# Exponential Lower Bounds and Separation for Query Rewriting[⋆]

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

[1] Department of Computer Science and Information Systems
Birkbeck, University of London, U.K.
{kikot,roman,michael}@dcs.bbk.ac.uk
[2] Steklov Mathematical Institute, Moscow, Russia. podolskii@mi.ras.ru

**Abstract.** We establish connections between the size of circuits and formulas computing monotone Boolean functions and the size of first-order and nonrecursive Datalog rewritings for conjunctive queries over *OWL 2 QL* ontologies. We use known lower bounds and separation results from circuit complexity to prove similar results for the size of rewritings that do not use non-signature constants. For example, we show that, in the worst case, positive existential and nonrecursive Datalog rewritings are exponentially longer than the original queries; nonrecursive Datalog rewritings are in general exponentially more succinct than positive existential rewritings; while first-order rewritings can be superpolynomially more succinct than positive existential rewritings.

## 1 Introduction

First-order (FO) rewritability is the key concept of ontology-based data access (OBDA), which is believed to lie at the foundations of the next generation of information systems. A language $\mathcal{L}$ enjoys *FO-rewritability* if any conjunctive query $\boldsymbol{q}$ over an ontology $\mathcal{T}$, formulated in $\mathcal{L}$, can be transformed into an FO-formula $\boldsymbol{q}'$ such that, for any data $\mathcal{A}$, the certain answers to $\boldsymbol{q}$ over the knowledge base $(\mathcal{T}, \mathcal{A})$ can be found by querying $\boldsymbol{q}'$ over $\mathcal{A}$ using a standard relational database management system (RDBMS). Ontology languages with this property include the *OWL 2 QL* profile of the Web Ontology Language *OWL 2*, which is based on the *DL-Lite* family of description logics [11, 4], and fragments of Datalog$^{\pm}$ such as linear or sticky sets of TGDs [9, 10]. Various rewriting techniques have been implemented in the systems QuOnto [1], REQUIEM [19], Presto [26], Nyaya [12] and Quest [25].

OBDA via FO-rewritability relies on the empirical fact that RDBMSs are usually very efficient in practice. However, this does not mean that they can efficiently evaluate any given query: after all, for expression complexity, database query answering is PSPACE-complete for FO-queries and NP-complete for conjunctive queries (CQs). Indeed, the first 'naïve' rewritings of CQs over *OWL 2 QL* ontologies turned out to be too lengthy even for modern RDBMSs [11, 19]. The

[⋆] A full version of this paper is available at http://arxiv.org/abs/1202.4193.

next step was to develop various rewriting optimisation techniques [26, 12, 23, 24]; however, they still produced exponential-size — $O((|\mathcal{T}| \cdot |\boldsymbol{q}|)^{|\boldsymbol{q}|})$ — rewritings in the worst case. An alternative two-step combined approach to OBDA with *OWL 2 EL* [18] and *OWL 2 QL* [17] first expands the data by applying the ontology axioms and introducing new individuals required by the ontology, and only then rewrites the query over the expanded data. Yet, even with these extra resources a simple polynomial rewriting was constructed only for the fragment of *OWL 2 QL* without role inclusions; the rewriting for the full language remained exponential. A breakthrough seemed to come in [13], which showed that one can construct, in polynomial time, a nonrecursive Datalog rewriting for some fragments of Datalog$^\pm$ containing *OWL 2 QL*. However, this rewriting uses the built-in predicate $\neq$ and numerical constants that are not present in the original query and ontology. Without additional constants, no FO-rewriting for *OWL 2 QL* can be *constructed* in polynomial time [15] (it remained unclear, however, whether such an FO-rewriting of polynomial size *exists*).

These developments bring forward a spectrum of theoretical and practical questions that could influence the future of OBDA. What is the worst-case size of FO- and nonrecursive Datalog rewritings for CQs over *OWL 2 QL* ontologies? What is the type/shape/size of rewritings we should aim at to make OBDA with *OWL 2 QL* efficient? What extra means (e.g., built-in predicates and constants) can be used in the rewritings? In this paper, we investigate the worst-case size of FO- and nonrecursive Datalog rewritings for CQs over *OWL 2 QL* ontologies depending on the available means. We distinguish between 'pure' rewritings, which cannot use constants that do not occur in the original query, and 'impure' ones, where such constants are allowed. Our results can be summarised as follows:

- An exponential blow-up is unavoidable for pure positive existential rewritings and pure nonrecursive Datalog rewritings. Even pure FO-rewritings with $=$ can blow-up superpolynomially unless NP $\subseteq$ P/poly.
- Pure nonrecursive Datalog rewritings are in general exponentially more succinct than pure positive existential rewritings.
- Pure FO-rewritings can be superpolynomially more succinct than pure positive existential rewritings.
- Impure positive existential rewritings can always be made polynomial, and so they are exponentially more succinct than pure rewritings.

We obtain these results by first establishing connections between pure rewritings for CQs over *OWL 2 QL* ontologies and circuits for monotone Boolean functions, and then using known lower bounds and separation results for the circuit complexity of such functions as $\textsc{Clique}_{n,k}$ 'a graph with $n$ nodes contains a $k$-clique' and $\textsc{Matching}_{2n}$ 'a bipartite graph with $n$ vertices in each part has a perfect matching.'

## 2 Queries over *OWL 2 QL* Ontologies

By a *signature*, $\Sigma$, we understand in this paper any set of constant symbols and predicate symbols (with their arity). Unless explicitly stated otherwise, $\Sigma$

does *not* contain any predicates with fixed semantics, such as $=$ or $\neq$. In the description logic (or *OWL 2 QL*) setting, constant symbols are called *individual names*, $a_i$, while unary and binary predicate symbols are called *concept names*, $A_i$, and *role names*, $P_i$, respectively, where $i \geq 1$. The language of *OWL 2 QL* is built using these names in the following way. The *roles* $R$, *basic concepts* $B$ and *concepts* $C$ of *OWL 2 QL* are defined by the grammar:

$$
\begin{array}{rclcll}
R & ::= & P_i & \mid & P_i^-, & \qquad (P_i(x,y) \ \mid \ P_i(y,x)) \\
B & ::= & \bot & \mid & A_i \quad \mid \quad \exists R, & \qquad (\bot \ \mid \ A_i(x) \ \mid \ \exists y\, R(x,y)) \\
C & ::= & B & \mid & \exists R.B, & \qquad (B(x) \ \mid \ \exists y\, (R(x,y) \wedge B(y)))
\end{array}
$$

where the formulas on the right give a first-order translation of the *OWL 2 QL* constructs. An *OWL 2 QL TBox*, $\mathcal{T}$, is a finite set of *inclusions* of the form

$$
\begin{array}{ll}
B \sqsubseteq C, & \qquad (\forall x\, (B(x) \to C(x))) \\
R_1 \sqsubseteq R_2, & \qquad (\forall x, y\, (R_1(x,y) \to R_2(x,y))) \\
B_1 \sqcap B_2 \sqsubseteq \bot, & \qquad (\forall x\, (B_1(x) \wedge B_2(x) \to \bot)) \\
R_1 \sqcap R_2 \sqsubseteq \bot. & \qquad (\forall x, y\, (R_1(x,y) \wedge R_2(x,y) \to \bot))
\end{array}
$$

Note that concepts of the form $\exists R.B$ can only occur in the right-hand side of concept inclusions in *OWL 2 QL*. An *ABox*, $\mathcal{A}$, is a finite set of *assertions* of the form $A_k(a_i)$ and $P_k(a_i, a_j)$. $\mathcal{T}$ and $\mathcal{A}$ together form the *knowledge base* (KB) $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. The semantics for *OWL 2 QL* is defined in the usual way [6], based on interpretations $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ with domain $\Delta^{\mathcal{I}}$ and interpretation function $\cdot^{\mathcal{I}}$. The set of individuals in $\mathcal{A}$ is denoted by $\mathsf{ind}(\mathcal{A})$. For concepts or roles $E_1, E_2$, we write $E_1 \sqsubseteq_{\mathcal{T}} E_2$ if $\mathcal{T} \models E_1 \sqsubseteq E_2$; and we set $[E] = \{E' \mid E \sqsubseteq_{\mathcal{T}} E' \text{ and } E' \sqsubseteq_{\mathcal{T}} E\}$.

A *conjunctive query* (CQ) $\boldsymbol{q}(\boldsymbol{x})$ is an FO-formula $\exists \boldsymbol{y}\, \varphi(\boldsymbol{x}, \boldsymbol{y})$, where $\varphi$ is a conjunction of atoms of the form $A_k(t_1)$ and $P_k(t_1, t_2)$, and each $t_i$ is a *term* (an individual or a variable from $\boldsymbol{x}$ or $\boldsymbol{y}$). A tuple $\boldsymbol{a} \subseteq \mathsf{ind}(\mathcal{A})$ is a *certain answer* to $\boldsymbol{q}(\boldsymbol{x})$ over $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ if $\mathcal{I} \models \boldsymbol{q}(\boldsymbol{a})$ for all $\mathcal{I} \models \mathcal{K}$; in this case we write $\mathcal{K} \models \boldsymbol{q}(\boldsymbol{a})$.

Query answering over *OWL 2 QL* KBs is based on the fact that, for any consistent KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, there is an interpretation $\mathcal{C}_{\mathcal{K}}$ such that, for all CQs $\boldsymbol{q}(\boldsymbol{x})$ and $\boldsymbol{a} \subseteq \mathsf{ind}(\mathcal{A})$, we have $\mathcal{K} \models \boldsymbol{q}(\boldsymbol{a})$ iff $\mathcal{C}_{\mathcal{K}} \models \boldsymbol{q}(\boldsymbol{a})$. The interpretation $\mathcal{C}_{\mathcal{K}}$, called the *canonical model* of $\mathcal{K}$, can be constructed as follows. For each pair $[R], [B]$ with $\exists R.B$ in $\mathcal{T}$ (we assume $\exists R$ is just a shorthand for $\exists R.\top$), we introduce a fresh symbol $w_{[RB]}$ and call it the *witness for* $\exists R.B$. We write $\mathcal{K} \models C(w_{[RB]})$ if $\exists R^- \sqsubseteq_{\mathcal{T}} C$ or $B \sqsubseteq_{\mathcal{T}} C$. Define a *generating relation*, $\rightsquigarrow$, on the set of these witnesses together with $\mathsf{ind}(\mathcal{A})$ by taking:

- $a \rightsquigarrow w_{[RB]}$ if $a \in \mathsf{ind}(\mathcal{A})$, $[R]$ and $[B]$ are $\sqsubseteq_{\mathcal{T}}$-minimal with $\mathcal{K} \models \exists R.B(a)$ and there is no $b \in \mathsf{ind}(\mathcal{A})$ with $\mathcal{K} \models R(a,b) \wedge B(b)$;
- $w_{[R'B']} \rightsquigarrow w_{[RB]}$ if, for some $u$, $u \rightsquigarrow w_{[R'B']}$, $[R]$, $[B]$ are $\sqsubseteq_{\mathcal{T}}$-minimal with $\mathcal{K} \models \exists R.B(w_{[R'B']})$ and it is not the case that $R' \sqsubseteq_{\mathcal{T}} R^-$ and $\mathcal{K} \models B'(u)$.

If $a \rightsquigarrow w_{[R_1B_1]} \rightsquigarrow \cdots \rightsquigarrow w_{[R_nB_n]}$, $n \geq 0$, then we say that *a generates the path* $aw_{[R_1B_1]} \cdots w_{[R_nB_n]}$. Denote by $\mathsf{path}_{\mathcal{K}}(a)$ the set of paths generated by $a$, and by $\mathsf{tail}(\pi)$ the last element in $\pi \in \mathsf{path}_{\mathcal{K}}(a)$. Then $\mathcal{C}_{\mathcal{K}}$ is defined by taking:

$$\begin{aligned}
\Delta^{\mathcal{C}_{\mathcal{K}}} &= \bigcup_{a \in \mathsf{ind}(\mathcal{A})} \mathsf{path}_{\mathcal{K}}(a), \quad a^{\mathcal{C}_{\mathcal{K}}} \ = \ a, \text{ for } a \in \mathsf{ind}(\mathcal{A}),\\
A^{\mathcal{C}_{\mathcal{K}}} &= \{\pi \in \Delta^{\mathcal{C}_{\mathcal{K}}} \mid \mathcal{K} \models A(\mathsf{tail}(\pi))\},\\
P^{\mathcal{C}_{\mathcal{K}}} &= \{(a,b) \in \mathsf{ind}(\mathcal{A}) \times \mathsf{ind}(\mathcal{A}) \mid \mathcal{K} \models P(a,b)\} \cup\\
&\quad\ \{(\pi, \pi \cdot w_{[RB]}) \mid \mathsf{tail}(\pi) \rightsquigarrow w_{[RB]},\ R \sqsubseteq_{\mathcal{T}} P\} \cup\\
&\quad\ \{(\pi \cdot w_{[RB]}, \pi) \mid \mathsf{tail}(\pi) \rightsquigarrow w_{[RB]},\ R \sqsubseteq_{\mathcal{T}} P^-\}.
\end{aligned}$$

**Theorem 1 ([11, 17]).** *For every OWL 2 QL KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, every CQ $\boldsymbol{q}(\boldsymbol{x})$ and every $\boldsymbol{a} \subseteq \mathsf{ind}(\mathcal{A})$, $\mathcal{K} \models \boldsymbol{q}(\boldsymbol{a})$ iff $\mathcal{C}_{\mathcal{K}} \models \boldsymbol{q}(\boldsymbol{a})$.*

Let $\Sigma$ be a signature that can be used to formulate queries and ABoxes (remember that $\Sigma$ does not contain any built-in predicates). Given an ABox $\mathcal{A}$ over $\Sigma$, define $\mathcal{I}_{\mathcal{A}}$ to be the interpretation whose domain consists of all individuals in $\Sigma$ (even if they do not occur in $\mathsf{ind}(\mathcal{A})$) and $\mathcal{I}_{\mathcal{A}} \models E(\boldsymbol{a})$ iff $E(\boldsymbol{a}) \in \mathcal{A}$, for all predicates $E(\boldsymbol{x})$.

Given a CQ $\boldsymbol{q}(\boldsymbol{x})$ and a TBox $\mathcal{T}$, a first-order formula $\boldsymbol{q}'(\boldsymbol{x})$ over $\Sigma$ is called an *FO-rewriting for $\boldsymbol{q}(\boldsymbol{x})$ and $\mathcal{T}$ over $\Sigma$* if, for any ABox $\mathcal{A}$ over $\Sigma$ and any $\boldsymbol{a} \subseteq \mathsf{ind}(\mathcal{A})$, we have $(\mathcal{T}, \mathcal{A}) \models \boldsymbol{q}(\boldsymbol{a})$ iff $\mathcal{I}_{\mathcal{A}} \models \boldsymbol{q}'(\boldsymbol{a})$. If $\boldsymbol{q}'$ is an FO-rewriting of the form $\exists \boldsymbol{y}\, \varphi(\boldsymbol{x}, \boldsymbol{y})$, where $\varphi$ is built from atoms using only $\wedge$ and $\vee$, then we call $\boldsymbol{q}'(\boldsymbol{x})$ a *positive existential rewriting for $\boldsymbol{q}(\boldsymbol{x})$ and $\mathcal{T}$ over $\Sigma$* (or a *PE-rewriting*, for short). The *size* $|\boldsymbol{q}'|$ of $\boldsymbol{q}'$ is the number of symbols in $\boldsymbol{q}'$.

All known FO-rewritings for CQs and *OWL 2 QL* ontologies are of exponential size in the worst case. More precisely, for any CQ $\boldsymbol{q}$ and *OWL 2 QL* TBox $\mathcal{T}$, there exists a PE-rewriting of size $O((|\mathcal{T}| \cdot |\boldsymbol{q}|)^{|\boldsymbol{q}|})$ [11, 19, 12, 17]. One of the main results of this paper is that this bound cannot be substantially improved in general, even for FO-rewritings. On the other hand, we also show that FO-rewritings can be superpolynomially more succinct than PE-rewritings.

We also consider rewritings in the form of nonrecursive Datalog queries. We remind the reader that a *Datalog program*, $\Pi$, is a finite set of Horn clauses $\forall \boldsymbol{x}\, (A_1 \wedge \dots \wedge A_m \to A_0)$, where each $A_i$ is an atom of the form $P(t_1, \dots, t_l)$ and each $t_j$ is either a variable from $\boldsymbol{x}$ or a constant. $A_0$ is called the *head* of the clause, and $A_1, \dots, A_m$ its *body*; all variables occurring in the head must also occur in the body. A predicate $P$ *depends* on a predicate $Q$ in $\Pi$ if $\Pi$ contains a clause whose head is $P$ and whose body contains $Q$. $\Pi$ is called *nonrecursive* if this dependence relation for $\Pi$ is acyclic. A *nonrecursive Datalog query* consists of a nonrecursive Datalog program $\Pi$ and a *goal* $G$, which is just a predicate. Given an ABox $\mathcal{A}$, a tuple $\boldsymbol{a} \subseteq \mathsf{ind}(\mathcal{A})$ is a *certain answer to* $(\Pi, G)$ *over* $\mathcal{A}$ if $\Pi, \mathcal{A} \models G(\boldsymbol{a})$. The *size* $|\Pi|$ of $\Pi$ is the number of symbols in $\Pi$.

We distinguish between *pure* and *impure* Datalog queries [7]. In a *pure query* $(\Pi, G)$, the clauses in $\Pi$ do not contain constant symbols in their heads. One reason for considering only pure queries in OBDA is that impure ones can add new facts to the database that do not follow from the intensional knowledge in the background ontology. Impure nonrecursive Datalog queries are known to be more succinct than pure ones.

Given a CQ $\boldsymbol{q}(\boldsymbol{x})$ and a TBox $\mathcal{T}$, a pure nonrecursive Datalog query $(\Pi, G)$ is called a *nonrecursive Datalog rewriting for* $\boldsymbol{q}(\boldsymbol{x})$ *and* $\mathcal{T}$ *over* $\Sigma$ (or an *NDL-rewriting*, for short) if, for any ABox $\mathcal{A}$ over $\Sigma$ and any $\boldsymbol{a} \subseteq \mathsf{ind}(\mathcal{A})$, we have $(\mathcal{T}, \mathcal{A}) \models \boldsymbol{q}(\boldsymbol{a})$ iff $\Pi, \mathcal{A} \models G(\boldsymbol{a})$ (note that $\Pi$ may define predicates that are not in $\Sigma$, but may not use non-signature constants). Similarly to FO-rewritings, known NDL-rewritings for *OWL 2 QL* are of exponential size [26, 12]. Here we show that, in general, one cannot make NDL-rewritings shorter. On the other hand, they can be exponentially more succinct than PE-rewritings.

The rewritings can be much shorter if non-signature predicates and constants become available. As follows from [13], every CQ over an *OWL 2 QL* ontology can be rewritten as a polynomial-size nonrecursive Datalog query if we can use the inequality predicate and at least two distinct constants (cf. also [5], which shows how two constants and $=$ can be used to eliminate definitions from first-order theories without an exponential blow-up). In fact, we observe that, using equality and two distinct constants, any CQ over an *OWL 2 QL* ontology can be rewritten into a PE-query of polynomial size.

## 3 Boolean Circuits, CNFs and OBDA

To establish the lower and upper bounds for the size of rewritings mentioned above, we show first how the problem of constructing formulas and circuits that compute monotone Boolean functions can be reduced to the problem of finding FO- and NDL-rewritings for CQs over *OWL 2 QL* ontologies.

By an *$n$-ary Boolean function*, for $n \geq 1$, we mean a function from $\{0,1\}^n$ to $\{0,1\}$. A Boolean function $f$ is *monotone* if $f(\boldsymbol{\alpha}) \leq f(\boldsymbol{\alpha}')$, for all $\boldsymbol{\alpha} \leq \boldsymbol{\alpha}'$, where $\leq$ is the component-wise relation $\leq$ on vectors of $\{0,1\}$.

We remind the reader (for more details see, e.g., [3, 14]) that an *$n$-input Boolean circuit*, $\mathbf{C}$, is a directed acyclic graph with $n$ sources, *inputs*, and one sink, *output*. Every non-source node of $\mathbf{C}$ is called a *gate* and is labelled with either $\land$ or $\lor$, in which case it has two incoming edges, or with $\neg$, in which case it has one incoming edge. A circuit is *monotone* if it contains only $\land$ and $\lor$ gates. *Boolean formulas* can be thought of as circuits in which every gate has at most one outgoing edge. For an input $\boldsymbol{\alpha} \in \{0,1\}^n$, the *output* of $\mathbf{C}$ on $\boldsymbol{\alpha}$ is denoted by $\mathbf{C}(\boldsymbol{\alpha})$, and $\mathbf{C}$ is said to *compute* an $n$-ary Boolean function $f$ if $\mathbf{C}(\boldsymbol{\alpha}) = f(\boldsymbol{\alpha})$, for every $\boldsymbol{\alpha} \in \{0,1\}^n$. The number of nodes in $\mathbf{C}$ is the *size* of $\mathbf{C}$, denoted $|\mathbf{C}|$.

A *family of Boolean functions* is a sequence $f^1, f^2, \dots$, where each $f^n$ is an $n$-ary Boolean function. We say that a family $f^1, f^2, \dots$ is in the complexity class NP if there exist polynomials $p$ and $T$ and, for each $n \geq 1$, a Boolean circuit $\mathbf{C}^n$ with $n + p(n)$ inputs such that $|\mathbf{C}^n| \leq T(n)$ and, for each $\boldsymbol{\alpha} \in \{0,1\}^n$, we have

$$f^n(\boldsymbol{\alpha}) = 1 \qquad \text{iff} \qquad \mathbf{C}^n(\boldsymbol{\alpha}, \boldsymbol{\beta}) = 1, \quad \text{for some } \boldsymbol{\beta} \in \{0,1\}^{p(n)}.$$

The additional $p(n)$ inputs for $\boldsymbol{\beta}$ in the $\mathbf{C}^n$ are called *advice inputs*.

Given a family $f^1, f^2, \dots$ of monotone Boolean functions in NP, we construct a sequence of *OWL 2 QL* TBoxes $\mathcal{T}_{f^n}$ and CQs $\boldsymbol{q}_{f^n}$ without answer variables, as well as ABoxes $\mathcal{A}_{\boldsymbol{\alpha}}$, $\boldsymbol{\alpha} \in \{0,1\}^n$, with a *single* individual such that

$$(\mathcal{T}_{f^n}, \mathcal{A}_{\boldsymbol{\alpha}}) \models \boldsymbol{q}_{f^n} \qquad \text{iff} \qquad f^n(\boldsymbol{\alpha}) = 1, \qquad \text{for all } \boldsymbol{\alpha} \in \{0,1\}^n.$$

Then we show that rewritings for $\boldsymbol{q}_{f^n}$ and $\mathcal{T}_{f^n}$ correspond to Boolean circuits computing $f^n$. The construction proceeds in two steps: first, we represent the $f^n$ by polynomial-size CNFs (in a way similar to the Tseitin transformation [27]), and then encode those CNFs in terms of *OWL 2 QL* query answering.

Let $f^1, f^2, \dots$ be a family of Boolean functions in NP and $\mathbf{C}^1, \mathbf{C}^2, \dots$ be a family of circuits computing the $f^n$ (according to the definition above). We consider the inputs $\boldsymbol{x}$ and the advice inputs $\boldsymbol{y}$ of $\mathbf{C}^n$ as Boolean variables; each of the gates $g_1, \dots, g_{|\mathbf{C}^n|}$ of $\mathbf{C}^n$ is also thought of as a Boolean variable whose value coincides with the output of the gate on a given input. We assume that $\mathbf{C}^n$ only contains $\neg$- and $\land$-gates, and so can be regarded as a set of equations of the form

$$g_i = \neg h_i \qquad \text{or} \qquad g_i = h_i \land h_i',$$

where $h_i$ and $h_i'$ are the inputs of the gate $g_i$, that is, either input variables $\boldsymbol{x}$, advice variables $\boldsymbol{y}$ or other gates $\boldsymbol{g} = (g_1, \dots, g_{|\mathbf{C}^n|})$. We assume $g_{|\mathbf{C}^n|}$ to be the output of $\mathbf{C}^n$. Now, with each $f^n$ and each $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_n) \in \{0,1\}^n$, we associate the following formula in CNF:

$$\begin{aligned}\varphi^{\boldsymbol{\alpha}}_{f^n}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{g}) \quad = \quad & \bigwedge_{\alpha_j = 0} \neg x_j \quad \land g_{|\mathbf{C}^n|} \quad \land \bigwedge_{g_i = \neg h_i \text{ in } \mathbf{C}^n} \big[(h_i \lor \neg g_i) \land (\neg h_i \lor g_i)\big] \quad \land \\ & \bigwedge_{g_i = h_i \land h_i' \text{ in } \mathbf{C}^n} \big[(h_i \lor \neg g_i) \land (h_i' \lor \neg g_i) \land (\neg h_i \lor \neg h_i' \lor g_i)\big].\end{aligned}$$

The clauses of the last two conjuncts encode the correct computation of the circuit: they are equivalent to $g_i \leftrightarrow \neg h_i$ and $g_i \leftrightarrow h_i \land h_i'$, respectively.

**Lemma 1.** *If $f^n$ is a monotone Boolean function then $f^n(\boldsymbol{\alpha}) = 1$ iff $\varphi^{\boldsymbol{\alpha}}_{f^n}$ is satisfiable, for each $\boldsymbol{\alpha} \in \{0,1\}^n$.*

The second step of the reduction is to encode satisfiability of $\varphi^{\boldsymbol{\alpha}}_{f^n}$ by means of the CQ answering problem in *OWL 2 QL*. Denote $\varphi^{\boldsymbol{\alpha}}_{f^n}$ for $\boldsymbol{\alpha} = (0, \dots, 0)$ by $\varphi_{f^n}$. It is immediate from the definitions that, for each $\boldsymbol{\alpha} \in \{0,1\}^n$, the CNF $\varphi^{\boldsymbol{\alpha}}_{f^n}$ can be obtained from $\varphi_{f^n}$ by removing the clauses $\neg x_j$ for which $\alpha_j = 1$, $1 \leq j \leq n$. The CNF $\varphi_{f^n}$ contains $d \leq 3|\mathbf{C}^n|$ clauses $C_1, \dots, C_d$ with $N = |\mathbf{C}^n|$ Boolean variables, which will be denoted by $p_1, \dots, p_N$.

Let $P$ be a role name and let $A_i$, $X_i^0$, $X_i^1$ and $Z_{i,j}$ be concept names. Consider the TBox $\mathcal{T}_{f^n}$ containing the following inclusions, for $1 \leq i \leq N$, $1 \leq j \leq d$:

$$\begin{aligned}
& A_{i-1} \sqsubseteq \exists P^-.X_i^\ell, && X_i^\ell \sqsubseteq A_i, \quad \text{for } \ell = 0, 1,\\
& \quad X_i^0 \sqsubseteq Z_{i,j} \quad \text{if} \quad \neg p_i \in C_j, &&\\
& \quad X_i^1 \sqsubseteq Z_{i,j} \quad \text{if} \quad p_i \in C_j, &&\\
& Z_{i,j} \sqsubseteq \exists P.Z_{i-1,j}, &&\\
& A_0 \sqcap A_i \sqsubseteq \bot, && A_0 \sqcap \exists P \sqsubseteq \bot,\\
& A_0 \sqcap Z_{i,j} \sqsubseteq \bot, \quad \text{for } (i,j) \notin \{(0,1), \dots, (0,n)\}. &&
\end{aligned}$$

It is not hard to check that $|\mathcal{T}_{f^n}| = O(|\mathbf{C}^n|^2)$. Consider also the CQ

$$\boldsymbol{q}_{f^n} = \exists \boldsymbol{y}\, \exists \boldsymbol{z}\, \Big[ A_0(y_0) \land \bigwedge_{i=1}^{N} P(y_i, y_{i-1}) \land \\ \bigwedge_{j=1}^{d} \Big( P(y_N, z_{N-1,j}) \land \bigwedge_{i=1}^{N-1} P(z_{i,j}, z_{i-1,j}) \land Z_{0,j}(z_{0,j}) \Big) \Big],$$

where $\boldsymbol{y} = (y_0, \dots, y_N)$ and $\boldsymbol{z} = (z_{0,1}, \dots, z_{N-1,1}, \dots, z_{0,d}, \dots, z_{N-1,d})$. Clearly, $|\boldsymbol{q}_{f^n}| = O(|\mathbf{C}^n|^2)$. Note that $\mathcal{T}_{f^n}$ is acyclic and $\boldsymbol{q}_{f^n}$ is tree-shaped and has no answer variables. For each $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_n) \in \{0,1\}^n$, we set

$$\mathcal{A}_{\boldsymbol{\alpha}} \quad = \quad \big\{A_0(a)\big\} \quad \cup \quad \big\{Z_{0,j}(a) \mid 1 \le j \le n \text{ and } \alpha_j = 1\big\}.$$

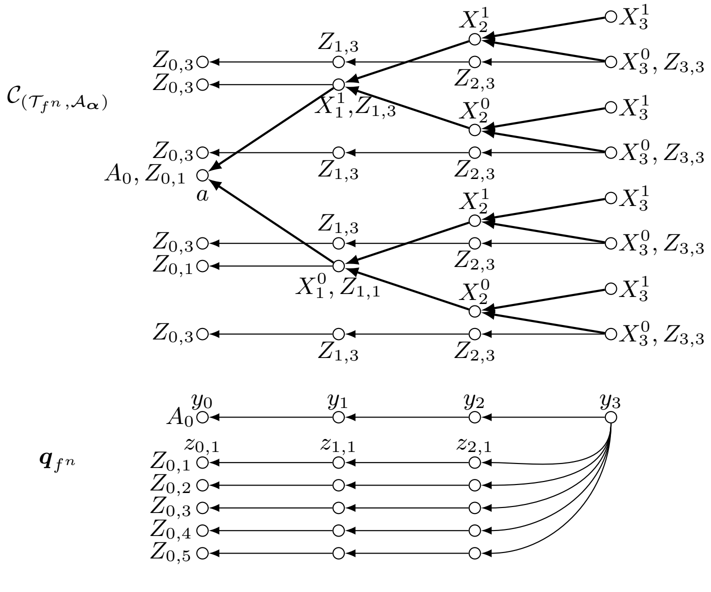

**Fig. 1.** Canonical model $\mathcal{C}_{(\mathcal{T}_{f^n}, \mathcal{A}_{\boldsymbol{\alpha}})}$ and query $\boldsymbol{q}_{f^n}$ for the Boolean function $f^n$, $n = 1$, computed by the circuit with one input $x$, one advice input $y$ and a single $\land$-gate. Thus, $N = 3$, $d = 5$ and $\varphi_{f^n}(x, y, g) = \neg x \land g \land (x \lor \neg g) \land (y \lor \neg g) \land (\neg x \lor \neg y \lor g)$. Points in $X_i^\ell$ are also in $A_i$, for all $1 \le i \le N$; the arrows denote role $P$ and the $Z_{i,j}$ branches in the canonical model are shown only for $j = 1, 3$, i.e., for $\neg x$ and $(x \lor \neg g)$.

We explain the intuition behind the $\mathcal{T}_{f^n}$, $\boldsymbol{q}_{f^n}$ and $\mathcal{A}_{\boldsymbol{\alpha}}$ using the example of Fig. 1, where the query $\boldsymbol{q}_{f^n}$ and the canonical model of $(\mathcal{T}_{f^n}, \mathcal{A}_{\boldsymbol{\alpha}})$, with $\mathcal{A}_{\boldsymbol{\alpha}} = \{A_0(a), Z_{0,1}(a)\}$, are illustrated for some Boolean function. To answer $\boldsymbol{q}_{f^n}$

in the canonical model, we have to check whether $\boldsymbol{q}_{f^n}$ can be homomorphically mapped into it. The variables $y_i$ are clearly mapped to one of the branches of the canonical model from $a$ to a point in $A_3$, say the lowest one, which corresponds to the valuation for the variables in $\varphi^{\boldsymbol{\alpha}}_{f^n}$ making all of them false. Now, there are two possible ways to map variables $z_{2,1}, z_{1,1}, z_{0,1}$ that correspond to the clause $C_1 = \neg x_1$ in $\varphi_{f^n}$. If they are sent to the same branch so that $z_{0,1} \mapsto a$ then $Z_{0,1}(a) \in \mathcal{A}_{\boldsymbol{\alpha}}$, whence the clause $C_1$ cannot be in $\varphi^{\boldsymbol{\alpha}}_{f^n}$. Otherwise, they are mapped to the points in a side-branch so that $z_{0,1} \not\mapsto a$, in which case $\neg x_1$ must be true under our valuation. Thus, we arrive at the following:

**Lemma 2.** *$(\mathcal{T}_{f^n}, \mathcal{A}_{\boldsymbol{\alpha}}) \models \boldsymbol{q}_{f^n}$ iff $\varphi^{\boldsymbol{\alpha}}_{f^n}$ is satisfiable, for all $\boldsymbol{\alpha} \in \{0,1\}^n$.*

We now use this result to reveal a close correspondence between PE-rewritings and monotone Boolean formulas, FO-rewritings and Boolean formulas, and between NDL-rewritings and monotone Boolean circuits.

**Lemma 3.** *Let $f^1, f^2, \dots$ be a family of monotone Boolean functions in* NP*, and let $f = f^n$, for some $n$.*

(i) *If $\boldsymbol{q}'_f$ is a PE-rewriting for $\boldsymbol{q}_f$ and $\mathcal{T}_f$ then there is a monotone Boolean formula $\psi_f$ computing $f$ with $|\psi_f| \leq |\boldsymbol{q}'_f|$.*

(ii) *If $\boldsymbol{q}'_f$ is an FO-rewriting for $\boldsymbol{q}_f$ and $\mathcal{T}_f$ over a signature with a single constant then there is a Boolean formula $\chi_f$ computing $f$ with $|\chi_f| \leq |\boldsymbol{q}'_f|$.*

(iii) *If $(\Pi_f, G)$ is an NDL-rewriting for $\boldsymbol{q}_f$ and $\mathcal{T}_f$ then there is a monotone Boolean circuit $\mathbf{C}_f$ computing $f$ with $|\mathbf{C}_f| \leq |\Pi_f|$.*

The proof proceeds by eliminating quantifiers from the given rewriting and replacing its predicates with propositional variables using the fact that, in the ABoxes $\mathcal{A}_{\boldsymbol{\alpha}}$, these predicates can only be true on the individual $a$. Lemmas 1 and 2 ensure that the resulting Boolean formula or circuit computes $f$.

The next lemma shows that, conversely, circuits computing $f$ can be turned into rewritings for $\boldsymbol{q}_f$ and $\mathcal{T}_f$ over ABoxes with a single individual.

**Lemma 4.** *Let $f^1, f^2, \dots$ be a family of monotone Boolean functions in* NP*, and let $f = f^n$, for some $n$. The following holds for signatures $\Sigma$ with a single constant:*

(i) *Suppose $\boldsymbol{q}'$ is an FO-sentence such that $(\mathcal{T}_f, \mathcal{A}_{\boldsymbol{\alpha}}) \models \boldsymbol{q}_f$ iff $\mathcal{I}_{\mathcal{A}_{\boldsymbol{\alpha}}} \models \boldsymbol{q}'$, for all $\boldsymbol{\alpha}$. Then*

$$\boldsymbol{q}'' \; = \; \exists x \Big[ A_0(x) \land \big(\boldsymbol{q}' \lor \bigvee_{A_0 \sqcap B \sqsubseteq_{\mathcal{T}_f} \bot} B(x)\big)\Big]$$

*is an FO-rewriting for $\boldsymbol{q}_f$ and $\mathcal{T}_f$ with $|\boldsymbol{q}''| = |\boldsymbol{q}'| + O(|\mathbf{C}^n|^2)$.*

(ii) *Suppose $(\Pi, G)$ is a pure NDL query with a propositional goal $G$ such that $(\mathcal{T}_f, \mathcal{A}_{\boldsymbol{\alpha}}) \models \boldsymbol{q}_f$ iff $\Pi, \mathcal{A}_{\boldsymbol{\alpha}} \models G$, for all $\boldsymbol{\alpha}$. Then $(\Pi', G')$ is an NDL-rewriting for $\mathcal{T}_f$ and $\boldsymbol{q}_f$ with $|\Pi'| = |\Pi| + O(|\mathbf{C}^n|^2)$, where $G'$ is a fresh propositional variable and $\Pi'$ is obtained by extending $\Pi$ with the following rules:*

- *$\forall x\, (A_0(x) \land G \to G')$,*
- *$\forall x\, (A_0(x) \land B(x) \to G')$, for all concepts $B$ such that $A_0 \sqcap B \sqsubseteq_{\mathcal{T}_f} \bot$.*

(In both statements above, $B(x)$ denotes $\exists y\, P(x,y)$ in the case of $B = \exists P$.)

We are in a position now to formulate our main theorem that connects the size of circuits computing monotone Boolean functions with the size of rewritings for the corresponding queries and ontologies. It follows from Lemmas 1–4.

**Theorem 2.** *For any family $f^1, f^2, \dots$ of monotone Boolean functions in* NP*, there exist polynomial-size CQs $\boldsymbol{q}_n$ and OWL 2 QL TBoxes $\mathcal{T}_n$ such that the following holds*:

(1) *Let $L(n)$ be a lower bound for the size of monotone Boolean formulas computing $f^n$. Then $|\boldsymbol{q}'_n| \geq L(n)$, for any PE-rewriting $\boldsymbol{q}'_n$ for $\boldsymbol{q}_n$ and $\mathcal{T}_n$.*

(2) *Let $L(n)$ and $U(n)$ be a lower and an upper bound for the size of monotone Boolean circuits computing $f^n$. Then*
  - *$|\Pi_n| \geq L(n)$, for any NDL-rewriting $(\Pi_n, G)$ for $\boldsymbol{q}_n$ and $\mathcal{T}_n$;*
  - *there exist a polynomial $p$ and an NDL-rewriting $(\Pi_n, G)$ for $\boldsymbol{q}_n$ and $\mathcal{T}_n$ over a signature with a single constant such that $|\Pi_n| \leq U(n) + p(n)$.*

(3) *Let $L(n)$ and $U(n)$ be a lower and an upper bound for the size of Boolean formulas computing $f^n$. Then*
  - *$|\boldsymbol{q}'_n| \geq L(n)$, for any FO-rewriting $\boldsymbol{q}'_n$ for $\boldsymbol{q}_n$ and $\mathcal{T}_n$ over any signature with a single constant*;
  - *there exist a polynomial $p$ and an FO-rewriting $\boldsymbol{q}'_n$ for $\boldsymbol{q}_n$ and $\mathcal{T}_n$ over a signature with a single constant such that $|\boldsymbol{q}'_n| \leq U(n) + p(n)$.*

## 4 Rewritings Long and Short

We apply Theorem 2 to three concrete families of Boolean functions and show that some queries and ontologies may only have very long rewritings, and some rewritings can be exponentially or superpolynomially more succinct than others.

First we prove that one cannot avoid an exponential blow-up for PE- and NDL-rewritings; moreover, even FO-rewritings can blow-up superpolynomially for signatures with a single constant under the assumption that NP $\not\subseteq$ P/poly (i.e., that NP-complete problems cannot be solved by polynomial-size circuits, which is an open problem; see, e.g., [3]). This can be done using the function $\textsc{Clique}_{n,k}$ of $n(n-1)/2$ variables $e_{ij}$, $1 \leq i < j \leq n$, which returns 1 iff the graph with vertices $\{1, \dots, n\}$ and edges $\{\{i,j\} \mid e_{ij} = 1\}$ contains a $k$-clique. A series of papers, started by Razborov's [22], 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 $k = 2n/3$ [21]. One can show that there is a nondeterministic circuit with $n$ advice inputs and $O(n^2)$ gates that computes $\textsc{Clique}_{n,k}$. As $\textsc{Clique}_{n,k}$ is NP-complete, the question whether $\textsc{Clique}_{n,k}$ can be computed by a polynomial-size deterministic circuit is equivalent to NP $\subseteq$ P/poly.

**Theorem 3.** *There is a sequence of CQs $\boldsymbol{q}_n$ of size $O(n)$ and OWL 2 QL TBoxes $\mathcal{T}_n$ of size $O(n)$ such that*:

- *any PE-rewriting for $\boldsymbol{q}_n$ and $\mathcal{T}_n$ is of size $\geq 2^{\Omega(n^{1/4})}$*;

- *any NDL-rewriting for $\boldsymbol{q}_n$ and $\mathcal{T}_n$ is of size* $\geq 2^{\Omega((n/\log n)^{1/12})}$;
- *there does not exist a polynomial-size FO-rewriting for $\boldsymbol{q}_n$ and $\mathcal{T}_n$ over a signature with a single constant unless* $\mathrm{NP} \subseteq \mathrm{P/poly}$.

By the Karp-Lipton theorem (see, e.g., [3]), $\mathrm{NP} \subseteq \mathrm{P/poly}$ implies $\mathrm{PH} = \Sigma_2^p$. So we can replace the assumption $\mathrm{NP} \not\subseteq \mathrm{P/poly}$ with $\mathrm{PH} \neq \Sigma_2^p$.

The next result shows that NDL-rewritings can be exponentially more succinct than PE-rewritings. Here we use the function $\textsc{Gen}_{n^3}$ of $n^3$ variables $x_{ijk}$, $1 \leq i, j, k \leq n$, defined as follows. We say that 1 *generates* $k \leq n$ if either $k = 1$ or $x_{ijk} = 1$ and 1 generates both $i$ and $j$. $\textsc{Gen}_{n^3}(x_{111}, \dots, x_{nnn})$ returns 1 iff 1 generates $n$. $\textsc{Gen}_{n^3}$ is clearly a monotone Boolean function computable by polynomial-size monotone circuits. On the other hand, any monotone formula computing $\textsc{Gen}_{n^3}$ is of size $2^{n^\varepsilon}$, for some $\varepsilon > 0$ [20].

**Theorem 4.** *There is a sequence of CQs $\boldsymbol{q}_n$ of size $O(n)$ and OWL 2 QL TBoxes $\mathcal{T}_n$ of size $O(n)$ for which there exists a polynomial-size NDL-rewriting over a signature with a single constant, but any PE-rewriting over this signature is of size $\geq 2^{n^\varepsilon}$, for some $\varepsilon > 0$.*

Finally, we show that FO-rewritings can be superpolynomially more succinct than PE-rewritings. We use the function $\textsc{Matching}_{2n}$ with $n^2$ variables $e_{ij}$, $1 \leq i, j \leq n$, which returns 1 iff there is a *perfect matching* in the bipartite graph $G$ with vertices $\{v_1^1, \dots, v_n^1, v_1^2, \dots, v_n^2\}$ and edges $\{\{v_i^1, v_j^2\} \mid e_{ij} = 1\}$, i.e., a subset $E$ of edges in $G$ such that every node in $G$ occurs exactly once in $E$. An exponential lower bound $2^{\Omega(n)}$ for the size of monotone formulas computing $\textsc{Matching}_{2n}$ was obtained in [21]. However, there are non-monotone formulas of size $n^{O(\log n)}$ computing this function [8]. On the other hand, it can also be computed by a nondeterministic circuit with $n^2$ advice inputs and $O(n^2)$ gates.

**Theorem 5.** *There is a sequence of CQs $\boldsymbol{q}_n$ of size $O(n)$ and OWL 2 QL TBoxes $\mathcal{T}_n$ of size $O(n)$ which has a polynomial-size FO-rewriting over a signature with a single constant, but any PE-rewriting over this signature is of size $\geq 2^{\Omega(2^{\log^{1/2} n})}$.*

In the proof of Theorem 3, we used the CQs $\boldsymbol{q}_n = \boldsymbol{q}_{\textsc{Clique}_{m,k}}$ containing no constant symbols. It follows that the theorem will still hold if we allow the built-in predicates $=$ and $\neq$ in the rewritings, but disallow the use of constants that *do not occur in the original query.* The situation changes drastically if $=$, $\neq$ and two additional constants, say 0 and 1, are allowed in the rewritings. As shown by Gottlob and Schwentick [13], in this case there is a polynomial-size NDL-rewriting for any CQ and *OWL 2 QL* TBox. Roughly, the rewriting uses the extra resources to encode in a succinct way the part of the canonical model that is relevant to answering the given query. We call rewritings of this kind *impure* (indicating thereby that they use predicates and constants that do not occur in the original query and ontology). In fact, using the ideas of [5] and [13], one can construct an impure polynomial-size PE-rewriting for any CQ and *OWL 2 QL* TBox. Thus, we obtain the following:

**Theorem 6.** *Impure PE- and NDL-rewritings for CQs and OWL 2 QL ontologies are exponentially more succinct than pure PE- and NDL-rewritings.*

The difference between short impure and long pure rewritings appears to be of the same kind as the difference between deterministic and nondeterministic Boolean circuits: the impure rewritings can guess (using =, 0 and 1) what the pure ones must specify explicitly. It is not clear, however, how the RDBMSs are going to cope with such guesses in practice.

## 5 Conclusion

The exponential lower bounds for the size of 'pure' rewritings above may look discouraging in the OBDA context. It is to be noted, however, that the ontologies and queries used in their proofs are extremely 'artificial' and never occur in practice (see the analysis in [16]). As demonstrated by the existing description logic reasoners (such as FaCT++, HermiT, Pellet, Racer), *real-world* ontologies can be classified efficiently despite the high worst-case complexity of the classification problem. We believe that practical query answering over *OWL 2 QL* ontologies can be feasible if supported by suitable optimisation and indexing techniques. It also remains to be seen whether polynomial impure rewritings can be used in practice. We conclude the paper by mentioning two open problems. Our exponential lower bounds were proved for a sequence of pairs $(\boldsymbol{q}_n, \mathcal{T}_n)$. It is unclear whether these bounds hold uniformly for all $\boldsymbol{q}_n$ over the same $\mathcal{T}$:

*Question 1.* Do there exist an *OWL 2 QL* TBox $\mathcal{T}$ and CQs $\boldsymbol{q}_n$ such that any pure PE- or NDL-rewritings for $\boldsymbol{q}_n$ and $\mathcal{T}$ are of exponential size?

As we saw, both FO- and NDL-rewritings are more succinct than PE-rewritings.

*Question 2.* What is the relation between the size of FO- and NDL-rewritings?

**Acknowledgments.** We thank the anonymous referees for their constructive feedback and suggestions. This paper was supported by the U.K. EPSRC grant EP/H05099X.

## References

1. Acciarri, A., Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Palmieri, M., Rosati, R.: QuOnto: Querying ontologies. In: Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI 2005). pp. 1670–1671 (2005)
2. Alon, N., Boppana, R.: The monotone circuit complexity of Boolean functions. Combinatorica 7(1), 1–22 (1987)
3. Arora, S., Barak, B.: Computational Complexity: A Modern Approach. Cambridge University Press, New York, NY, USA, 1st edn. (2009)
4. Artale, A., Calvanese, D., Kontchakov, R., Zakharyaschev, M.: The DL-Lite family and relations. J. of Artificial Intelligence Research (JAIR) 36, 1–69 (2009)

5. Avigad, J.: Eliminating definitions and Skolem functions in first-order logic. In: Proc. of LICS. pp. 139–146 (2001)
6. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.): The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press (2003)
7. Benedikt, M., Gottlob, G.: The impact of virtual views on containment. PVLDB 3(1), 297–308 (2010)
8. Borodin, A., von zur Gathen, J., Hopcroft, J.E.: Fast parallel matrix and gcd computations. In: Proc. of FOCS. pp. 65–71 (1982)
9. Calì, A., Gottlob, G., Lukasiewicz, T.: A general Datalog-based framework for tractable query answering over ontologies. In: Proc. of PODS. pp. 77–86 (2009)
10. Calì, A., Gottlob, G., Pieris, A.: Advanced processing for ontological queries. PVLDB 3(1), 554–565 (2010)
11. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The DL-Lite family. J. of Automated Reasoning 39(3), 385–429 (2007)
12. Gottlob, G., Orsi, G., Pieris, A.: Ontological queries: Rewriting and optimization. In: Proc. of the IEEE Int. Conf. on Data Engineering, ICDE (2011)
13. Gottlob, G., Schwentick, T.: Rewriting ontological queries into small nonrecursive Datalog programs. In: Proc. of DL 2011. vol. 745. CEUR-WS.org (2011)
14. Jukna, S.: Boolean Function Complexity: Advances and Frontiers. Springer (2012)
15. Kikot, S., Kontchakov, R., Zakharyaschev, M.: On (in)tractability of OBDA with OWL 2 QL. In: Proc. of DL 2011. vol. 745. CEUR-WS.org (2011)
16. Kikot, S., Kontchakov, R., Zakharyaschev, M.: Conjunctive query answering with OWL 2 QL. In: Proc. of the 13th Int. Conf. KR 2012. AAAI Press (2012)
17. Kontchakov, R., Lutz, C., Toman, D., Wolter, F., Zakharyaschev, M.: The combined approach to query answering in DL-Lite. In: Proc. of the 12th Int. Conf. KR 2010. AAAI Press (2010)
18. Lutz, C., Toman, D., Wolter, F.: Conjunctive query answering in the description logic EL using a relational database system. In: Proc. of the 21st Int. Joint Conf. on Artificial Intelligence, IJCAI 2009. pp. 2070–2075 (2009)
19. Pérez-Urbina, H., Motik, B., Horrocks, I.: A comparison of query rewriting techniques for DL-Lite. In: Proc. of DL 2009. vol. 477. CEUR-WS.org (2009)
20. Raz, R., McKenzie, P.: Separation of the monotone NC hierarchy. In: Proc. of FOCS. pp. 234–243 (1997)
21. Raz, R., Wigderson, A.: Monotone circuits for matching require linear depth. J. ACM 39(3), 736–744 (1992)
22. Razborov, A.: Lower bounds for the monotone complexity of some Boolean functions. Dokl. Akad. Nauk SSSR 281(4), 798–801 (1985)
23. Rodríguez-Muro, M., Calvanese, D.: Dependencies to optimize ontology based data access. In: Proc. of DL 2011. vol. 745. CEUR-WS.org (2011)
24. Rodríguez-Muro, M., Calvanese, D.: Semantic index: Scalable query answering without forward chaining or exponential rewritings. In: Proc. of the 10th Int. Semantic Web Conf., ISWC (2011)
25. Rodríguez-Muro, M., Calvanese, D.: High performance query answering over DL-Lite ontologies. In: Proc. of the 13th Int. Conf. KR 2012. AAAI Press (2012)
26. Rosati, R., Almatelli, A.: Improving query answering over DL-Lite ontologies. In: Proc. of the 12th Int. Conf. KR 2010. AAAI Press (2010)
27. Tseitin., G.: On the complexity of derivation in propositional calculus. In: Automation of Reasoning 2: Classical Papers on Computational Logic 1967–1970 (1983)