathcal{T}_n$ of depth 2 any PE- and 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 $\boldsymbol{q}'_{n,k}$ of $\boldsymbol{q}_{n,k}$ 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 $\text{CLIQUE}_{n,k}$ of size $|\boldsymbol{q}'_{n,k}|$.

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

If $(\Pi, \boldsymbol{q}'_{n,k})$ is an NDL-rewriting of $\boldsymbol{q}_{n,k}$, we replace all the individual variables in Π with a and then perform the replacement described above. Denote the resulting propositional NDL-program by Π'. The program Π' can now be transformed into a monotone Boolean circuit computing $\text{CLIQUE}_{n,k}$: for every (propositional) variable p occurring in the head of a clause in Π', we introduce an $\vee$-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 $\wedge$-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 $\text{CLIQUE}_{n,k}$ given at the beginning of this section. ❑

As the function $\text{CLIQUE}_{n,k}$ 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 iff all functions in* NP/*poly are computed by polynomial-size formulas, that is, iff* NP/*poly* $\subseteq \text{NC}^1$.

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 $\mathsf{tw}_\mathfrak{t}$ formula (3) defines a predicate over the data that can be computed in linear time. It follows that, for a tree-shaped $\boldsymbol{q}$, every disjunct of (4) can also be regarded as a tree-shaped CQ of size $\leq |\boldsymbol{q}|$. So, bearing in mind that $|\Theta^{\boldsymbol{q}}_{\mathcal{T}}| \leq 3^{|\boldsymbol{q}|}$, we obtain the following:

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

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.

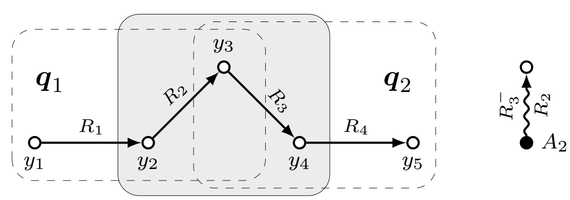
Example 20. Consider the following ontology:

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

and the following CQ:

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

illustrated in the picture below:



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

$$\boldsymbol{q}_1(y_3) = \exists y_1 y_2\, (R_1(y_1, y_2) \wedge R_2(y_2, y_3)),$$
$$\boldsymbol{q}_2(y_3) = \exists y_4 y_5\, (R_3(y_3, y_4) \wedge R_4(y_4, y_5)).$$

If $h(y_3)$ is a labelled null, then y_3 must be an internal point of some tree witness for $\boldsymbol{q}$ and $\mathcal{T}$. We have only one such tree witness, $\mathfrak{t} = (\mathfrak{t}_\mathsf{r}, \mathfrak{t}_\mathsf{i})$, generated by $A_2(x)$ with $\mathfrak{t}_\mathsf{r} = \{y_2, y_4\}$, $\mathfrak{t}_\mathsf{i} = \{y_3\}$ and $\boldsymbol{q}_\mathfrak{t} = \{R_2(y_2, y_3), R_3(y_3, y_4)\}$ (shaded in the picture above). But then $h(y_2) = h(y_4)$ and this element is in the data part of $\mathcal{C}_{\mathcal{T},\mathcal{A}}$. So, we need PE-rewritings, $\boldsymbol{q}_3^\dagger(y_2)$ and $\boldsymbol{q}_4^\dagger(y_4)$, of the remaining fragments of $\boldsymbol{q}$:

$$\boldsymbol{q}_3(y_2) = \exists y_1\, R_1(y_1, y_2), \qquad \boldsymbol{q}_4(y_4) = \exists y_5\, R_4(y_4, y_5).$$

If the required rewritings $\boldsymbol{q}_i^\dagger$, $1 \leq i \leq 4$, are constructed then we obtain a PE-rewriting $\boldsymbol{q}^\dagger$ of $\boldsymbol{q}$ and $\mathcal{T}$ by taking

$$\boldsymbol{q}^\dagger = \exists y_3 \big(\boldsymbol{q}_1^\dagger(y_3) \land \boldsymbol{q}_2^\dagger(y_3)\big) \lor \\ \exists y_2 y_4 \big(A_2(y_2) \land (y_2 = y_4) \land \boldsymbol{q}_3^\dagger(y_2) \land \boldsymbol{q}_4^\dagger(y_4)\big).$$

We analyse $\boldsymbol{q}_1, \boldsymbol{q}_2, \boldsymbol{q}_3$ and $\boldsymbol{q}_4$ in the same way and obtain

$$\boldsymbol{q}_1^\dagger(y_3) = \exists y_2\big(\boldsymbol{q}_3^\dagger(y_2) \land R_2(y_2, y_3)\big) \lor \exists y_1\big(A_1(y_3) \land (y_1 = y_3)\big),$$
$$\boldsymbol{q}_2^\dagger(y_3) = \exists y_4\big(R_3(y_3, y_4) \land \boldsymbol{q}_4^\dagger(y_4)\big) \lor \exists y_5\big(A_3(y_3) \land (y_5 = y_3)\big),$$

$\boldsymbol{q}_3^\dagger(y_2)$ and $\boldsymbol{q}_4^\dagger(y_4)$ equal to $\boldsymbol{q}_3(y_2)$ and $\boldsymbol{q}_4(y_4)$, 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 $\boldsymbol{q}(\vec{x}) = \exists \vec{y}\, \varphi(\vec{x}, \vec{y})$ and an ontology $\mathcal{T}$ be given. We recursively define a PE-query $\boldsymbol{q}^\dagger(\vec{x})$ as follows. Take the finest partition of $\exists \vec{y}\, \varphi(\vec{x}, \vec{y})$ into a conjunction $\bigwedge_j \exists \vec{y}_j\, \varphi_j(\vec{x}, \vec{y}_j)$ such that every atom containing some $y \in \vec{y}_j$ belongs to the same conjunct $\varphi_j(\vec{x}, \vec{y}_j)$. (Informally, the Gaifman graph of φ is cut along the answer variables $\vec{x}$.) By definition, the set of tree witnesses for $\exists \vec{y}\, \varphi(\vec{x}, \vec{y})$ and $\mathcal{T}$ is the disjoint union of the sets of tree witnesses for the $\exists \vec{y}_j\, \varphi_j(\vec{x}, \vec{y}_j)$ and $\mathcal{T}$. Then we set $(\exists \vec{y}\, \varphi(\vec{x}, \vec{y}))^\dagger = \bigwedge_j \psi_j$, where ψ_j is $\varphi_j(\vec{x})$ in case $\vec{y}_j$ is empty; otherwise, we choose a variable z in $\vec{y}_j$ and define ψ_j to be the formula

$$\exists z \left(\exists\, [\vec{y}_j \setminus \{z\}]\, \varphi_j(\vec{x}, \vec{y}_j)\right)^\dagger \ \lor \\ \bigvee_{\substack{\mathfrak{t} \text{ a tree witness for } \exists \vec{y}_j\, \varphi_j(\vec{x}, \vec{y}_j) \text{ and } \mathcal{T} \\ \text{such that } \mathfrak{t} = (\mathfrak{t}_\mathsf{r}, \mathfrak{t}_\mathsf{i}) \text{ and } z \in \mathfrak{t}_\mathsf{i}}} \exists \vec{y}_{j,\mathfrak{t}} \left(\left(\exists\, [\vec{y}_j \setminus \vec{y}_{j,\mathfrak{t}}]\, \varphi_{j,\mathfrak{t}}(\vec{x}, \vec{y}_j)\right)^\dagger \land \mathsf{tw}_\mathfrak{t}(\mathfrak{t}_\mathsf{r})\right),$$

where $\vec{y}_{j,\mathfrak{t}} = \vec{y}_j \cap \mathfrak{t}_\mathsf{r}$ contains the variables in $\vec{y}_j$ that occur among $\mathfrak{t}_\mathsf{r}$, the quantifiers $\exists\, [\vec{y}_j \setminus \{z\}]$ and $\exists\, [\vec{y}_j \setminus \vec{y}_{j,\mathfrak{t}}]$ contain all variables in $\vec{y}_j$ but z and $\vec{y}_{j,\mathfrak{t}}$, respectively, and $\varphi_{j,\mathfrak{t}}$ consists of all the atoms of φ_j except those in $\boldsymbol{q}_\mathfrak{t}$. Note that the variables in $\mathfrak{t}_\mathsf{i}$ (in particular, z) do not occur in the disjunct for $\mathfrak{t}$ (and so can be removed from the respective quantifier). Intuitively, the first disjunct represents the situation where z is mapped to a data individual and treated as a free variable in the rewriting of φ_j. The other disjuncts reflect the cases where z is mapped to a labelled null, and so z is an internal variable of a tree witness $\mathfrak{t} = (\mathfrak{t}_\mathsf{r}, \mathfrak{t}_\mathsf{i})$ for $\exists \vec{y}_j\, \varphi_j(\vec{x}, \vec{y}_j)$ and $\mathcal{T}$. As the variables in $\mathfrak{t}_\mathsf{r}$ must be mapped to data individuals, this only leaves the set of atoms $\varphi_{j,\mathfrak{t}}$ with existentially quantified $\vec{y}_j \setminus \vec{y}_{j,\mathfrak{t}}$ for further rewriting. The existentially quantified variables in each of the disjuncts do not contain z, and so our recursion is well-founded. The proof of the following theorem is straightforward (remember that all our rewritings in this paper are over complete data):

Theorem 21. *For any CQ $\boldsymbol{q}(x)$ and ontology $\mathcal{T}$, $\boldsymbol{q}^\dagger(\vec{x})$ is a PE-rewriting of $\boldsymbol{q}$ and $\mathcal{T}$ (over complete data).*

The exact form of the rewriting $\boldsymbol{q}^\dagger$ 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

$$d_\mathcal{T}^{\boldsymbol{q}} \ = \ 1 + \max_{z \in \vec{y}} \big|\{\mathfrak{t} = (\mathfrak{t}_\mathsf{r}, \mathfrak{t}_\mathsf{i}) \in \Theta_\mathcal{T}^{\boldsymbol{q}} \mid z \in \mathfrak{t}_\mathsf{i}\}\big|.$$

We call $d_\mathcal{T}^{\boldsymbol{q}}$ the *tree-witness degree of* $\boldsymbol{q}$ and $\mathcal{T}$. 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 $1 + |\Theta_\mathcal{T}^{\boldsymbol{q}}|$.

Given a tree-shaped CQ $\boldsymbol{q}(\vec{x}) = \exists \vec{y}\, \varphi(\vec{x}, \vec{y})$, we pick some variable as its root and define a partial order $\preceq$ on the variables of $\boldsymbol{q}$ by taking $z \preceq z'$ iff z' occurs in the subtree of $\boldsymbol{q}$ rooted in z. The strategy used in [8] chooses the smallest z with respect to $\preceq$. Since the number of distinct subtrees of $\boldsymbol{q}$ is bounded by $|\boldsymbol{q}|$ and NDL programs allow for structure sharing, this strategy yields an NDL-rewriting of size $|\mathcal{T}| \cdot |\boldsymbol{q}| \cdot d_\mathcal{T}^{\boldsymbol{q}}$:

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 rewiring process with the above strategy is bounded by $|\boldsymbol{q}|$, we can only obtain a PE-rewriting of exponential size in $|\boldsymbol{q}|$. However, if we adopt the strategy of choosing z that splits the graph of each φ_j in half, then the depth of recursion does not exceed $\log |\boldsymbol{q}|$, and so the resulting PE-rewriting is of polynomial size for $\boldsymbol{q}$ and $\mathcal{T}$ 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 $\boldsymbol{q}$ and any ontology $\mathcal{T}$, there is a PE-rewriting of size $|\mathcal{T}| \cdot |\boldsymbol{q}|^{1+\log d_\mathcal{T}^{\boldsymbol{q}}}$ (over complete data).*

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

$$F(n_j) + \sum_{i=1}^{d-1} (F(n_j - m_{ji}) + |\mathcal{T}| \cdot m_{ji}),$$

where n_j is the number of atoms in $\boldsymbol{p}_j$ and m_{ji} is the number of atoms in the ith tree witness with $z \in \mathfrak{t}_\mathsf{i}$, $1 \leq m_{ji} \leq n_j$. By the induction hypothesis, the length of the contribution of $\boldsymbol{p}_j$ does not exceed

$$|\mathcal{T}| \cdot n_j^{1+\log d} + |\mathcal{T}| \cdot \sum_{i=1}^{d-1} \big((n_j - m_{ji})^{1+\log d} + m_{ji}\big) \ \leq \\ |\mathcal{T}| \cdot \big(n_j^{1+\log d} + (d-1) \cdot n_j^{1+\log d}\big) = |\mathcal{T}| \cdot d \cdot n_j^{1+\log d}.$$

By Proposition 23, we can choose z (at the preceding step) so that $\boldsymbol{p}$ with n atoms is split into components $\boldsymbol{p}_1, \dots, \boldsymbol{p}_k$ each of which has $n_j \leq n/2$ atoms (by definition, $\sum_{j=1}^k n_j = n$). This gives

$$F(n) \ \leq \ |\mathcal{T}| \cdot d \cdot \sum_{j=1}^{k} \big((n/2)^{\log d} \cdot n_j\big) \ \leq |\mathcal{T}| \cdot n^{1+\log d}$$

as required. ❑

Corollary 25. *Any tree-shaped CQ $\boldsymbol{q}$ and ontology $\mathcal{T}$ of depth 1 have a PE-rewriting of size $|\mathcal{T}| \cdot |\boldsymbol{q}|^2$ (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

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

[2] N. Alon and R. Boppana. The monotone circuit complexity of Boolean functions. *Combinatorica*, 7(1):1–22, 1987.

[3] S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, New York, USA, 1st edition, 2009.

[4] A. Artale, D. Calvanese, R. Kontchakov, and M. Zakharyaschev. The DL-Lite family and relations. *J. Artif. Intell. Res. (JAIR)*, 36:1–69, 2009.

[5] B. Aspvall, M. Plass, and R. Tarjan. A linear-time algorithm for testing the truth of certain quantified Boolean formulas. *Inf. Proc. Letters*, 8(3):121–123, 1979.

[6] J. Avigad. Eliminating definitions and Skolem functions in first-order logic. In *Proc. of LICS*, pages 139–146. IEEE Computer Society, 2001.

[7] J.-F. Baget, M. Leclère, M.-L. Mugnier, and E. Salvat. Extending decidable cases for rules with existential variables. In *Proc. of IJCAI*, pages 677–682, 2009.

[8] M. Bienvenu, M. Ortiz, M. Simkus, and G. Xiao. Tractable queries for lightweight description logics. In *Proc. of IJCAI*. IJCAI/AAAI, 2013.

[9] E. Börger, E. Grädel, and Y. Gurevich. *The Classical Decision Problem*. Perspectives in Mathematical Logic. Springer, 1997.

[10] A. Calì, G. Gottlob, and T. Lukasiewicz. A general datalog-based framework for tractable query answering over ontologies. *J. Web Semant.*, 14:57–83, 2012.

[11] A. Calì, G. Gottlob, and A. Pieris. Towards more expressive ontology languages: the query answering problem. *Artif. Intell.*, 193:87–128, 2012.

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

[13] C. Chekuri and A. Rajaraman. Conjunctive query containment revisited. *Theor. Comput. Sci.*, 239(2):211–229, 2000.

[14] A. Chortaras, D. Trivela, and G. Stamou. Optimized query rewriting for OWL 2 QL. In *Proc. of CADE*, vol. 6803 of *LNCS*, pages 192–206. Springer, 2011.

[15] T. Eiter, M. Ortiz, M. Šimkus, T.-K. Tran, and G. Xiao. Query rewriting for Horn-SHIQ plus rules. In *Proc. of AAAI*. AAAI Press, 2012.

[16] G. Gottlob, S. Kikot, R. Kontchakov, V. Podolskii, T. Schwentick, and M. Zakharyaschev. The price of query rewriting in ontology-based data access. *Artif. Intell.* 2014.

[17] G. Gottlob, G. Orsi, and A. Pieris. Ontological queries: Rewriting and optimization. In *Proc. of ICDE*, pages 2–13. IEEE Computer Society, 2011.

[18] G. Gottlob and T. Schwentick. Rewriting ontological queries into small nonrecursive datalog programs. In *Proc. of KR*. AAAI Press, 2012.

[19] S. Jukna. *Boolean Function Complexity — Advances and Frontiers*, volume 27 of *Algorithms and Combinatorics*. Springer, 2012.

[20] M. Karchmer and A. Wigderson. Monotone circuits for connectivity require super-logarithmic depth. In *Proc. of STOC*, pages 539–550. ACM, 1988.

[21] S. Kikot, R. Kontchakov, V. Podolskii, and M. Zakharyaschev. Exponential lower bounds and separation for query rewriting. In *Proc. of ICALP*, vol. 7392 of *LNCS*, pages 263–274. Springer, 2012.

[22] S. Kikot, R. Kontchakov, V. Podolskii, and M. Zakharyaschev. On the succinctness of query rewriting over OWL 2 QL ontologies with shallow chases. *CoRR*, abs/1401.4420, 2014.

[23] S. Kikot, R. Kontchakov, and M. Zakharyaschev. On (in)tractability of OBDA with OWL 2 QL. In *Proc. of DL*, vol. 745 of *CEUR-WS*, 2011.

[24] S. Kikot, R. Kontchakov, and M. Zakharyaschev. Conjunctive query answering with OWL 2 QL. In *Proc. of KR*. AAAI Press, 2012.

[25] M. König, M. Leclère, M.-L. Mugnier, and M. Thomazo. On the exploration of the query rewriting space with existential rules. In *Proc. of RR*, vol. 7994 of *LNCS*, pages 123–137. Springer, 2013.

[26] C. Lutz, I. Seylan, D. Toman, and F. Wolter. The combined approach to OBDA: taming role hierarchies using filters. In *Proc. of ISWC*, vol. 8218 of *LNCS*, pages 314–330. Springer, 2013.

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

[28] H. Pérez-Urbina, B. Motik, and I. Horrocks. A comparison of query rewriting techniques for DL-Lite. In *Proc. of DL*, vol. 477 of *CEUR-WS*, 2009.

[29] H. Pérez-Urbina, E. Rodríguez-Díaz, M. Grove, G. Konstantinidis, and E. Sirin. Evaluation of query rewriting approaches for OWL 2. In *Proc. of SSWS+HPCSW*, vol. 943 of *CEUR-WS*, 2012.

[30] A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. Linking data to ontologies. *J. on Data Semantics*, X:133–173, 2008.

[31] R. Raz and A. Wigderson. Monotone circuits for matching require linear depth. *J. ACM*, 39(3):736–744, 1992.

[32] A. Razborov. Lower bounds for the monotone complexity of some Boolean functions. *Dokl. Akad. Nauk SSSR*, 281(4):798–801, 1985.

[33] A. Razborov. Lower bounds for deterministic and nondeterministic branching programs. In *Proc. of FCT*, vol. 529 of *LNCS*, pages 47–60. Springer, 1991.

[34] M. Rodríguez-Muro, R. Kontchakov, and M. Zakharyaschev. Ontology-based data access: Ontop of databases. In *Proc. of ISWC*, vol. 8218 of LNCS, pages 558–573. Springer, 2013.

[35] R. Rosati. Prexto: Query rewriting under extensional constraints in DL-Lite. In *Proc. of ESWC*, vol. 7295 of *LNCS*, pages 360–374. Springer, 2012.

[36] R. Rosati and A. Almatelli. Improving query answering over DL-Lite ontologies. In *Proc. of KR*. AAAI Press, 2010.

[37] M. Yannakakis. Algorithms for acyclic database schemes. In *Proc. of VLDB*, pages 82–94. IEEE Computer Society, 1981.