The Price of Query Rewriting in Ontology-Based Data Access

Georg Gottlob[a], Stanislav Kikot[b], Roman Kontchakov[b], Vladimir Podolskii[c], Thomas Schwentick[d], Michael Zakharyaschev[b,*]

[a]*Department of Computer Science, University of Oxford, U.K.*
[b]*Department of Computer Science and Information Systems, Birkbeck, University of London, U.K.*
[c]*Steklov Mathematical Institute, Moscow, Russia*
[d]*Fakultät für Informatik, TU Dortmund, Germany*

Abstract

We give a solution to the succinctness problem for the size of first-order rewritings of conjunctive queries in ontology-based data access with ontology languages such as *OWL 2 QL*, linear Datalog$^{\pm}$ and sticky Datalog$^{\pm}$. We show that positive existential and nonrecursive datalog rewritings, which do not use extra non-logical symbols (except for intensional predicates in the case of datalog rewritings), suffer an exponential blowup in the worst case, while first-order rewritings can grow superpolynomially unless NP $\subseteq$ P/poly. We also prove that 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. On the other hand, we construct polynomial-size positive existential and nonrecursive datalog rewritings under the assumption that any data instance contains two fixed constants.

Keywords: Ontology, datalog, conjunctive query, query rewriting, succinctness, Boolean circuit, monotone complexity.

1. Introduction

Our aim in this article is to give a solution to the succinctness problem for various types of conjunctive query rewriting in ontology-based data access (OBDA) with basic ontology languages such as *OWL 2 QL* and fragments of Datalog$^{\pm}$.

The idea of OBDA has been around since about 2005 [14, 19, 28, 47]. In the OBDA paradigm, an ontology defines a high-level global schema and provides a vocabulary for user queries. An OBDA system rewrites these queries into the vocabulary of the data and then delegates the actual query evaluation to the data sources (which can be relational databases, triple stores, datalog engines, etc.). OBDA is often regarded as an important ingredient of the new generation of information systems because it (*i*) gives a high-level conceptual view of the data, (*ii*) provides the users with a convenient vocabulary for queries, thus isolating them from the details of the structure of data sources, (*iii*) allows the system to enrich incomplete data with background knowledge, and (*iv*) supports queries to multiple and possibly heterogeneous data sources.

A key concept of OBDA is first-order (FO) rewritability. An ontology language $\mathcal{L}$ is said to enjoy *FO-rewritability* if any conjunctive query (CQ) $\boldsymbol{q}$ over any ontology Σ, formulated in $\mathcal{L}$, can be rewritten to an FO-query $\boldsymbol{q}'$ such that, for any data instance D, the answers to the original CQ $\boldsymbol{q}$ over the knowledge base (Σ, D) can be computed by evaluating the rewriting $\boldsymbol{q}'$ over D. As $\boldsymbol{q}'$ is an FO-query, the answers to $\boldsymbol{q}'$ can be obtained 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 description logics of the *DL-Lite* family [16, 4], and fragments

*Corresponding author

Email addresses: `georg.gottlob@cs.ox.ac.uk` (Georg Gottlob), `staskikotx@gmail.com` (Stanislav Kikot), `roman@dcs.bbk.ac.uk` (Roman Kontchakov), `podolskii.vv@gmail.com` (Vladimir Podolskii), `thomas.schwentick@udo.edu` (Thomas Schwentick), `michael@dcs.bbk.ac.uk` (Michael Zakharyaschev)

of Datalog$^{\pm}$ such as linear tgds [11] (also known as atomic-body existential rules [6]) or sticky tgds [12, 13]. To illustrate, consider an *OWL 2 QL*-ontology Σ consisting of the following tuple-generating dependencies (tgds):

$$\forall x\,(RA(x) \to \exists y\,(worksOn(x,y) \land Project(y))), \tag{1}$$

$$\forall x\,(Project(x) \to \exists y\,(isManagedBy(x,y) \land Professor(y))), \tag{2}$$

$$\forall x, y\,(worksOn(x,y) \to involves(y,x)), \tag{3}$$

$$\forall x, y\,(isManagedBy(x,y) \to involves(x,y)), \tag{4}$$

and the CQ $\boldsymbol{q}(x)$ asking to find those who work with professors:

$$\boldsymbol{q}(x) = \exists y, z\,(worksOn(x,y) \land involves(y,z) \land Professor(z)). \tag{5}$$

A moment's thought should convince the reader that the (positive existential) query

$$\begin{aligned}\boldsymbol{q}'(x) \quad = \quad & \exists y, z\,[worksOn(x,y) \land (worksOn(z,y) \lor isManagedBy(y,z) \lor involves(y,z)) \land Professor(z)] \lor \\ & \exists y\,[worksOn(x,y) \land Project(y)] \quad \lor \quad RA(x)\end{aligned}$$

is an FO-rewriting of $\boldsymbol{q}(x)$ and Σ in the sense that, for any set D of ground atoms and any constant a in D, we have

$$(\Sigma, D) \models \boldsymbol{q}(a) \qquad \text{if and only if} \qquad D \models \boldsymbol{q}'(a).$$

(In Section 2, we shall consider this example in more detail.) A number of different rewriting techniques have been proposed and implemented for *OWL 2 QL* (PerfectRef [47], Presto/Prexto [55, 54], Rapid [18], the combined approach [37], Ontop [51, 33]) and its various extensions (Requiem/Blackout [45, 46], Nyaya [25, 43], Clipper [20] and [35]). However, all FO-rewritings constructed so far have, in the worst case, been exponential in the size of the query $\boldsymbol{q}$. Thus, despite the fact that, for data complexity, CQ answering over ontologies with FO-rewritability is as complex as standard database query evaluation (both are in AC^0), rewritings can be too large for RDBMSs to cope with. It has become apparent, in both theory and experiments, that for the OBDA paradigm to work in practice, we have to restrict attention to those ontologies and CQs that ensure *polynomial FO-rewritability* (in the very least).

The major open question we are going to attack in this article is whether the standard ontology languages for OBDA (in particular, *OWL 2 QL*) enjoy polynomial FO-rewritability. Naturally, the answer depends on what means we can use in the rewritings. For example, in the rewriting $\boldsymbol{q}'$ of $\boldsymbol{q}$ and Σ above, we did not use any non-logical symbols other than those that occurred in $\boldsymbol{q}$ and Σ. Such rewritings (perhaps also containing equality) may be described as 'pure' as they can be used with all possible databases; cf. [16]. (Note that all known rewritings apart from the one in the combined approach [37] are pure in this sense.) Other important parameters are the available logical means (connectives and quantifiers) in rewritings and the way we represent them. Apart from the class of arbitrary FO-queries, we shall also consider positive existential (PE) queries and nonrecursive datalog (NDL) queries as possible formalisms for rewritings (needless to say that pure NDL-rewritings may contain new intensional predicates).

At first sight, the results we obtain in this article could be divided into negative and positive. The bad news is that there is a sequence of CQs $\boldsymbol{q}_n$ and *OWL 2 QL* ontologies Σ_n, both of size $O(n)$, such that any pure PE- or NDL-rewriting of $\boldsymbol{q}_n$ and Σ_n is of exponential size in n, while any pure FO-rewriting is of superpolynomial size unless $\mathrm{NP} \subseteq \mathrm{P/poly}$. We obtain this negative result by first showing that OBDA with *OWL 2 QL* is powerful enough to compute monotone Boolean functions in NP, and that PE-rewritings correspond to monotone Boolean formulas, NDL-rewritings to monotone Boolean circuits, and FO-rewritings to arbitrary Boolean formulas. Then we use the celebrated exponential lower bounds for the size of monotone circuits and formulas computing the (NP-complete) Boolean function $\textsc{Clique}_{n,k}$ 'a graph with n nodes contains a k-clique' [50, 49]; a superpolynomial lower bound for the size of arbitrary (not necessarily monotone) Boolean formulas computing $\textsc{Clique}_{n,k}$ is a consequence of the assumption $\mathrm{NP} \not\subseteq \mathrm{P/poly}$. We also use known separation results [49, 48] for monotone Boolean functions such as 'a bipartite graph with n vertices in each part has a perfect matching' and 'a given vertex is accessible in a path accessibility system with n vertices' to show that pure NDL-rewritings are in general exponentially more succinct than pure PE-rewritings, while pure FO-rewritings can be superpolynomially more succinct than pure PE-rewritings.

On the other hand, we have some good news as well: assuming that every data instance contains *two* fixed distinct individual constants, we construct polynomial-size *impure* PE- and NDL-rewritings of any CQ and any ontology with the polynomial witness property (in particular, any ontology in *OWL 2 QL*, linear Datalog$^{\pm}$ of bounded arity or sticky Datalog$^{\pm}$ of bounded arity). In essence, the rewriting guesses a polynomial number of ground atoms with database individuals and labelled nulls (encoded as tuples over the two fixed constants), and checks whether these atoms satisfy the given CQ and form a sequence of chase steps. We first construct a polynomial-size impure PE-rewriting and then show how its disjunctions can be encoded by a polynomial-size NDL-rewriting with intensional predicates of small arity. As the two constants in the impure PE-rewriting can be replaced with two fresh existentially quantified variables, say x and y, such that $x \neq y$, we also obtain a polynomial-size *pure* FO-rewriting over data instances with *at least two domain elements*.

How to reconcile these seemingly contradictory results? To establish exponential and superpolynomial lower bounds for the size of pure rewritings, we show that computing monotone Boolean functions in NP is polynomially reducible to answering CQs over *OWL 2 QL*-ontologies and data instances with a *single individual*. As evaluating queries over such data instances is tractable, *pure* rewritings of the CQs and ontologies computing NP-complete monotone Boolean functions such as $\textsc{Clique}_{n,k}$ cannot be constructed in polynomial time—unless P = NP. (Our argument in Section 3 is a bit subtler: we prove that pure polynomial rewritings of the CQs and ontologies computing NP-complete monotone Boolean functions do not actually *exist*.) In fact, standard pure rewritings represent explicitly all distinct homomorphisms of the given CQ into the *labelled nulls* of possible chases for the given ontology, and our construction shows that there may be exponentially-many such homomorphisms. On the other hand, our *impure* rewritings employ polynomially-many additional existential quantifiers over two fixed distinct domain elements in order to *guess* those homomorphisms. Thus, we show that the additional NP-overhead of OBDA compared to CQ evaluation over plain databases can be represented in a succinct way. The exponential succinctness of impure rewritings compared to pure ones is of the same kind as the succinctness of nondeterministic finite automata or $\exists$-QBFs compared to deterministic automata [42] or, respectively, SAT (cf. also [5]).

The plan of the article is as follows. In Section 2, we introduce *OWL 2 QL*, linear and sticky Datalog$^{\pm}$ as fragments of the language of tuple-generating dependencies and illustrate the construction of an FO-rewriting for *OWL 2 QL*-ontologies. We also introduce nonrecursive datalog rewritings and formulate the succinctness and separation problems. The exponential and superpolynomial lower bounds on the size of pure rewritings are obtained in Section 3. The polynomial-size impure PE- and NDL-rewritings for families of ontologies with the polynomial witness property are constructed in Section 4. We prove the separation results mentioned above in Section 5. Open problems and directions for future research are discussed in Section 6.

Some of the results in this article first appeared in the conference proceedings [26, 32].

2. First-Order Rewritability: Size of Rewritings Matters

Let $\mathcal{R}$ be a relational schema. Given a data instance D over $\mathcal{R}$, we denote by Δ_D the set of individual constants in D. We regard D as a (finite) set of ground atoms. A *conjunctive query* (*CQ*, for short) $\boldsymbol{q}(\boldsymbol{x})$ is a formula of the form $\exists \boldsymbol{y}\, \varphi(\boldsymbol{x}, \boldsymbol{y})$, where φ is a conjunction of atoms $P(\boldsymbol{t})$ over $\mathcal{R}$ extended with equality, and each t in $\boldsymbol{t}$ is a *term* (an individual constant or a variable from $\boldsymbol{x}, \boldsymbol{y}$). The *size* $|\boldsymbol{q}|$ of a CQ $\boldsymbol{q}$ is the number of symbols in $\boldsymbol{q}$.

Let Σ be a set of first-order sentences over $\mathcal{R}$. The pair (Σ, D) is called a *knowledge base* (KB, for short). A tuple $\boldsymbol{a}$ of elements in Δ_D is said to be a *certain answer* to $\boldsymbol{q}(\boldsymbol{x})$ over the KB (Σ, D) if $\mathfrak{M} \models \boldsymbol{q}(\boldsymbol{a})$ for every model $\mathfrak{M}$ of $\Sigma \cup D$; in this case we write $(\Sigma, D) \models \boldsymbol{q}(\boldsymbol{a})$. If the tuple $\boldsymbol{x}$ of *answer variables* is empty, a *certain answer* to $\boldsymbol{q}$ over (Σ, D) is 'yes' in case $\mathfrak{M} \models \boldsymbol{q}$ for every model $\mathfrak{M}$ of $\Sigma \cup D$, and 'no' otherwise. CQs without answer variables are called *Boolean* CQs.

For the purposes of OBDA, we are interested in ontologies (or theories) Σ for which the problem of finding certain answers can be reduced to standard database query evaluation. More precisely, a first-order formula $\boldsymbol{q}'(\boldsymbol{x})$ is called a *first-order rewriting* of $\boldsymbol{q}$ and Σ (*FO-rewriting*, for short) if, for any data instance D, a tuple $\boldsymbol{a}$ of elements in Δ_D is a certain answer to $\boldsymbol{q}(\boldsymbol{x})$ over (Σ, D) just in case $\boldsymbol{a}$ is an answer to $\boldsymbol{q}'(\boldsymbol{x})$ over D. We say that Σ enjoys *first-order rewritability* if, for any CQ $\boldsymbol{q}(\boldsymbol{x})$, there exists an FO-rewriting of $\boldsymbol{q}$ and Σ.

There are two types of recognised ontology languages that guarantee first-order rewritability. The languages of the first type were introduced by the description logic community; they are based on the *DL-Lite* family of description logics [16, 4] and include the *OWL 2 QL* profile of the Web Ontology Language *OWL 2*.[1] The languages of the second

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

type were designed by the datalog community; they belong to the Datalog$^\pm$ family [12, 11] and are also known as existential rules [7]. All of these ontology languages can be formulated in terms of tuple-generating dependencies.

We remind the reader [1] that a *tuple-generating dependency* (a *tgd*, for short) is a first-order sentence of the form

$$\forall \boldsymbol{x}\, (\varphi(\boldsymbol{x}) \to \exists \boldsymbol{y}\, \psi(\boldsymbol{x}, \boldsymbol{y})), \tag{6}$$

where $\varphi(\boldsymbol{x})$, the *body*, and $\psi(\boldsymbol{x}, \boldsymbol{y})$, the *head* of the tgd, are conjunctions of atoms and all the variables in $\boldsymbol{x}$ actually occur in $\varphi(\boldsymbol{x})$ (note that both $\varphi(\boldsymbol{x})$ and $\psi(\boldsymbol{x}, \boldsymbol{y})$ can contain individual constants). Following the description logic tradition, we also consider *negative constraints* of the form

$$\forall \boldsymbol{x}\, (\varphi(\boldsymbol{x}) \to \bot).$$

Finite sets of tgds and negative constraints will be called *ontologies*. (Note that ontologies can be inconsistent.) Given an ontology Σ, we denote by $|\Sigma|$ its *size*, that is, the number of symbols in Σ.

An important property of tgds is the well-known fact [1] that, for any ontology Σ and any consistent KB (Σ, D), there exists a (possibly infinite) model $\mathfrak{C}_{\Sigma,D}$ of (Σ, D), known as a *universal* (or *canonical*) *model* of (Σ, D), such that, for any CQ $\boldsymbol{q}(\boldsymbol{x})$ and any tuple $\boldsymbol{a}$ from Δ_D, we have $(\Sigma, D) \models \boldsymbol{q}(\boldsymbol{a})$ if and only if $\mathfrak{C}_{\Sigma,D} \models \boldsymbol{q}(\boldsymbol{a})$. Such a universal model can be constructed by the following (*oblivious*) *chase procedure*, which, intuitively, 'repairs' D with respect to Σ (but not in the most economical way). We require the following definitions to describe the chase procedure formally. Let $\mathfrak{C}$ be a set of ground atoms and $\varphi(\boldsymbol{x})$ a conjunction of atoms (the body of a tgd or a negative constraint). We say that a map h from $\boldsymbol{x}$ to the individual constants in $\mathfrak{C}$ is a *homomorphism* from $\varphi(\boldsymbol{x})$ to $\mathfrak{C}$ if $h(\varphi(\boldsymbol{x})) \subseteq \mathfrak{C}$, where $h(\varphi(\boldsymbol{x}))$ denotes the set of atoms $P(h(\boldsymbol{t}))$, for $P(\boldsymbol{t})$ in $\varphi(\boldsymbol{x})$ (as usual, we assume that $h(a) = a$, for any individual constant a). We say that $\mathfrak{C}$ is *consistent with* Σ if there is no negative constraint $\forall \boldsymbol{x}\, (\varphi(\boldsymbol{x}) \to \bot)$ in Σ with a homomorphism h from $\varphi(\boldsymbol{x})$ to $\mathfrak{C}$.

The chase algorithm initially sets $\mathfrak{C}^0_{\Sigma,D} = D$. Suppose now that $\mathfrak{C}^{k-1}_{\Sigma,D}$ has already been defined. A tgd τ of the form $\forall \boldsymbol{x}\, (\varphi(\boldsymbol{x}) \to \exists \boldsymbol{y}\, \psi(\boldsymbol{x}, \boldsymbol{y}))$ is said to be *applicable to* $\mathfrak{C}^{k-1}_{\Sigma,D}$ *via* h if h is a homomorphism from $\varphi(\boldsymbol{x})$ to $\mathfrak{C}^{k-1}_{\Sigma,D}$ with either $k = 1$ or $h(\varphi(\boldsymbol{x})) \nsubseteq \mathfrak{C}^{k-2}_{\Sigma,D}$. Define an extension h' of h by taking $h'(x) = h(x)$ for every x in $\boldsymbol{x}$, and $h'(y) = c_y$ for every y in $\boldsymbol{y}$, where c_y is a fresh individual constant (a *labelled null*) different from all constants already used in the construction. An *application of* τ *under* h *to* $\mathfrak{C}^{k-1}_{\Sigma,D}$ adds the ground atoms of $h'(\psi(\boldsymbol{x}, \boldsymbol{y}))$ to $\mathfrak{C}^{k-1}_{\Sigma,D}$ if they are not there yet. If $\mathfrak{C}^{k-1}_{\Sigma,D}$ is consistent with Σ, the algorithm constructs $\mathfrak{C}^{k}_{\Sigma,D}$ as follows: it takes some enumeration of all distinct pairs (τ_i, h_i), $i \le n$, such that $\tau_i \in \Sigma$ is applicable to $\mathfrak{C}^{k-1}_{\Sigma,D}$ via h_i, and sets $\mathfrak{C}^{k}_{\Sigma,D}$ to be the result of applying each τ_i under h_i to $\mathfrak{C}^{k-1}_{\Sigma,D}$. The *chase* $\mathfrak{C}_{\Sigma,D}$ of (Σ, D) is the union of all $\mathfrak{C}^{k}_{\Sigma,D}$ for $k < \omega$, provided that the $\mathfrak{C}^{k}_{\Sigma,D}$ are consistent with Σ.

For example, Fig. 1 shows the chase $\mathfrak{C}_{\Sigma,D}$ for the ontology Σ consisting of the tgds (1)–(4) from the introduction and the data instance $D = \{\, \mathit{RA}(ck),\ \mathit{worksOn}(ck, e),\ \mathit{Project}(e),\ \mathit{isManagedBy}(e, gg)\,\}$ (note that, in general, the chase is not necessarily finite).

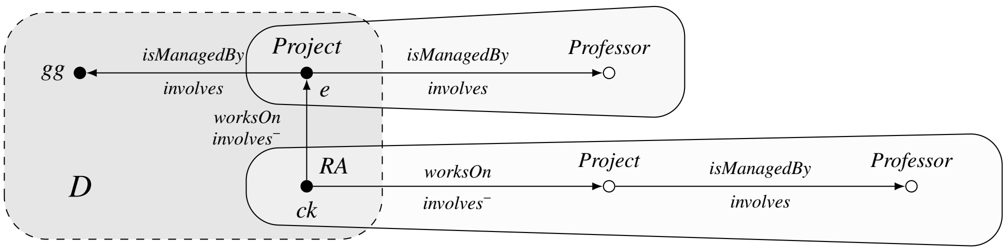

Figure 1: The chase $\mathfrak{C}_{\Sigma,D}$ for $\Sigma = \{(1), \dots, (4)\}$ and $D = \{\, \mathit{RA}(ck),\ \mathit{worksOn}(ck, e),\ \mathit{Project}(e),\ \mathit{isManagedBy}(e, gg)\,\}$.

The model $\mathfrak{C}_{\Sigma,D}$ is called *universal* because, for any model $\mathfrak{M}$ of (Σ, D), there is a homomorphism from $\mathfrak{C}_{\Sigma,D}$ to $\mathfrak{M}$. It is this property of the universal models that makes sure that all certain answers to CQs over (Σ, D) are contained in $\mathfrak{C}_{\Sigma,D}$. Furthermore, we say that an ontology has the *bounded derivation depth property* (*BDDP*, for short) if there is a function $d\colon \mathbb{N} \to \mathbb{N}$ such that, for any CQ $\boldsymbol{q}(\boldsymbol{x})$ and any data instance D, a tuple $\boldsymbol{a}$ from Δ_D is a certain answer to $\boldsymbol{q}$ over (Σ, D) if and only if $\mathfrak{C}^{d(|\boldsymbol{q}|)}_{\Sigma,D} \models \boldsymbol{q}(\boldsymbol{a})$. (Note that $d(|\boldsymbol{q}|)$ does not depend on D but can depend on Σ.) The following theorem gives a characterisation of ontologies enjoying FO-rewritability:

Theorem 1. *An ontology has the BDDP if and only if it enjoys FO-rewritability.*

Proof. For a proof of ($\Rightarrow$) see [11, Theorem 9]. To show ($\Leftarrow$), we use [9, Proposition 4] (based on [56]) according to which, whenever there is an FO-rewriting of $\boldsymbol{q}(\boldsymbol{x})$ and Σ, there is also a rewriting of the form $\boldsymbol{q}'(\boldsymbol{x}) = \bigvee_i \exists \boldsymbol{y}_i\, \varphi_i(\boldsymbol{x}, \boldsymbol{y}_i)$, where each $\exists \boldsymbol{y}_i\, \varphi_i(\boldsymbol{x}, \boldsymbol{y}_i)$ is a CQ. Let k be the maximum number of atoms in the CQs $\exists \boldsymbol{y}_i\, \varphi_i(\boldsymbol{x}, \boldsymbol{y}_i)$, which depends only on $\boldsymbol{q}$ (for a fixed Σ). Clearly, every answer $\boldsymbol{a}$ to $\boldsymbol{q}'(\boldsymbol{x})$ over D is also an answer to $\boldsymbol{q}'(\boldsymbol{x})$ over some subset $D' \subseteq D$ with $|D'| \leq k$. It follows that $\mathfrak{C}_{\Sigma,D} \models \boldsymbol{q}(\boldsymbol{a})$ if and only if $\mathfrak{C}_{\Sigma,D'} \models \boldsymbol{q}(\boldsymbol{a})$ for some $D' \subseteq D$ with $|D'| \leq k$. Observe that the number of pairwise non-isomorphic D with $|D| \leq k$ is finite and depends only on $\boldsymbol{q}$ (for a fixed Σ). Thus, we can take $d(|\boldsymbol{q}|)$ to be a number d such that $\mathfrak{C}^d_{\Sigma,D} \models \boldsymbol{q}(\boldsymbol{a})$ whenever $\mathfrak{C}_{\Sigma,D} \models \boldsymbol{q}(\boldsymbol{a})$, for any D with $|D| \leq k$. ❑

Disjunctions of CQs, used in the proof of Theorem 1, are known as *unions of conjunctive queries* or *UCQs*, for short. An FO-rewriting of $\boldsymbol{q}$ and Σ in the form of a UCQ is called a *UCQ-rewriting* of $\boldsymbol{q}$ and Σ. (That the BDDP of Σ is equivalent to the existence of UCQ-rewritings for all CQs over Σ can be shown using an earlier result from graph databases [57] and the fact that minimal UCQ-rewritings are unique up to isomorphism [36]; an ontology with UCQ-rewritings for all CQs is called a *finite unification set* by Baget *et al.* [6].)

The following ontology languages ensure the BDDP:

– *linear tgds* [11], that is, tgds with a single atom in the body;

– *OWL 2 QL-tgds*, that is, linear tgds with atoms of arity ≤ 2 and without individual constants;

– *sticky sets of tgds* [13], that is, sets of tgds such that the variables that appear more than once in the body of a tgd (join variables) are propagated (or 'stick') during the chase to all the inferred atoms

(other examples include sticky-join sets of tgds [13] and domain-restricted rules [7]). Each of the above ontology languages can also include negative constraints; they do not affect the chase procedure but can make a knowledge base inconsistent [11].

Remark 2. It is not hard to see that the standard *OWL 2 QL* profile of the Web Ontology Language *OWL 2* can be represented in terms of *OWL 2 QL*-tgds and negative constraints, but not the other way round: for example, the tgd $\forall x\, (R(x, x) \to A(x))$ cannot be expressed in *OWL 2 QL*. However, all the *OWL 2 QL*-tgds and negative constraints we use in this article are expressible in *OWL 2 QL*. Thus, the linear tgd of the form

$$\forall x\, (A(x) \to \exists y\, (R(x, y) \land B(y)))$$

used in (1) and (2) as well as in the construction of Section 3 can be encoded by the concept inclusion $A \sqsubseteq \exists R.B$ in the *OWL 2 QL* description logic syntax (where A and B are concept names and R is a role name), or as the following set of concept and role inclusions in the syntax of $\textit{DL-Lite}^{\mathcal{H}}_{core}$ [4]:

$$A \sqsubseteq \exists R_B, \qquad \exists R_B^- \sqsubseteq B, \qquad R_B \sqsubseteq R,$$

where R_B is a fresh role name. Because of this, we slightly abuse terminology and call ontologies with *OWL 2 QL*-tgds simply *OWL 2 QL-ontologies*.

We now give an example showing how one can construct FO-rewritings of CQs and *OWL 2 QL*-ontologies.

Example 3. Consider again the *OWL 2 QL*-ontology $\Sigma = \{(1), \dots, (4)\}$ and the CQ (5) from the introduction. Suppose $a \in \Delta_D$ is a certain answer to $\boldsymbol{q}(x)$ over (Σ, D), for some data instance D. This means that $\mathfrak{C}_{\Sigma,D} \models \boldsymbol{q}(a)$, and so there is a homomorphism h from $\boldsymbol{q}(x)$ to $\mathfrak{C}_{\Sigma,D}$ with $h(x) = a$. We construct an FO-rewriting $\boldsymbol{q}'(x)$ of $\boldsymbol{q}(x)$ and Σ by analysing possible locations of $h(y)$ and $h(z)$ in $\mathfrak{C}_{\Sigma,D}$. To begin with, both of them can belong to Δ_D. To take account of such a homomorphism, we include $\exists y, z\, (\mathit{worksOn}(x, y) \land (\mathit{worksOn}(z, y) \lor \mathit{isManagedBy}(y, z) \lor \mathit{involves}(y, z)) \land \mathit{Professor}(z))$ in $\boldsymbol{q}'(x)$ as a disjunct. Another possible homomorphism, h_1, can have $h_1(y)$ in Δ_D but $h_1(z)$ among the labelled nulls, which can happen if $h_1(y)$ is an instance of *Project* (see Fig. 2 in the middle). To take such a homomorphism into account, we include the disjunct $\exists y\, (\mathit{worksOn}(x, y) \land \mathit{Project}(y))$ in $\boldsymbol{q}'$. Then, there can be a homomorphism, h_2, with both $h_2(y)$ and $h_2(z)$ being labelled nulls, which can happen if $h_2(x)$ is an instance of *RA* (see Fig. 2 on the left). This gives us the third disjunct, $\mathit{RA}(x)$, in $\boldsymbol{q}'(x)$. Finally, there can be a homomorphism, h_3, such that $h_3(y)$ is a labelled

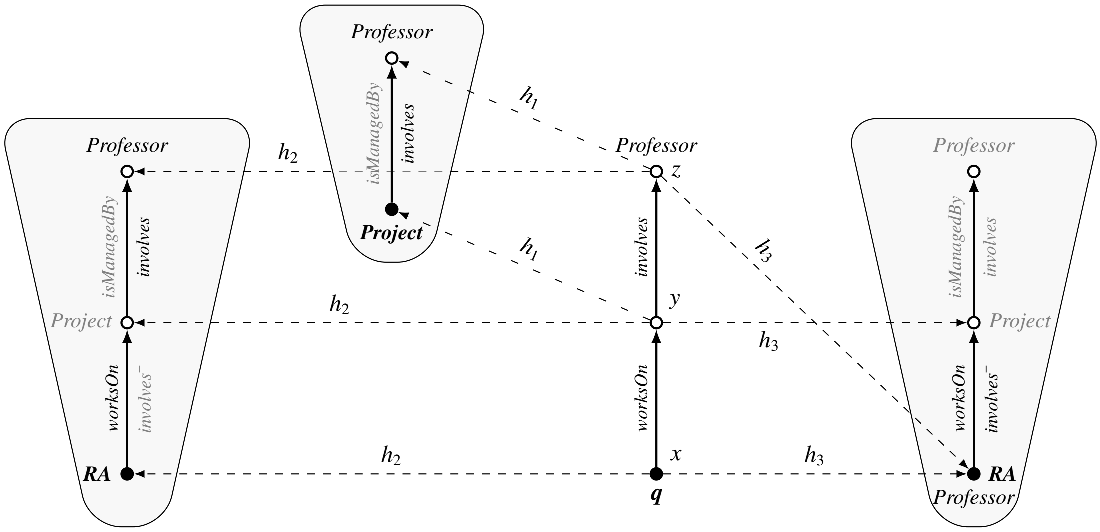

Figure 2: Three homomorphisms from $\boldsymbol{q}(x)$ to a hypothetical $\mathfrak{C}_{\Sigma,D}$.

null but $h_3(z)$ is in Δ_D—this can happen if $h_3(z) = h_3(x)$ is an instance of both *RA* and *Professor* (see Fig. 2 on the right). This homomorphism, however, gives a disjunct $RA(x) \land Professor(z) \land (x = z)$, which is subsumed by the third disjunct, $RA(x)$, and so is redundant. Thus, we obtain the FO-rewriting $\boldsymbol{q}'(x)$ of $\boldsymbol{q}(x)$ and Σ given in the introduction.

Our next example gives an ontology without BDDP.

Example 4. Consider the ontology $\Sigma = \{\forall x, y\,(R(x, y) \land A(y) \to A(x))\}$, whose single tgd is not linear or *OWL 2 QL* (because of the two atoms in the body) and not sticky either (because of the variable y). Given a data instance D, we can again construct a universal model of (Σ, D) using the chase procedure. However, to derive $A(a)$ for some $a \in \Delta_D$, we have to find an R-chain between a and some b with $A(b) \in D$. The number of chase steps producing chains of this kind may clearly depend on D. Ontologies such as Σ are allowed in the *OWL 2 EL* profile of *OWL 2*. CQ answering over *OWL 2 EL*-ontologies is known to be P-complete for data complexity [15], which means that in general they do not enjoy FO-rewritability. (A different approach to OBDA with *OWL 2 EL* was suggested by Lutz *et al.* [40].) On the other hand, CQs over ontologies formulated in *OWL 2 EL* and the description logics Horn-$\mathcal{SHIQ}$ and Horn-$\mathcal{SROIQ}$ can be rewritten into (recursive) datalog queries [53, 44, 20] and used together with datalog engines.

OBDA via FO-rewritability is based on the empirical assumption that query evaluation using RDBMSs is efficient in practice. However, this assumption only works for reasonably small CQs; evaluation of large CQs can be a very hard problem for RDBMSs (see, e.g., [41]), which should not come as a surprise because CQ evaluation is $W[1]$-complete[2] [21]. Recall, however, that CQs of bounded treewidth can be evaluated in polynomial time in $|\boldsymbol{q}|$ and $|\Delta_D|$ [60, 34, 17, 27]. Since such CQs occur most often in practice, this result can serve as a theoretical justification for the empirical assumption above.

But what is the size of the existing FO-rewritings for CQs and ontologies in the languages under consideration? The following theorem summarises some of the known results:

Theorem 5 ([16, 11, 25, 13, 24])**.** *For any set Σ of tgds, let K_Σ be the number of predicates in Σ and let L_Σ be the maximum arity of the predicates in Σ.*

(i) *There exist CQs $\boldsymbol{q}$ and sets Σ of OWL 2 QL-tgds any UCQ-rewritings of which have $\Omega(K_\Sigma^{|\boldsymbol{q}|})$ CQs.*

[2] More precisely, evaluation of a Boolean CQ $\boldsymbol{q}$ over D can be done in time $O(|\boldsymbol{q}| \cdot |\Delta_D|^{|\boldsymbol{q}|})$, but cannot be done in time $f(|\boldsymbol{q}|) \cdot |\Delta_D|^{O(1)}$, for any computable function f, unless FPT = $W[1]$.

(*ii*) *Any CQ* $\boldsymbol{q}$ *and any set* Σ *of linear tgds without constants have a UCQ-rewriting with* $O((K_\Sigma \cdot (L_\Sigma \cdot |\boldsymbol{q}|)^{L_\Sigma})^{|\boldsymbol{q}|})$ *CQs such that the number of atoms in each CQ does not exceed the number of atoms in* $\boldsymbol{q}$.
In particular, for OWL 2 QL-tgds Σ, $L_\Sigma \leq 2$ *and the UCQ-rewriting has* $O((K_\Sigma \cdot (2|\boldsymbol{q}|)^2)^{|\boldsymbol{q}|})$ *CQs.*

(*iii*) *Any CQ* $\boldsymbol{q}$ *and any sticky set* Σ *of tgds without constants have a UCQ-rewriting with* $2^{O(K_\Sigma \cdot (L_\Sigma \cdot |\boldsymbol{q}|)^{L_\Sigma})}$ *CQs, each of which has* $O(K_\Sigma \cdot (L_\Sigma \cdot |\boldsymbol{q}|)^{L_\Sigma})$ *atoms.*

Proof. (*i*) Let $\Sigma = \{\forall x\,(A_i(x) \to A_0(x)) \mid 1 \leq i \leq n\}$ and $\boldsymbol{q} = \exists x_1, \dots, x_k(A_0(x_1) \land \dots \land A_0(x_k))$. It should be clear that any UCQ-rewriting of $\boldsymbol{q}$ and Σ must contain CQs with all possible combinations of $A_0(x_j), A_1(x_j), \dots, A_n(x_j)$, for each $1 \leq j \leq k$.

For (*ii*) and (*iii*), we only briefly comment on the UCQ-rewritings constructed in [16, 11, 25, 13, 24] using backward chaining. (*ii*) Since the tgds have a single atom in the body, the number of atoms in each of the CQs of the resulting UCQ-rewriting cannot be larger than the number of atoms in $\boldsymbol{q}$. Thus, each of these CQs contains at most $L_\Sigma \cdot |\boldsymbol{q}|$ terms, and we can assume that they use the same names for existentially quantified variables. The total number of atoms we can form using these terms does not exceed $K_\Sigma \cdot (L_\Sigma \cdot |\boldsymbol{q}|)^{L_\Sigma}$. Given that each CQ of the UCQ-rewriting has at most $|\boldsymbol{q}|$ atoms, the total number of possible component CQs is bounded by $(K_\Sigma \cdot (L_\Sigma \cdot |\boldsymbol{q}|)^{L_\Sigma})^{|\boldsymbol{q}|}$. (*iii*) Observe that the new variables arising in the UCQ-rewriting are all existentially quantified. Due to the stickiness condition, any such new variable must occur at most once in the body of the tgd used for the rewriting. This variable cannot interact with any other variable, and we can use a unique special symbol for it, which corresponds to the 'don't care' underscore symbol in Prolog. Then each term in each atom of the rewritten query is either a variable from $\boldsymbol{q}$ or the special underscore symbol (in the end, each underscore symbol is replaced by a fresh existentially quantified variable). There are at most $L_\Sigma \cdot |\boldsymbol{q}| + 1$ such terms. It follows that there are at most $K_\Sigma \cdot (L_\Sigma \cdot |\boldsymbol{q}| + 1)^{L_\Sigma} = O(K_\Sigma \cdot (L_\Sigma \cdot |\boldsymbol{q}|)^{L_\Sigma})$ atoms in any CQ of the UCQ-rewriting. Each of the atoms is either included in a CQ or not included in it, which gives $2^{O(K_\Sigma \cdot (L_\Sigma \cdot |\boldsymbol{q}|)^{L_\Sigma})}$ possible CQs in the UCQ-rewriting. ❑

Thus, even for the weakest ontology language *OWL 2 QL*, the available (UCQ) rewritings are of exponential size in the worst case. The chief problem we analyse in this article is whether there exist shorter rewritings. Together with FO- and UCQ-rewritings defined above, we also consider positive existential and nonrecursive datalog rewritings.

A *positive existential rewriting* (*PE-rewriting*, for short) of a CQ $\boldsymbol{q}(\boldsymbol{x})$ and an ontology Σ is an FO-rewriting $\boldsymbol{q}'(\boldsymbol{x})$ of the form $\exists \boldsymbol{z}\,\psi(\boldsymbol{x}, \boldsymbol{z})$, where ψ is built from atoms using only $\land$ and $\lor$. (Every PE-rewriting can obviously be transformed to an equivalent UCQ-rewriting but at the expense of an exponential blowup.) To define nonrecursive datalog rewritings, we remind the reader [1] that a *datalog program*, Π, is a finite set of Horn clauses

$$A_0 \leftarrow A_1 \land \dots \land A_m,$$

where each A_i is an atom of the form $P(\boldsymbol{t})$ and each term t in $\boldsymbol{t}$ is either a (universally quantified) variable or an individual constant. A_0 is called the *head* of the clause, and $A_1, \dots, A_m$ its *body*. All variables occurring in the head A_0 must also occur in the body in one of $A_1, \dots, A_m$. A predicate P *depends* on a predicate Q if Π contains a clause whose head's predicate is P and whose body contains an atom with predicate Q. A datalog program Π is called *nonrecursive* if this dependence relation is acyclic. A *nonrecursive datalog query* consists of a nonrecursive datalog program Π and a *goal* $G(\boldsymbol{x})$, which is just an atom. Given a data instance D, a tuple $\boldsymbol{a}$ of elements in Δ_D is called a *certain answer* to $(\Pi, G(\boldsymbol{x}))$ over D if $\Pi \cup D \models G(\boldsymbol{a})$. A nonrecursive datalog query $(\Pi, G(\boldsymbol{x}))$ is called a *nonrecursive datalog rewriting* of a CQ $\boldsymbol{q}(\boldsymbol{x})$ and an ontology Σ (*NDL-rewriting*, for short) if, for any data instance D and any tuple $\boldsymbol{a}$ of elements in Δ_D, we have $(\Sigma, D) \models \boldsymbol{q}(\boldsymbol{a})$ if and only if $\Pi \cup D \models G(\boldsymbol{a})$.

So far we have not specified what means one is allowed to use in rewritings. The first FO-rewritings of [16, 45] were formulated in the signature that contained only constant and predicate symbols from $\boldsymbol{q}$ and Σ as well as equality. As argued by Calvanese *et al.* [16], FO-rewritings should be data-independent (and so applicable to all possible data instances). We start by adopting this definition for FO- and PE-rewritings; in NDL-rewritings, we can, of course, use new definable (or intensional) predicates, but no constants that do not occur in $\boldsymbol{q}$.

We are interested in three major questions: (*i*) Do there exist polynomial-size FO-, PE-, NDL-rewritings of CQs and *OWL 2 QL*-ontologies? (*ii*) Can rewritings of one type be substantially shorter than rewritings of other types? (*iii*) What extra means in rewritings can make them substantially shorter?

3. Exponential and Superpolynomial Lower Bounds for the Size of Rewritings

In this section, we give an answer to question (*i*). To this end, we show how the problem of constructing circuits that compute monotone Boolean functions in NP can be reduced to the problem of finding rewritings for CQs and *OWL 2 QL*-ontologies. This reduction coupled with the known lower bounds on the size of monotone Boolean circuits and formulas will provide us with similar lower bounds on the size of rewritings.

We begin by reminding the reader of some basic definitions from the theory of circuit complexity (for more details see, e.g., [3, 29]). 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 $f(\boldsymbol{\alpha}) \leq f(\boldsymbol{\beta})$ for all $\boldsymbol{\alpha} \leq \boldsymbol{\beta}$, where $\leq$ is the component-wise $\leq$ on vectors of $\{0,1\}$. 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 *size* of $\mathbf{C}$, denoted $|\mathbf{C}|$, is the number of nodes in $\mathbf{C}$ (that is, the number of inputs and gates).

A *family of Boolean functions* is a sequence $f^1, f^2, \dots$, where each f^n is an n-ary Boolean function. A family $f^1, f^2, \dots$ is in the complexity class NP if the language $\{\, \boldsymbol{\alpha} \in \{0,1\}^n \mid f^n(\boldsymbol{\alpha}) = 1 \,\}$ is in NP. For each such family, there exist polynomials p, q and Boolean circuits $\mathbf{C}^1, \mathbf{C}^2, \dots$ such that $\mathbf{C}^n$ has $n + p(n)$ inputs, $|\mathbf{C}^n| \leq q(n)$ and, for any $\boldsymbol{\alpha} \in \{0,1\}^n$, we have

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

We call the additional $p(n)$ inputs for $\boldsymbol{\beta}$ in $\mathbf{C}^n$ *nondeterministic inputs* ($\boldsymbol{\beta}$ is also known as a *certificate* [3]). A family $f^1, f^2, \dots$ is NP-*complete* if the corresponding language $\{\, \boldsymbol{\alpha} \in \{0,1\}^n \mid f^n(\boldsymbol{\alpha}) = 1 \,\}$ is NP-complete.

The class of languages that are decidable by families of *polynomial-size* circuits is denoted by P/poly. It is known that P $\subsetneqq$ P/poly. Thus, we would obtain P $\neq$ NP if we could show that NP $\nsubseteq$ P/poly. By the Karp-Lipton theorem (see, e.g., [3]), NP $\subseteq$ P/poly implies PH $= \Sigma_2^p$.

In this section, given a family of monotone Boolean functions f^n in NP, we first encode them—via the Tseitin transformation [59]—by means of polynomial-size CNFs, which are used to construct a sequence of *OWL 2 QL*-ontologies Σ_{f^n} and Boolean CQs $\boldsymbol{q}_{f^n}$ such that

$$(\Sigma_{f^n}, D_{\boldsymbol{\alpha}}) \models \boldsymbol{q}_{f^n} \qquad \text{if and only if} \qquad f^n(\boldsymbol{\alpha}) = 1, \qquad \text{for any } \boldsymbol{\alpha} \in \{0,1\}^n,$$

where the database instance $D_{\boldsymbol{\alpha}}$ is determined by $\boldsymbol{\alpha}$. Then, using the fact that the $D_{\boldsymbol{\alpha}}$ have a single domain element, we show that if we have, say, PE-rewritings of the $\boldsymbol{q}_{f^n}$ and Σ_{f^n}, then those rewritings are in essence monotone Boolean formulas (that is, propositional PE-formulas), and so, by the known results on circuit complexity, cannot be polynomial, for example, in the case of the family of Boolean functions that check whether a given graph (encoded by arguments of the functions) contains a *clique* of the specified size.

Suppose we are given a family of Boolean functions f^n in NP and a corresponding family of Boolean circuits $\mathbf{C}^n$. We can consider the inputs (including nondeterministic ones) of the circuits $\mathbf{C}^n$ as Boolean variables. Each gate of $\mathbf{C}^n$ can also be thought of as a Boolean variable whose value coincides with the output of the gate on a given input. Let $\boldsymbol{g} = (g_1, \dots, g_{|\mathbf{C}^n|})$ be the Boolean variables for the nodes of $\mathbf{C}^n$. We may assume that a Boolean circuit $\mathbf{C}^n$ contains only $\land$- and $\neg$-gates, so it can be regarded as a set of equations of the form

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

where $g_{i'}$ and $g_{i''}$ are the variables for the inputs of the gate g_i. We assume that g_i can depend only on $g_1, \dots, g_{i-1}$ and that $g_1, \dots, g_n$ are the inputs of $\mathbf{C}^n$, $g_{n+1}, \dots, g_{n+p(n)}$ are the nondeterministic inputs of $\mathbf{C}^n$, and $g_{|\mathbf{C}^n|}$ its output. Now, with each $\mathbf{C}^n$ we associate the following Boolean formula in CNF with the variables $\boldsymbol{h} = (h_1, \dots, h_n)$ and $\boldsymbol{g}$:

$$\begin{aligned}\psi_n(\boldsymbol{h}, \boldsymbol{g}) \;=\; & \bigwedge_{i=1}^{n} (\neg g_i \lor h_i) \;\land\; g_{|\mathbf{C}^n|} \;\land \\ & \bigwedge_{g_i = \neg g_{i'} \text{ in } \mathbf{C}^n} [(g_{i'} \lor g_i) \land (\neg g_{i'} \lor \neg g_i)] \;\land \bigwedge_{g_i = g_{i'} \land g_{i''} \text{ in } \mathbf{C}^n} [(g_{i'} \lor \neg g_i) \land (g_{i''} \lor \neg g_i) \land (\neg g_{i'} \lor \neg g_{i''} \lor g_i)].\end{aligned}$$

The clauses of the last two conjuncts encode the correct computation of the circuit: they are equivalent to $g_i \leftrightarrow \neg g_{i'}$ and $g_i \leftrightarrow g_{i'} \wedge g_{i''}$, respectively. In what follows, we denote by $\psi_n(\boldsymbol{\alpha}, \boldsymbol{g})$ the result of replacing the variables in $\boldsymbol{h}$ with the respective truth-values from a vector $\boldsymbol{\alpha} \in \{0,1\}^n$ (thus, the $\boldsymbol{g}$ are the only variables of this formula).

Lemma 6. *For any family of monotone Boolean functions f^n in* NP *and any $\boldsymbol{\alpha} \in \{0,1\}^n$, we have $f^n(\boldsymbol{\alpha}) = 1$ if and only if $\psi_n(\boldsymbol{\alpha}, \boldsymbol{g})$ is satisfiable.*

Proof. ($\Rightarrow$) If $f^n(\boldsymbol{\alpha}) = 1$ then $\mathbf{C}^n(\boldsymbol{\alpha},\boldsymbol{\beta}) = 1$, for some $\boldsymbol{\beta}$. Consider $\psi_n(\boldsymbol{\alpha}, \boldsymbol{\gamma})$, where the γ_i in $\boldsymbol{\gamma}$ are given by the output values of the respective nodes g_i in $\mathbf{C}^n$ on the input $(\boldsymbol{\alpha},\boldsymbol{\beta})$ (the output value of an input or a nondeterministic input of $\mathbf{C}^n$ is the respective value itself). By definition, the last two conjuncts of $\psi_n(\boldsymbol{\alpha}, \boldsymbol{\gamma})$ are true under such an assignment. The first conjunct is trivially true, while the second conjunct is true because $\gamma_{|\mathbf{C}^n|} = \mathbf{C}^n(\boldsymbol{\alpha},\boldsymbol{\beta})$.

($\Leftarrow$) Conversely, suppose $\psi_n(\boldsymbol{\alpha}, \boldsymbol{\gamma}) = 1$, for some $\boldsymbol{\gamma}$. Let $\boldsymbol{\alpha}'$ be the values of the inputs of $\mathbf{C}^n$ in $\boldsymbol{\gamma}$. By the first conjunct, $\boldsymbol{\alpha}' \leq \boldsymbol{\alpha}$ and, as f^n is monotone, we obtain $f^n(\boldsymbol{\alpha}') \leq f^n(\boldsymbol{\alpha})$. So, it suffices to show that $f^n(\boldsymbol{\alpha}') = 1$. To this end, we prove by induction on the structure of $\mathbf{C}^n$ that the values of the variables of $\psi_n(\boldsymbol{\alpha}, \boldsymbol{\gamma})$ are equal to the output values of the corresponding nodes of $\mathbf{C}^n$ on $(\boldsymbol{\alpha}',\boldsymbol{\beta})$, where $\boldsymbol{\beta}$ are the values of the nondeterministic inputs from $\boldsymbol{\gamma}$: for the inputs (including nondeterministic ones), this is immediate by definition; for the gates, the claim easily follows from the last two conjuncts of ψ_n. Then, by the second conjunct, $\gamma_{|\mathbf{C}^n|} = 1$, and so $\mathbf{C}^n(\boldsymbol{\alpha}',\boldsymbol{\beta}) = 1$, whence $f^n(\boldsymbol{\alpha}') = 1$. ❑

The second step of the reduction is to encode satisfiability of $\psi_n(\boldsymbol{\alpha}, \boldsymbol{g})$ by means of the CQ answering problem in *OWL 2 QL*. The CNF $\psi_n(\boldsymbol{h}, \boldsymbol{g})$ contains $d \leq 3|\mathbf{C}^n| + 1$ clauses $C_1, \dots, C_d$ with n variables $h_1, \dots, h_n$ and $m = |\mathbf{C}^n|$ variables $g_1, \dots, g_m$. Recall that $g_1, \dots, g_n$ correspond to the inputs and $C_1, \dots, C_n$ are clauses of the form $\neg g_i \vee h_i$. We take a binary predicate $P(x,y)$ and unary predicates $A_0(x)$ and $A_i(x)$, $X_i^0(x)$, $X_i^1(x)$, for each variable g_i, as well as $Z_{0,j}(x), \dots, Z_{m,j}(x)$, for each clause C_j of $\psi_n(\boldsymbol{h}, \boldsymbol{g})$.

Consider an *OWL 2 QL*-ontology Σ_{f^n} with the following tgds, for $1 \leq i \leq m$, $1 \leq j \leq d$ and $\ell = 0, 1$:

$$\forall x\, (A_{i-1}(x) \to \exists y\, (P(y,x) \wedge X_i^\ell(y))), \qquad \forall x\, (X_i^\ell(x) \to A_i(x)),$$

$$\forall x\, (Z_{i,j}(x) \to \exists y\, (P(x,y) \wedge Z_{i-1,j}(y))), \qquad \begin{array}{ll} \forall x\, (X_i^0(x) \to Z_{i,j}(x)), & \text{if } \neg g_i \in C_j, \\ \forall x\, (X_i^1(x) \to Z_{i,j}(x)), & \text{if } g_i \in C_j. \end{array}$$

It is not hard to check that $|\Sigma_{f^n}| = O(|\mathbf{C}^n|^2)$ and that the chase of Σ_{f^n} is finite for any data. Consider also the following tree-shaped Boolean CQ:

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

where $\boldsymbol{y} = (y_0, \dots, y_m)$ and $\boldsymbol{z} = (z_{0,1}, \dots, z_{m-1,1}, \dots, z_{0,d}, \dots, z_{m-1,d})$. It should be clear that $|\boldsymbol{q}_{f^n}| = O(|\mathbf{C}^n|^2)$.

For each $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_n) \in \{0,1\}^n$, we take the data instance

$$D_{\boldsymbol{\alpha}} = \{\, A_0(a) \,\} \cup \{\, Z_{0,i}(a) \mid 1 \leq i \leq n \text{ and } \alpha_i = 1 \,\}.$$

We explain the intuition behind Σ_{f^n}, $\boldsymbol{q}_{f^n}$ and $D_{\boldsymbol{\alpha}}$ using the example in Fig. 3, where the chase $\mathfrak{C}_{\Sigma_{f^n}, D_{\boldsymbol{\alpha}}}$ of $(\Sigma_{f^n}, D_{\boldsymbol{\alpha}})$ is depicted for a particular f^n and $\boldsymbol{\alpha}$. To answer $\boldsymbol{q}_{f^n}$ over $(\Sigma_{f^n}, D_{\boldsymbol{\alpha}})$, we have to check whether $\boldsymbol{q}_{f^n}$ can be homomorphically mapped into $\mathfrak{C}_{\Sigma_{f^n}, D_{\boldsymbol{\alpha}}}$. The variables y_i are clearly mapped to one of the main branches of the model, from a to a point in A_3, say the leftmost one, which corresponds to the valuation for the variables $\boldsymbol{g}$ in $\psi_n(\boldsymbol{\alpha}, \boldsymbol{g})$ making all of them false. Consider now, for example, variables $z_{2,3}, z_{1,3}, z_{0,3}$ that correspond to the clause $C_3 = g_1 \vee \neg g_3$ in $\psi_n(\boldsymbol{\alpha}, \boldsymbol{g})$. Since $Z_{0,3}(a) \notin D_{\boldsymbol{\alpha}}$, in order to map $z_{2,3}, z_{1,3}, z_{0,3}$ we have to choose at least one of its literals, g_1 or $\neg g_3$, that is true under such an assignment, and then $z_{2,3}, z_{1,3}, z_{0,3}$ can be sent to the points in the respective 'hanging' branch, resulting in $z_{0,3} \not\mapsto a$. On the other hand, there are two possible ways (depending on α_1) of mapping variables $z_{2,1}, z_{1,1}, z_{0,1}$ for the clause $C_1 = \neg g_1 \vee h$ of $\psi_n(\boldsymbol{\alpha}, \boldsymbol{g})$. (1) If $\alpha_1 = 0$ then C_1 in $\psi_n(\boldsymbol{\alpha}, \boldsymbol{g})$ is equivalent to $\neg g_1$ and, since $Z_{0,1}(a) \notin D_{\boldsymbol{\alpha}}$, we have to be able to send $z_{2,1}, z_{1,1}, z_{0,1}$ to the points in a 'hanging' branch, resulting in $z_{0,1} \not\mapsto a$. (2) If, however, $\alpha_1 = 1$ then the clause C_1 is true anyway and $Z_{0,1}(a) \in D_{\boldsymbol{\alpha}}$, whence $z_{2,1}, z_{1,1}, z_{0,1}$ can be sent to the same branch from A_2 to A_0, so that $z_{0,1} \mapsto a$. Thus, we arrive to the following:

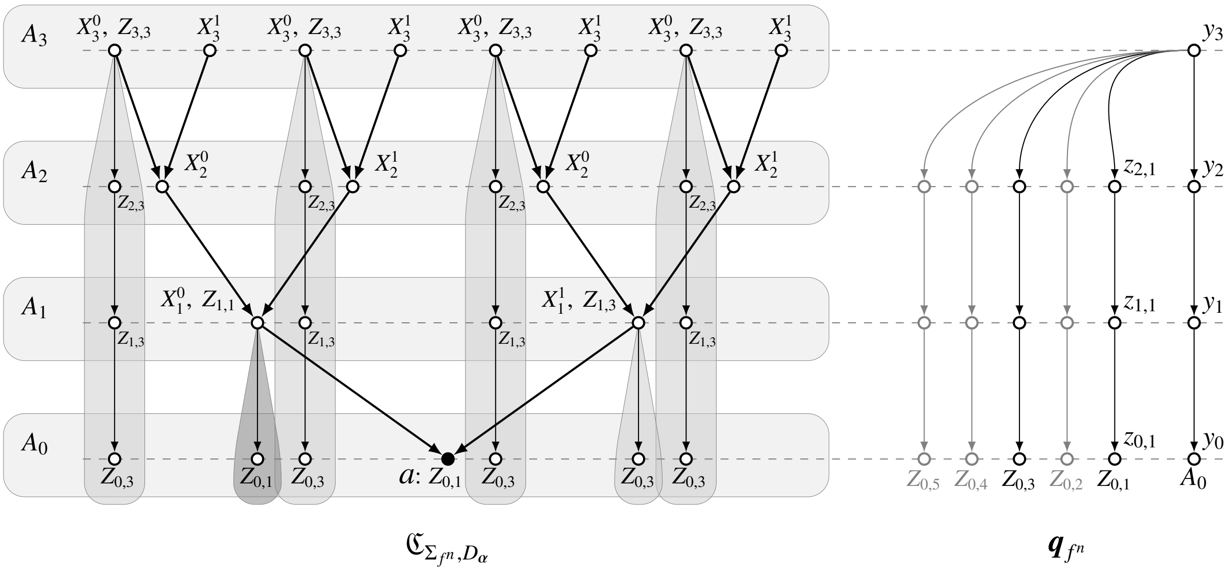

Figure 3: The chase $\mathfrak{C}_{\Sigma_{f^n},D_\alpha}$ and CQ $\boldsymbol{q}_{f^n}$ for $\boldsymbol{\alpha} = (1)$ and a function f^n with one input and one nondeterministic input to one $\wedge$-gate. Thus, $n = 1$, $m = 3$, $d = 5$ and $\psi_n(h, g_1, g_2, g_3) = (\neg g_1 \vee h) \wedge g_3 \wedge (g_1 \vee \neg g_3) \wedge (g_2 \vee \neg g_3) \wedge (\neg g_1 \vee \neg g_2 \vee g_3)$. Only two groups of the 'hanging' $Z_{i,j}$ branches are shown in $\mathfrak{C}_{\Sigma_{f^n},D_\alpha}$: for $j = 1$ and $j = 3$, that is, for $C_1 = \neg g_1 \vee h$ and $C_3 = g_1 \vee \neg g_3$.

Lemma 7. *For any family of Boolean functions f^n in* NP *and any $\boldsymbol{\alpha} \in \{0,1\}^n$, we have $(\Sigma_{f^n}, D_{\boldsymbol{\alpha}}) \models \boldsymbol{q}_{f^n}$ if and only if $\psi_n(\boldsymbol{\alpha}, \boldsymbol{g})$ is satisfiable.*

Proof. ($\Rightarrow$) Consider a homomorphism h from $\boldsymbol{q}_{f^n}$ to the chase $\mathfrak{C}_{\Sigma_{f^n},D_\alpha}$ of $(\Sigma_{f^n}, D_{\boldsymbol{\alpha}})$. Clearly, $h(y_0) = a$ and both $A_i(h(y_i))$ and $P(h(y_i), h(y_{i-1}))$ are in $\mathfrak{C}_{\Sigma_{f^n},D_\alpha}$, for all $1 \le i \le m$. So, for each variable g_i in $\boldsymbol{g}$, we set $\gamma_i = 1$ if $X_i^1(h(y_i)) \in \mathfrak{C}_{\Sigma_{f^n},D_\alpha}$ and $\gamma_i = 0$ otherwise (in which case $X_i^0(h(y_i)) \in \mathfrak{C}_{\Sigma_{f^n},D_\alpha}$). We claim that $\psi_n(\boldsymbol{\alpha}, \boldsymbol{\gamma}) = 1$. Take any clause C_j in $\psi_n(\boldsymbol{\alpha}, \boldsymbol{g})$ and consider two cases for $h(z_{0,j})$. If $h(z_{0,j}) = a$ then $1 \le j \le n$ with $Z_{0,j}(a) \in D_{\boldsymbol{\alpha}}$, and so $\alpha_j = 1$, whence the clause $C_j = \neg g_j \vee h_j$ is true anyway. Otherwise, $h(z_{0,j}) \neq a$ which means that $Z_{i,j}(h(y_i)) \in \mathfrak{C}_{\Sigma_{f^n},D_\alpha}$, for some $1 \le i \le m$, and so the clause C_j contains g_i if $X_i^1(h(y_i)) \in \mathfrak{C}_{\Sigma_{f^n},D_\alpha}$ and $\neg g_i$ if $X_i^0(h(y_i)) \in \mathfrak{C}_{\Sigma_{f^n},D_\alpha}$. The claim follows.

($\Leftarrow$) Suppose $\psi_n(\boldsymbol{\alpha}, \boldsymbol{\gamma}) = 1$, for some $\boldsymbol{\gamma} \in \{0,1\}^m$. We construct a homomorphism h from $\boldsymbol{q}_{f^n}$ to the chase $\mathfrak{C}_{\Sigma_{f^n},D_\alpha}$ of $(\Sigma_{f^n}, D_{\boldsymbol{\alpha}})$. Observe that $\mathfrak{C}_{\Sigma_{f^n},D_\alpha}$ contains a path $u_0, \dots, u_m$ from $a = u_0$ to some u_m such that $P(u_i, u_{i-1}) \in \mathfrak{C}_{\Sigma_{f^n},D_\alpha}$, for $1 \le i \le m$, and the path corresponds to $\boldsymbol{\gamma}$ in the following sense: $X_i^1(u_i) \in \mathfrak{C}_{\Sigma_{f^n},D_\alpha}$ if $\gamma_i = 1$ and $X_i^0(u_i) \in \mathfrak{C}_{\Sigma_{f^n},D_\alpha}$ otherwise. So, for $0 \le i \le m$, we set $h(y_i) = u_i$. For $1 \le j \le d$, we define $h(z_{m-1,j}), \dots, h(z_{0,j})$ recursively, starting from $h(z_{m-1,j})$ and assuming that $z_{m,j} = y_m$: let $h(z_{i,j}) = u_i$ if $Z_{i+1,j}(h(z_{i+1,j})) \notin \mathfrak{C}_{\Sigma_{f^n},D_\alpha}$; otherwise, let $h(z_{i,j})$ be the labelled null chosen for y when applying $\forall x\, (Z_{i+1,j}(x) \to \exists y\, (P(x,y) \wedge Z_{i,j}(x)))$ in $h(z_{i+1,j})$. It is easy to check that h is indeed a homomorphism from $\boldsymbol{q}_{f^n}$ into $\mathfrak{C}_{\Sigma_{f^n},D_\alpha}$. ❑

We now use the reduction above to show that there is a close correspondence between PE-rewritings and monotone Boolean formulas, between FO-rewritings and (not necessarily monotone) Boolean formulas, and between NDL-rewritings and monotone Boolean circuits.

Lemma 8. *Suppose $f^1, f^2, \dots$ is a family of monotone Boolean functions in* NP.

(*i*) *If $\boldsymbol{q}'_{f^n}$ is an FO-rewriting of $\boldsymbol{q}_{f^n}$ and Σ_{f^n}, then there is a Boolean formula φ_n computing f^n with $|\varphi_n| \le |\boldsymbol{q}'_{f^n}|$.*

(*ii*) *If $\boldsymbol{q}'_{f^n}$ is a PE-rewriting of $\boldsymbol{q}_{f^n}$ and Σ_{f^n}, then there is a monotone Boolean formula φ_n computing f^n with $|\varphi_n| \le |\boldsymbol{q}'_{f^n}|$.*

(*iii*) *If (Π_{f^n}, G) is an NDL-rewriting of $\boldsymbol{q}_{f^n}$ and Σ_{f^n}, then there is a monotone Boolean circuit $\mathbf{B}^n$ computing f^n with $|\mathbf{B}^n| \le |\Pi_{f^n}|$.*

Proof. (*i*) By Lemmas 6 and 7, for any FO-rewriting $\boldsymbol{q}'_{f^n}$ of $\boldsymbol{q}_{f^n}$ and Σ_{f^n},

$$D_\alpha \models \boldsymbol{q}'_{f^n} \quad \text{if and only if} \quad f^n(\boldsymbol{\alpha}) = 1, \qquad \text{for any } \boldsymbol{\alpha} \in \{0,1\}^n.$$

Since Δ_{D_α} is a singleton, $\{a\}$, we can remove all the quantifiers and replace all the individual variables in $\boldsymbol{q}'_{f^n}$ with a. The resulting Boolean FO-query $\boldsymbol{q}''_{f^n}$ has the same truth-value in D_α as $\boldsymbol{q}'_{f^n}$. Then we observe that the ground atoms other than $a = a$, $A_0(a)$ and the $Z_{0,j}(a)$, for $1 \le j \le n$, are false in D_α, and so we can replace all $a = a$ and $A_0(a)$ with $\top$, and all the atoms different from $a = a$, $A_0(a)$ and $Z_{0,j}(a)$, for $1 \le j \le n$, with $\bot$ without affecting the truth-value of $\boldsymbol{q}''_{f^n}$ in D_α. The resulting quantifier-free query can be regarded as a Boolean formula, φ_n, with 'propositional variables' $Z_{0,1}(a), \dots, Z_{0,n}(a)$. But then $\varphi_n(\boldsymbol{\alpha}) = f^n(\boldsymbol{\alpha})$, for each $\boldsymbol{\alpha} \in \{0,1\}^n$; that is, φ_n computes f^n. Clearly, $|\varphi_n| \le |\boldsymbol{q}'_{f^n}|$.

(*ii*) In the same way as above we can transform any PE-rewriting $\boldsymbol{q}'_{f^n}$ of $\boldsymbol{q}_{f^n}$ and Σ_{f^n} into a monotone Boolean formula φ_n (with connectives $\lor$ and $\land$ only) and propositional variables $Z_{0,1}(a), \dots, Z_{0,n}(a)$ such that φ_n computes f^n and $|\varphi_n| \le |\boldsymbol{q}'_{f^n}|$.

(*iii*) Suppose that (Π_{f^n}, G) is an NDL-rewriting of $\boldsymbol{q}_{f^n}$ and Σ_{f^n}, and $\boldsymbol{\alpha} \in \{0,1\}^n$. Again, since Δ_{D_α} is a singleton, each variable in the head of a clause also occurs in its body and Π_{f^n} does not contain constants (as $\boldsymbol{q}_{f^n}$ does not have them), we can replace all the individual variables in Π_{f^n} with a and the resulting NDL-query (Π'_{f^n}, G) has the same truth-value in D_α as (Π_{f^n}, G). Then, in Π'_{f^n}, we remove all $a = a$ and $A_0(a)$ (as they are true) and remove all clauses containing atoms different from $a = a$, $A_0(a)$ and $Z_{0,j}(a)$, for $1 \le j \le n$ (because such atoms are false in D_α and do not occur in the heads of the clauses). Denote the resulting propositional NDL-program by Π''_{f^n}. It follows that $\Pi''_{f^n}, D_\alpha \models G$ if and only if $f^n(\boldsymbol{\alpha}) = 1$. We can regard (Π''_{f^n}, G) as an NDL-query in which $Z_{0,1}(a), \dots, Z_{0,n}(a)$ are 'propositional variables' and the heads of all clauses also have no arguments (i.e., are propositional variables). Such a program Π''_{f^n} can now be transformed into a monotone Boolean circuit computing f^n: for every propositional variable p occurring in the head of a clause in Π''_{f^n}, we introduce a $\lor$-gate whose output is p and inputs are the bodies of the clauses with the head p; and for each such body, we introduce a cascade of $\land$-gates whose inputs are the propositional variables in the body. The resulting monotone Boolean circuit with inputs $Z_{0,1}(a), \dots, Z_{0,n}(a)$ and output G is denoted by $\mathbf{B}^n$. Clearly, $|\mathbf{B}^n| \le |\Pi_{f^n}|$. ❑

We are now in a position to prove that one cannot avoid an exponential blowup for PE- and NDL-rewritings; moreover, even FO-rewritings can blowup superpolynomially under the assumption that NP $\not\subseteq$ P/poly. This can be done using the function $\textsc{Clique}_{m,k}$ of $m(m-1)/2$ variables e_{ij}, $1 \le i < j \le m$, which returns 1 if and only if the graph with vertices $\{1, \dots, m\}$ and edges $\{\{i,j\} \mid e_{ij} = 1\}$ contains a k-clique. One can show that there is a Boolean circuit with m nondeterministic inputs and $O(m^2)$ gates that computes $\textsc{Clique}_{m,k}$. As $\textsc{Clique}_{m,k}$ is NP-complete, the question whether $\textsc{Clique}_{m,k}$ can be computed by polynomial-size circuits (without nondeterministic inputs) is equivalent to the open NP $\subseteq$ P/poly problem. Further, a series of papers, started by Razborov [50], gave an exponential lower bound for the size of *monotone* circuits computing $\textsc{Clique}_{m,k}$: $2^{\Omega(\sqrt{k})}$ for $k \le \frac{1}{4}(m/\log m)^{2/3}$ [2]. For monotone formulas, an even better lower bound is known: $2^{\Omega(k)}$ for $k = 2m/3$ [49].

Theorem 9. *There is a sequence of CQs $\boldsymbol{q}_n$ of size $O(n)$ and OWL 2 QL-ontologies Σ_n of size $O(n)$ such that*
(*i*) *any PE-rewritings of $\boldsymbol{q}_n$ and Σ_n are of size $\ge 2^{\Omega(n^{1/4})}$;*
(*ii*) *any NDL-rewritings of $\boldsymbol{q}_n$ and Σ_n are of size $\ge 2^{\Omega((n/\log n)^{1/12})}$;*
(*iii*) *there are no polynomial-size FO-rewritings of $\boldsymbol{q}_n$ and Σ_n unless* NP $\subseteq$ P/*poly or* PH $= \Sigma_2^p$.

Proof. Consider the family of Boolean functions $f^n = \textsc{Clique}_{m,k}$ with $m = \lfloor n^{1/4} \rfloor$ and $k = \lfloor 2m/3 \rfloor = \Omega(n^{1/4})$. As the size of the circuits $\mathbf{C}^n$ (with nondeterministic inputs) is $O(m^2)$, the size of $\boldsymbol{q}_n = \boldsymbol{q}_{f^n}$ and $\Sigma_n = \Sigma_{f^n}$ is $O(n)$. So, claim (*i*) follows from Lemma 8 (*ii*) and the lower bound for the size of monotone formulas computing $\textsc{Clique}_{m,k}$. Then we take the same family f^n and redefine its elements f^n with *even n*: take $f^n = \textsc{Clique}_{m,k}$ with m as above and $k = \lfloor (m/\log m)^{2/3} \rfloor = \Omega((n/\log n)^{1/6})$. Claim (*ii*) follows from Lemma 8 (*iii*) and the lower bound on the size of monotone circuits computing $\textsc{Clique}_{m,k}$. If we assume that NP $\not\subseteq$ P/poly then there is no polynomial-size circuit for $\textsc{Clique}_{m,k}$, and so (*iii*) follows for the constructed f^n by Lemma 8 (*i*). ❑

Using a similar argument we can also prove the following:

Theorem 10. *Suppose $f^1, f^2, \dots$ is an* NP*-complete family of monotone Boolean functions. If* NP $\not\subseteq$ P/*poly then $\boldsymbol{q}_{f^n}$ and Σ_{f^n} do not have polynomial-size FO- and NDL-rewritings.*

Proof. Suppose to the contrary that there are polynomial-size FO- or NDL-rewritings of $\boldsymbol{q}_{f^n}$ and Σ_{f^n}. Then, by Lemma 8 (*i*) and (*iii*), there is a family of polynomial-size circuits computing $f^1, f^2, \dots$. Since the family f^n is NP-complete, it follows that all families of Boolean functions in NP can be computed by polynomial-size circuits, that is NP $\subseteq$ P/poly. ❑

The construction of this section also reveals the overhead of CQ answering via *OWL 2 QL*-ontologies compared to CQ answering over plain databases in complexity-theoretic terms. Indeed, since the Boolean CQs $\boldsymbol{q}_{f^n}$ are tree-shaped, the problem '$D_\alpha \models \boldsymbol{q}_{f^n}$?' is in P for combined complexity [60], while the problem '$(\Sigma_{f^n}, D_\alpha) \models \boldsymbol{q}_{f^n}$?' is NP-hard. (On the other hand, both problems are in AC^0 for data complexity.)

We also observe that the quantifier elimination in the proof of Lemma 8 relies on the fact that $|\Delta_{D_\alpha}| = 1$. As we shall see in the next two sections, if we restrict attention to data instances with at least two individuals, then Theorem 9 does not hold any longer.

4. Polynomial Rewritings with Two Constants

To prove the exponential and superpolynomial lower bounds for the size of rewritings in the previous section, we established a connection between monotone circuits for Boolean functions and rewritings of certain CQs and *OWL 2 QL*-ontologies. In fact, this connection also suggests a way of making rewritings substantially shorter. Indeed, recall from Section 3 that although no family of monotone Boolean circuits of polynomial size can compute $\textsc{Clique}_{m,k}$, there exists a family of polynomial-size circuits *with nondeterministic inputs* computing $\textsc{Clique}_{m,k}$. Nondeterministic inputs make Boolean circuits exponentially more succinct—in the same way as nondeterministic automata are exponentially more succinct than deterministic ones [42]. To introduce the corresponding nondeterministic guesses into query rewritings, we can use additional existentially quantified variables—provided that the domain of quantification contains at least two elements (cf. [5]). For this purpose, we can extend the signature of PE-, FO- and NDL-rewritings with a set X of constant symbols *assuming that they occur in every relevant data instance*, in which case we are talking about *PE_X-, FO_X- and NDL_X-rewritings*. In this section, we show that allowing additional constants in rewritings really makes them exponentially more succinct.

We say that a family of ontologies has the *polynomial witness property* (*PWP*, for short) if there is a polynomial $d(m, n)$ such that, for any ontology Σ in the family, any CQ $\boldsymbol{q}(\boldsymbol{x})$ and any data instance D, whenever $(\Sigma, D) \models \boldsymbol{q}(\boldsymbol{a})$, for a tuple $\boldsymbol{a}$ from Δ_D, then there is a sequence of $d(|\boldsymbol{q}|, |\Sigma|)$ applications of tgds from Σ to D that entails $\boldsymbol{q}(\boldsymbol{a})$ (in the sense that there is a homomorphism from $\boldsymbol{q}(\boldsymbol{a})$ to the set of atoms generated by those tgd applications). Clearly, PWP implies BDDP (but not the other way round). The following are examples of ontology languages with the PWP:

– linear tgds with predicates of bounded arity [26] and, in particular, *OWL 2 QL* [16],

– sticky sets of tgds with predicates of bounded arity [23]

(note that the degree of the polynomial depends on the maximum arity of predicates).

Theorem 11. *Let $\boldsymbol{q}(\boldsymbol{x})$ be a CQ and Σ an ontology from a family with the PWP.*
(*i*) *There is a $PE_{\{0,1\}}$-rewriting of $\boldsymbol{q}$ and Σ whose size is polynomial in $|\boldsymbol{q}|$ and $|\Sigma|$.*
(*ii*) *There is an $NDL_{\{0,1\}}$-rewriting of $\boldsymbol{q}$ and Σ whose size is polynomial in $|\boldsymbol{q}|$ and $|\Sigma|$.*

Proof. Without loss of generality we assume that all predicates in Σ and $\boldsymbol{q}$ are of some arity L and that all tgds in Σ have precisely m atoms in the body and one atom in the head, and the head contains at most one existentially quantified variable. In other words, all our tgds are of the form

$$\forall \boldsymbol{x}\, (P_1(\boldsymbol{t}_1) \land \dots \land P_m(\boldsymbol{t}_m) \to \exists z\, P_0(\boldsymbol{t}_0)), \tag{7}$$

where each term in the $\boldsymbol{t}_i = (t_{i1}, \dots, t_{iL})$, for $1 \le i \le m$, is a (universally quantified) variable from $\boldsymbol{x}$ or a constant and each term in $\boldsymbol{t}_0 = (t_{01}, \dots, t_{0L})$ either belongs to $\boldsymbol{x}$ (in which case it is universally quantified) or is a constant or coincides with z (in which case it is existentially quantified). To simplify notation, we assume that $\boldsymbol{q}$ is a Boolean CQ:

$$\boldsymbol{q} = \exists \boldsymbol{y} \bigwedge_{k=1}^{M} R_k(y_{k1}, \dots, y_{kL}).$$

We also assume that Σ contains no negative constraints (for a reduction of the general case, see [11]). In view of the PWP, there is a number $d(|\boldsymbol{q}|, |\Sigma|)$ polynomial in $|\boldsymbol{q}|$ and $|\Sigma|$ such that, for any data instance D with $(\Sigma, D) \models \boldsymbol{q}$, there is a sequence of $d(|\boldsymbol{q}|, |\Sigma|)$ applications of tgds from Σ to D that entails $\boldsymbol{q}$. Let $N = (m+1) \cdot d(|\boldsymbol{q}|, |\Sigma|)$. Denote $\mu = \max(K, M, N, S)$, where K is the number of predicates in $\boldsymbol{q}$ and Σ, and S is the number of tgds in Σ. Let Q be the set of natural numbers from 0 to μ.

(i) First, we give a PE_Q-rewriting $\boldsymbol{q}'$ of $\boldsymbol{q}$ and Σ assuming that the constants in Q cannot occur in any predicate of data instances but still are interpreted by distinct elements in every model (equality is a built-in predicate). Then we show how this rewriting can be transformed to a proper $\mathrm{PE}_{\{0,1\}}$-rewriting (without any condition on 0 and 1 apart from that they must occur in all relevant data instances).

In essence, our PE_Q-rewriting guesses a sequence of N ground atoms $A_1, \dots, A_N$ and then checks whether these atoms give a positive answer to $\boldsymbol{q}$ and the sequence can indeed be obtained by a series of applications of the tgds from Σ to D (all the data atoms required for the applications must be among the A_i). To encode the atoms $A_1, \dots, A_N$, we associate with each predicate P a unique number, denoted $[P]$, so that each A_i is represented by the number of its predicate and the values of its arguments, which range over the domain Δ_D of D and the labelled nulls null_i, for $1 \le i \le N$ (the labelled nulls are numbers from Q, but we use this notation for readability). Thus, for each atom A_i in the sequence, $1 \le i \le N$, we need the following variables:

- r_i is the number of the predicate of A_i and $u_{i1}, \dots, u_{iL}$ are the arguments of A_i;
- $w_{i1}, \dots, w_{i\ell}$, where ℓ is the maximum number of universally quantified variables $\boldsymbol{x}$ in tgds ($\ell \le m \cdot L$), are the arguments of the predicates in the body of the tgd used to obtain A_i.

Note that the r_i range over Q and the u_{ij} and the w_{il} range over the domain Δ_D and the labelled nulls (that is, over $\Delta_D \cup Q$). The PE_Q-rewriting of $\boldsymbol{q}$ and Σ is defined by taking:

$$\boldsymbol{q}' \quad = \quad \exists \boldsymbol{y} \exists \boldsymbol{u} \exists \boldsymbol{r} \exists \boldsymbol{w} \Big(\bigwedge_{k=1}^{M} \Gamma_k \wedge \bigwedge_{i=1}^{N} \Phi_i \Big).$$

The first conjunct of $\boldsymbol{q}'$ chooses, for each atom in the query, a match among $A_1, \dots, A_N$:

$$\Gamma_k \quad = \quad \bigvee_{i=1}^{N} \Big[(r_i = [R_k]) \wedge \bigwedge_{j=1}^{L} (u_{ij} = y_{kj}) \Big].$$

The second conjunct guesses, for each ground atom $A_1, \dots, A_N$ whether it is taken from the data instance or obtained by a tgd application:

$$\Phi_i \quad = \bigvee_{P \text{ is a predicate in } \boldsymbol{q} \text{ or } \Sigma} ((r_i = [P]) \wedge P(u_{i1}, \dots, u_{iL})) \quad \vee$$

$$\bigvee_{\tau = \forall \boldsymbol{x}\, (P_1(\boldsymbol{t}_1) \wedge \dots \wedge P_m(\boldsymbol{t}_m) \to \exists z\, P_0(\boldsymbol{t}_0)) \in \Sigma} \Big[(r_i = [P_0]) \wedge \bigwedge_{t_{0j} \text{ is } a} (u_{ij} = a) \wedge \bigwedge_{t_{0j} \text{ is } x_l} (u_{ij} = w_{il}) \wedge \bigwedge_{t_{0j} \text{ is } z} (u_{ij} = \mathit{null}_i) \wedge \bigwedge_{k=1}^{m} \Psi_{\tau,i,k} \Big].$$

The first group of disjuncts is for the case when A_i is taken from the data instance (r_i is such that $P(u_{i1}, \dots, u_{iL})$ appears in the data instance for a predicate P with the number r_i). The second group of disjuncts models the chase rule application, for each tgd τ in Σ. Informally, if A_i is obtained by an application of τ, then r_i is the number $[P_0]$ of the head predicate P_0 and the existential variable z of the head gets a unique labelled null value null_i (the fourth conjunct). Then, by the last conjunct, for each of the m atoms of the body, one can choose a number i' that is *less than* i such that the predicate of $A_{i'}$ is the same as the predicate of the body atom and their arguments match:

$$\Psi_{\tau,i,k} \quad = \quad \bigvee_{i'=1}^{i-1} ((r_{i'} = [P_k]) \wedge \bigwedge_{t_{kj} = x_l} (u_{i'j} = w_{il}) \wedge \bigwedge_{t_{kj} = a} (u_{i'j} = a)),$$

where the variables w_{il} ensure that the same universally quantified variable of τ gets the same value in the body atoms and in the head (if it occurs there, see the second conjunct in the last group of Φ_i). We assume that the empty disjunction is $\bot$, and so $\Psi_{\tau,1,k} = \bot$, for all τ and k.

It is not hard to check that $\boldsymbol{q}'$ can be constructed in polynomial time, $|\boldsymbol{q}'| = O(|\boldsymbol{q}| \cdot |\Sigma| \cdot N^2 \cdot L)$ and that $(\Sigma, D) \models \boldsymbol{q}$ if and only if $\boldsymbol{q}'$ is true in the model of D extended with the constants in Q, which are distinct and do not belong to the interpretation of any predicate but $=$.

We can replace the natural numbers in Q with two distinct constants, say, 0 and 1 (provided that they are present in every data instance), thus obtaining a polynomial $\mathrm{PE}_{\{0,1\}}$-rewriting of $\boldsymbol{q}$ and Σ. Recall that each of the variables u_{ij} ranges over the domain Δ_D and numbers from Q (more precisely, labelled nulls $null_1, \dots, null_N$). Thus, such a variable u_{ij} can be modelled by means a tuple $(\hat{u}_{ij}, u^p_{ij}, \dots, u^0_{ij})$ of variables, where $\hat{u}_{ij}$ ranges over the domain Δ_D, while $u^p_{ij}, \dots, u^0_{ij}$, for $p = \lceil \log |Q| \rceil$, range over $\{0, 1\}$ and represent a natural number from 0 to μ in binary. More precisely, if u_{ij} has a value $d \in \Delta_D$ then $\hat{u}_{ij}$ is interpreted by d and $u^p_{ij}, \dots, u^0_{ij}$ are all zeros; otherwise, u_{ij} is a labelled null, say $null_k$, and so $\hat{u}_{ij}$ is a fixed value, say 0, and $u^p_{ij}, \dots, u^0_{ij}$ represent k in binary (note that 0 is not a labelled null). Similarly, we model the w_{il}; the r_i are even simpler to model as they do not have the $\hat{r}_i$ component. The equality atoms in the rewriting $\boldsymbol{q}'$ are replaced by the component-wise equalities and each $P(u_{i1}, \dots, u_{iL})$ is replaced by $P(\hat{u}_{i1}, \dots, \hat{u}_{iL}) \land \bigwedge_{j=1}^{L} \bigwedge_{k=0}^{p} (u^k_{ij} = 0)$.

(*ii*) We show how to construct a polynomial-size NDL_Q-rewriting (Π, G) of $\boldsymbol{q}$ and Σ. Its transformation into an $\mathrm{NDL}_{\{0,1\}}$-rewriting can be done similarly to PE_Q-rewritings. The program Π has one main rule that is very similar to the query $\boldsymbol{q}'$ in the previous construction. However, $\boldsymbol{q}'$ uses disjunction which is not allowed in a datalog rule. The elimination of disjunction (without an exponential blowup and with small arity of predicates) is based on the equivalence

$$\bigvee_{i \in \Upsilon} \rho_i \quad \equiv \quad \bigvee_{i \in \Upsilon} (v = i) \;\; \land \;\; \bigwedge_{i \in \Upsilon} ((v = i) \to \rho_i), \tag{8}$$

where $\Upsilon \subseteq Q$. To this end, Π uses additional rules and intensional predicates.

– $\mathit{OneOf}(x, y, z)$ should hold if x is a natural number from Q in the interval from y to z (this predicate will replace the disjunction of the $(v = i)$ in (8)):

$$\mathit{OneOf}(i, j, k), \qquad \text{for all } 0 \le j \le i \le k \le \mu;$$

– $\mathit{Dom}(z)$ should hold if z appears in the data instance D or is one of the labelled nulls $null_k$:

$$\begin{aligned} \mathit{Dom}(y_j) &\leftarrow P(y_1, \dots, y_L), && \text{for all predicates } P \text{ in } \boldsymbol{q} \text{ and } \Sigma \text{ and all } 1 \le j \le L,\\ \mathit{Dom}(null_k),& && \text{for all } 1 \le k \le N; \end{aligned}$$

– $\mathit{If}(x_1, x_2, z_1, z_2)$ should hold if $x_1 = x_2 \to z_1 = z_2$ is true, where x_1, x_2 are natural numbers from Q (this predicate will replace the implication in (8)):

$$\begin{aligned} \mathit{If}(i, i, z, z) &\leftarrow \mathit{Dom}(z), && \text{for every } 0 \le i \le \mu,\\ \mathit{If}(i, j, z_1, z_2) &\leftarrow \mathit{Dom}(z_1), \mathit{Dom}(z_2), && \text{for every } 0 \le i \ne j \le \mu; \end{aligned}$$

– $\mathit{IfAnd}(x_1, x_2, y_1, y_2, z_1, z_2)$ should hold if $(x_1 = x_2 \land y_1 = y_2) \to z_1 = z_2$ is true, where x_1, x_2, y_1, y_2 are natural numbers from Q (the rules for IfAnd are similar to those for If);

– $DB(x, z, \boldsymbol{y})$ should hold if $x = 0$ and z is the number $[P]$ of some predicate P in $\boldsymbol{q}$ or Σ such that $P(\boldsymbol{y}) \in D$:

$$DB(0, [P], \boldsymbol{y}) \leftarrow P(\boldsymbol{y}), \qquad \text{for all predicates } P \text{ in } \boldsymbol{q} \text{ and } \Sigma.$$

Now we can describe the construction of the main rule of Π, which mimicks $\boldsymbol{q}'$:

$$G \;\leftarrow\; \bigwedge_{k=1}^{M} \Gamma_k \;\land\; \bigwedge_{i=1}^{N} \Phi_i,$$

where G is a 0-ary goal predicate. The components, the Γ_k and the Φ_i, are defined as follows. In these definitions, we make use of the quantified variables $\boldsymbol{y}, \boldsymbol{u}, \boldsymbol{r}, \boldsymbol{w}$ with the same the intended meaning as in the previous construction; the meaning of additional quantified variables will be explained below. For each $1 \leq k \leq M$, let

$$\Gamma_k \quad = \quad \mathit{OneOf}(s_k, 1, N) \ \wedge \ \bigwedge_{i=1}^{N} \Big(\mathit{If}(s_k, i, r_i, [R_k]) \ \wedge \ \bigwedge_{j=1}^{L} \mathit{If}(s_k, i, u_{ij}, y_{kj}) \Big),$$

where s_k is a fresh variable meant to be the number i of the atom A_i to which $R_k(y_{k1}, \dots, y_{kL})$ is mapped; the variable s_k encodes the choice of the disjunct of Γ_k in the previous construction; cf. (8). For each $1 \leq i \leq N$, let

$$\Phi_i \quad = \quad \mathit{OneOf}(v_i, 0, K) \ \wedge \ \mathit{DB}(v_i, r_i, u_{i1}, \dots, u_{iL}) \ \wedge \bigwedge_{\tau = \forall \boldsymbol{x}\, (P_1(\boldsymbol{t}_1) \wedge \dots \wedge P_m(\boldsymbol{t}_m) \to \exists z\, P_0(\boldsymbol{t}_0)) \in \Sigma} \Big(\mathit{If}(v_i, [\tau], r_i, [P_0]) \ \wedge$$

$$\bigwedge_{t_{0j} \text{ is } a} \mathit{If}(v_i, [\tau], u_{ij}, a) \ \wedge \ \bigwedge_{t_{0j} \text{ is } x_l} \mathit{If}(v_i, [\tau], u_{ij}, w_{il}) \ \wedge \ \bigwedge_{t_{0j} \text{ is } z} \mathit{If}(v_i, [\tau], u_{ij}, \mathit{null}_i) \ \wedge \ \bigwedge_{k=1}^{m} \Psi_{\tau,i,k} \Big),$$

where v_i is meant to take the number $[\tau]$ of the tgd ($1 \leq [\tau] \leq S$) that derives the atom A_i or 0, if A_i is from the data instance: the second conjunct accounts for the case where A_i is an atom of the data instance and the last group of conjuncts for the case where A_i is obtained by an application of a tgd from Σ. Finally, for $i > 1$, we take

$$\Psi_{\tau,i,k} \quad = \quad \mathit{OneOf}(p_{ik}, 1, i-1) \wedge$$

$$\bigwedge_{i'=1}^{i-1} \Big(\mathit{IfAnd}(v_i, [\tau], p_{ik}, i', r_{i'}, [P_k]) \wedge \bigwedge_{t_{kj} = x_l} \mathit{IfAnd}(v_i, [\tau], p_{ik}, i', u_{i'j}, w_{il}) \wedge \bigwedge_{t_{kj} = a} \mathit{IfAnd}(v_i, [\tau], p_{ik}, i', u_{i'j}, a) \Big),$$

where, for every $1 \leq i \leq N$ and $1 \leq k \leq m$, p_{ik} is meant to be the number i' of the chase step that derives the k^{th} atom used in the i^{th} chase step. We take $\Psi_{\tau,1,k} = \mathit{OneOf}(v_1, 0, 0)$, which ensures $v_1 = 0$.

It is straightforward to verify that (Π, G) is indeed equivalent to $\boldsymbol{q}'$, thus establishing (*ii*). ❑

As sets of linear tgds of bounded arity and sets of sticky tgds of bounded arity enjoy the PWP, we obtain:

Corollary 12. *Any CQ and any set of linear tgds of bounded arity* (*in particular, OWL 2 QL-ontology*) *have polynomial-size PE$_{\{0,1\}}$- and NDL$_{\{0,1\}}$-rewritings.*

Any CQ and any set of sticky tgds of bounded arity have polynomial-size PE$_{\{0,1\}}$- and NDL$_{\{0,1\}}$-rewritings.

The following result is an immediate consequence of the proof of Theorem 11; we shall use it to prove Lemma 15 in the next section:

Corollary 13. *Let $\boldsymbol{q}(\boldsymbol{x})$ be a CQ and Σ an ontology from a family with the PWP.*

(*i*) *There is a polynomial-size PE-formula $\gamma(\boldsymbol{x}, y_0, y_1)$ such that $\gamma(\boldsymbol{x}, 0, 1)$ is a PE$_{\{0,1\}}$-rewriting of $\boldsymbol{q}$ and Σ.*

(*ii*) *There is a polynomial-size NDL-query* $(\Pi, G(\boldsymbol{x}, y_0, y_1))$ *such that* $(\Pi, G(\boldsymbol{x}, 0, 1))$ *is an NDL$_{\{0,1\}}$-rewriting of $\boldsymbol{q}$ and Σ.*

By taking the formula $\exists y_0, y_1\, ((y_0 \neq y_1) \wedge \gamma(\boldsymbol{x}, y_0, y_1))$ with γ given in Corollary 13 (*i*), we also obtain the following result on polynomial FO-rewritability *over databases with at least two individuals*:

Corollary 14. *For any CQ $\boldsymbol{q}(\boldsymbol{x})$ and any ontology Σ from a family with the PWP, there is an FO-formula $\boldsymbol{q}'(\boldsymbol{x})$ such that its size is polynomial in $|\boldsymbol{q}|$ and $|\Sigma|$ and $(\Sigma, D) \models \boldsymbol{q}(\boldsymbol{a})$ if and only if $D \models \boldsymbol{q}'(\boldsymbol{a})$, for any data instance D with $|\Delta_D| \geq 2$ and any tuple $\boldsymbol{a}$ of elements in Δ_D.*

Note that the compact representation of the FO-rewriting in this corollary is achieved—compared to the FO-rewritings of CQs and *OWL 2 QL*-ontologies known so far—with the help of polynomially-many new existentially quantified variables that are used for guessing a derivation of the given CQ in the chase.

5. Separation Results

In this section, we again consider 'pure' rewritings (without additional constants) and prove two separation results saying that NDL-rewritings can be exponentially more succinct than PE-rewritings, and that FO-rewritings can be superpolynomially more succinct than PE-rewritings. To this end we need a construction for transforming Boolean formulas and circuits into rewritings.

Consider a family $f^1, f^2, \dots$ of monotone Boolean functions in NP and a corresponding family $\mathbf{C}^1, \mathbf{C}^2, \dots$ of polynomial-size Boolean circuits with nondeterministic inputs. Recall that in Section 3 we constructed a family ψ_n of CNFs encoding the $\mathbf{C}^n$. The CNF ψ_n, which contains $d \le 3|\mathbf{C}^n| + 1$ clauses with $m = |\mathbf{C}^n|$ Boolean variables, was then transformed into a set Σ_{f^n} of *OWL 2 QL*-tgds and a Boolean CQ $\boldsymbol{q}_{f^n}$ such that

$$(\Sigma_{f^n}, D_{\boldsymbol{\alpha}}) \models \boldsymbol{q}_{f^n} \qquad \text{if and only if} \qquad f^n(\boldsymbol{\alpha}) = 1, \qquad \text{for all } \boldsymbol{\alpha} \in \{0,1\}^n.$$

Consider now the *OWL 2 QL*-ontology $\Sigma^*_{f^n}$ that extends Σ_{f^n} with the negative constraints

$$\forall x\, (A_0(x) \land B(x) \to \bot), \qquad \text{for } B(x) \in \Theta,$$

where Θ is the set comprising the following formulas:

$$\begin{array}{ll} \exists y\, P(x,y), & \\ A_i(x),\ \ X_i^0(x),\ \ X_i^1(x), & \text{for } 1 \le i \le m,\\ Z_{i,j}(x), & \text{for } 0 \le i \le m \text{ and } 1 \le j \le d \text{ with } (i,j) \notin \{(0,1),\dots,(0,n)\}. \end{array}$$

We observe that $|\Sigma^*_{f^n}| = O(|\mathbf{C}^n|^2)$ and the claims of Lemma 8 are equally applicable to $\Sigma^*_{f^n}$ (the proof requires that the query $\boldsymbol{q}_{f^n}$ and the ontology $\Sigma_{f^n}/\Sigma^*_{f^n}$ give 'correct' answers only for data $D_{\boldsymbol{\alpha}}$ which, by definition, are consistent with the negative constraints above).

Lemma 15. *Let $f^1, f^2, \dots$ be a family of monotone Boolean functions in* NP *and $\mathbf{C}^1, \mathbf{C}^2, \dots$ a corresponding family of polynomial-size Boolean circuits with nondeterministic inputs.*

(*i*) *If the f^n are computed by Boolean formulas φ_n then there are a polynomial p and FO-rewritings $\boldsymbol{q}'_{f^n}$ of $\boldsymbol{q}_{f^n}$ and $\Sigma^*_{f^n}$ such that $|\boldsymbol{q}'_{f^n}| \le |\varphi_n| + p(|\mathbf{C}^n|)$.*

(*ii*) *If the f^n are computed by monotone Boolean circuits $\mathbf{B}^n$ then there are a polynomial p and NDL-rewritings (Π_{f^n}, G) of $\boldsymbol{q}_{f^n}$ and $\Sigma^*_{f^n}$ such that $|\Pi_{f^n}| \le 2|\mathbf{B}^n| + p(|\mathbf{C}^n|)$.*

Proof. (*i*) Let $\gamma_n(0,1)$ be the polynomial-size $\mathrm{PE}_{\{0,1\}}$-rewriting of $\boldsymbol{q}_{f^n}$ and Σ_{f^n} given by Corollary 13 (*i*). We denote by $\varphi_n(x)$ the result of replacing each propositional variable p_j in φ_n with the atom $Z_{0,j}(x)$, for $1 \le j \le n$, and consider the FO-query

$$\boldsymbol{q}'_{f^n} \ = \ \exists x \Big[A_0(x) \land \Big(\varphi_n(x) \lor \exists y\, (P(y,x) \land \gamma_n(x,y)) \lor \bigvee_{B(x)\in\Theta} B(x) \Big) \Big].$$

Clearly, $|\boldsymbol{q}'_{f^n}| = |\varphi_n| + p(|\mathbf{C}^n|)$, for a polynomial p (note that the size of both $\boldsymbol{q}_{f^n}$ and Σ_{f^n} is quadratic in $|\mathbf{C}^n|$ and their $\mathrm{PE}_{\{0,1\}}$-rewriting is in turn polynomial in their size). It remains to show that $\boldsymbol{q}'_{f^n}$ is an FO-rewriting of $\boldsymbol{q}_{f^n}$ and $\Sigma^*_{f^n}$.

Suppose $(\Sigma^*_{f^n}, D) \models \boldsymbol{q}_{f^n}$. If $(\Sigma^*_{f^n}, D)$ is inconsistent, it can only be due to the negative constraints of $\Sigma^*_{f^n}$, in which case there is $a \in \Delta_D$ and $B(x) \in \Theta$ such that $D \models A_0(a) \land B(a)$, whence $D \models \boldsymbol{q}'_{f^n}$. Otherwise, the chase of $(\Sigma^*_{f^n}, D)$ coincides with the chase of (Σ_{f^n}, D) and there is a homomorphism h from $\boldsymbol{q}_{f^n}$ into the chase of (Σ_{f^n}, D). Let $h(y_0) = a_0 \in \Delta_D$ (recall that y_0 is the root of the query $\boldsymbol{q}_{f^n}$). Clearly, $A(a_0) \in D$. Two cases are possible now. If there is some $a_1 \in \Delta_D \setminus \{a_0\}$ with $P(a_1, a_0) \in D$ then, as $\gamma_n(0,1)$ is a $\mathrm{PE}_{\{0,1\}}$-rewriting of $\boldsymbol{q}_{f^n}$ and Σ_{f^n}, we obtain $D \models \gamma_n(a_0, a_1)$, whence $D \models \boldsymbol{q}'_{f^n}$. Otherwise, $D \not\models \exists y\, P(y, a_0)$ and $Z_{i,j}(a_0) \in D$ only if $i = 0$ and $1 \le j \le n$. Consider $\boldsymbol{\alpha}$ defined by taking $\alpha_j = 1$ iff $Z_{0,j}(a_0) \in D$, for $1 \le j \le n$. We obtain $(\Sigma_{f^n}, D_{\boldsymbol{\alpha}}) \models \boldsymbol{q}_{f^n}$, and thus, by Lemma 7, $f^n(\boldsymbol{\alpha}) = 1$. So $D_{\boldsymbol{\alpha}} \models \varphi_n(a_0)$, whence $D \models \boldsymbol{q}'_{f^n}$.

Conversely, suppose $D \models \boldsymbol{q}'_{f^n}$. Then there is $a_0 \in \Delta_D$ with $A_0(a_0) \in D$. If the last disjunct of $\boldsymbol{q}'_{f^n}$ holds on a_0 then $(\Sigma^*_{f^n}, D)$ is inconsistent, whence $(\Sigma^*_{f^n}, D) \models \boldsymbol{q}_{f^n}$. So, from now on, we assume that the last disjunct does not hold on any $a \in \Delta_D$ with $A_0(a_0) \in D$, and so $(\Sigma^*_{f^n}, D)$ is consistent and its chase coincides with the chase of (Σ_{f^n}, D). Two cases are possible now. If the second disjunct holds then there is $a_1 \in \Delta_D \setminus \{a_0\}$ with $P(a_1, a_0) \in D$ (note that if $a_0 = a_1$

then $P(a_0, a_0) \in D$, and so $(\Sigma^*_{f^n}, D)$ is inconsistent, contrary to our assumption). Then, as $\gamma_n(0,1)$ is a $\mathrm{PE}_{\{0,1\}}$-rewriting of $\boldsymbol{q}_{f^n}$ and Σ_{f^n}, we obtain $(\Sigma_{f^n}, D) \models \boldsymbol{q}_{f^n}$. Otherwise, the first disjunct, $\varphi_n(x)$, holds on a_0, $D \not\models \exists y\, P(y, a_0)$ and $Z_{i,j}(a_0) \in D$ only if $i = 0$ and $1 \leq j \leq n$. Consider $\boldsymbol{\alpha}$ defined by taking $\alpha_j = 1$ iff $Z_{0,j}(a_0) \in D$, for $1 \leq j \leq n$. As φ_n computes f^n, we have $f^n(\boldsymbol{\alpha}) = 1$, and so, by Lemma 7, $(\Sigma_{f^n}, D) \models \boldsymbol{q}_{f^n}$. In either case, $(\Sigma^*_{f^n}, D) \models \boldsymbol{q}_{f^n}$.

(*ii*) Let $(\Phi_n, F(0,1))$ be the polynomial-size $\mathrm{NDL}_{\{0,1\}}$-rewriting of $\boldsymbol{q}_{f^n}$ and Σ_{f^n} given by Corollary 13 (*ii*). We denote by Ξ_n the NDL-program built from $\mathbf{B}^n$ by replacing each input with the respective unary predicate atom $Z_{0,j}(x)$, for $1 \leq j \leq n$. More precisely, for each gate g_i with inputs $g_{i'}$ and $g_{i''}$ in the monotone Boolean circuit $\mathbf{B}^n$, we take a unary predicate $Q_i(x)$ and include the following rules in Ξ_n:

$$Q_i(x) \leftarrow Q_{i'}(x), Q_{i''}(x), \qquad \text{if } g_i = g_{i'} \wedge g_{i''}, \qquad \text{and} \qquad \begin{array}{ll} Q_i(x) & \leftarrow Q_{i'}(x), \\ Q_i(x) & \leftarrow Q_{i''}(x), \end{array} \qquad \text{if } g_i = g_{i'} \vee g_{i''}$$

(if $g_{i'}$ is the j^{th} input of $\mathbf{B}^n$ then $Q_{i'}(x)$ denotes $Z_{0,j}(x)$; and similarly for $g_{i''}$). Consider now the NDL-query (Π_{f^n}, G), where the goal G is a fresh 0-ary predicate, and Π_{f^n} comprises the rules of Φ_n and Ξ_n as well as the following rules:

$$\begin{array}{ll} G \leftarrow A_0(x), Q_{|\mathbf{B}^n|}(x), & \\ G \leftarrow A_0(x), P(y,x), F(x,y), & \\ G \leftarrow A_0(x), B(x), & \text{for all } B(x) \in \Theta \end{array}$$

(recall that $Q_{|\mathbf{B}^n|}$ corresponds to the output gate of $\mathbf{B}^n$). Clearly, $|\Pi_{f^n}| \leq 2|\mathbf{B}^n| + p(|\mathbf{C}^n|)$, for a polynomial p (note that the size of both $\boldsymbol{q}_{f^n}$ and Σ_{f^n} is quadratic in $|\mathbf{C}^n|$ and their $\mathrm{NDL}_{\{0,1\}}$-rewriting is in turn polynomial in their size). We claim that (Π_{f^n}, G) is an NDL-rewriting of $\boldsymbol{q}_{f^n}$ and $\Sigma^*_{f^n}$; the proof is as in case (*i*). ❑

We are now in a position to show that NDL-rewritings can be exponentially more succinct than PE-rewritings. To this end, we use the Boolean function $\textsc{Gen}_{m^3}$ of m^3 variables x_{ijk}, $1 \leq i, j, k \leq m$, defined as follows. We say that 1 *generates* $k \leq m$ if either $k = 1$ or $x_{ijk} = 1$, for some i and j, and 1 generates both i and j. $\textsc{Gen}_{m^3}(x_{111}, \dots, x_{mmm})$ returns 1 if and only if 1 generates m. This monotone function, also known as Path System Accessibility [22], is computable by polynomial-size monotone circuits [58]. On the other hand, any monotone formula computing $\textsc{Gen}_{m^3}$ is of size at least 2^{m^ε}, for some $\varepsilon > 0$ [48].

Theorem 16. *There is a sequence of CQs $\boldsymbol{q}_n$ of size $O(n)$ and OWL 2 QL-ontologies Σ_n of size $O(n)$ that have polynomial-size NDL-rewritings, but any PE-rewritings of $\boldsymbol{q}_n$ and Σ_n are of size $\geq 2^{n^\varepsilon}$, for some $\varepsilon > 0$.*

Proof. It is known that $\textsc{Gen}_{m^3}$ can be computed by monotone Boolean circuits of size $p(m)$, for a polynomial p. So, for each n, we can choose a suitable $m = \Theta(n^\delta)$, with a fixed $\delta > 0$, such that the family of functions $f^n = \textsc{Gen}_{m^3}$ gives rise to the queries $\boldsymbol{q}_n = \boldsymbol{q}_{f^n}$ and *OWL 2 QL*-ontologies $\Sigma_n = \Sigma^*_{f^n}$ of size $O(n)$. By Lemma 15 (*ii*), there are NDL-rewritings of $\boldsymbol{q}_n$ and Σ_n of size polynomial in n. However, by Lemma 8 (*ii*), any PE-rewritings for $\boldsymbol{q}_n$ and Σ_n are of size $\geq 2^{m^{\varepsilon_0}}$, for some $\varepsilon_0 > 0$. Then there is $\varepsilon > 0$ such that any PE-rewritings of $\boldsymbol{q}_n$ and Σ_n are of size $\geq 2^{n^\varepsilon}$. ❑

FO-rewritings can also be substantially shorter than the PE-rewritings. To show this, we need the function $\textsc{Matching}_{2m}$ of m^2 variables e_{ij}, $1 \leq i, j \leq m$, that returns 1 if there is a *perfect matching* in the bipartite graph G with m vertices in each part, which contains an edge $\{i, j\}$ if and only if $e_{ij} = 1$; that is, it returns 1 if there is a subset E of edges in G such that every node of G occurs exactly once in E. It is not hard to see that $\textsc{Matching}_{2m}$ can be computed by a Boolean circuit with m^2 nondeterministic inputs and $O(m^2)$ gates. On the other hand, monotone Boolean formulas computing $\textsc{Matching}_{2m}$ are exponential, $2^{\Omega(m)}$ [49]; but there are non-monotone Boolean formulas computing this function and having size $m^{O(\log m)}$ [10]. So, we can use the standard padding trick from circuit complexity [3, page 57] to show that FO-rewritings can be superpolynomially more succinct than PE-rewritings:

Theorem 17. *There is a sequence of CQs $\boldsymbol{q}_n$ of size $O(n)$ and OWL 2 QL-ontologies Σ_n size $O(n)$ that have polynomial-size FO-rewritings, but any PE-rewritings of $\boldsymbol{q}_n$ and Σ_n are of size $\geq 2^{\Omega(2^{\log^{1/2} n})}$.*

Proof. We define f^n to be a slightly modified $\textsc{Matching}_{2m}$ with $m = \lfloor 2^{\log^{1/2} n} \rfloor$: namely, f^n has $\max(\lfloor n^{1/4} \rfloor, m^2)$ variables, of which m^2 are the proper variables of $\textsc{Matching}_{2m}$, while the rest are dummy variables used for padding (note that $\lfloor n^{1/4} \rfloor > m^2$, for all sufficiently large n). Using Lemma 15 (*i*) and observing that $m^{O(\log m)} = n^{O(1)}$, we obtain a polynomial upper bound for the size of FO-rewritings. The required superpolynomial lower bound for PE-rewritings follows from Lemma 8 (*ii*). ❑

Unfortunately, no separation results for FO- and NDL-rewritings are known at the moment. As follows from the connection between rewritings and various computation models for monotone Boolean functions established in this article, such results would imply the corresponding separation results for formulas and monotone circuits, thereby giving solutions to major open problems in Boolean circuit complexity [29].

6. Conclusions

We have shown in this article that FO-rewritability of conjunctive queries and *OWL 2 QL*-ontologies does not yet mean that database systems can evaluate the rewritings as efficiently as they usually do for standard SQL queries. Indeed, the rewritings can be prohibitively large and/or complex compared to the user queries. We have also seen that the size of rewritings depends on the logical and non-logical means we want or are allowed to use. These results clearly indicate that more theoretical and experimental research is needed to make the OBDA paradigm successful. Here we briefly outline some important directions for future research that are related to this article.

On the one (theoretical) hand, we obviously need various conditions ensuring efficient OBDA, with first promising steps having already been made. For example, a sufficient semantic-based condition on CQs and *OWL 2 QL*-ontologies that guarantees polynomial PE-rewritability has been obtained in [33]. It has also been demonstrated [30, 31] that there exist polynomial-size NDL-rewritings of CQs and *OWL 2 QL*-ontologies of depth 1 (whose chases do not contain two labelled nulls that are involved in some relation), as well as polynomial-size PE-rewritings of tree-shaped CQs (but not of arbitrary ones). For tree-shaped Boolean CQs $\boldsymbol{q}$, the problem '$(\Sigma, D) \models \boldsymbol{q}$?' turns out to be fixed-parameter tractable (with parameter $|\boldsymbol{q}|$) [31]. Moreover, any tree-shaped CQ and *OWL 2 QL* ontology with polynomially-many tree-witnesses have a polynomial-size NDL-rewriting [8]. A kind of preservation result has been obtained in [8]: if CQs in some class can be evaluated in polynomial time over plain databases, then answering CQs in that class over *OWL 2 QL*-ontologies without role inclusion axioms, that is, without tgds of the form $\forall x, y\,(P(x, y) \to R(x, y))$, is also tractable (a polynomial-time NDL-rewriting algorithm is given for acyclic CQs). These initial results open a way to a more comprehensive description of classes of queries and ontologies with and without polynomial rewritability. To fully understand the complexity of OBDA with *OWL 2 QL*-ontologies, we also plan to investigate the size of rewritings over a fixed ontology and the size of rewritings of tree-shaped CQs and ontologies of bounded depth.

On the other (practical) hand, we have to study the structure of queries and ontologies that can typically be used in OBDA systems. The recent experiments [20, 35, 46, 54, 52, 51] indicate that rewritings of the available 'real-world' CQs and ontologies are often of acceptable size and can be further optimised using various techniques. However, the ontologies used in those experiments do not seem to be sufficiently representative. It would also be interesting to evaluate performance of database systems on rewritings with additional quantifiers and special constants, which can be used to encode nondeterministic guesses in a compact way as in Section 4 (another rewriting of [33] employs a single special constant to guess whether an existentially quantified variable in the query is matched in Δ_D or in the labelled nulls). Additional constants are also used in the combined approach to OBDA [40, 37, 38, 39], where they represent the labelled nulls in the database.

Acknowledgements. This research was partially funded by EPSRC joint grants EP/H051511 and EP/H05099X: "ExODA: Integrating Description Logics and Database Technologies for Expressive Ontology-Based Data Access."

References

[1] Abiteboul, S., Hull, R., and Vianu, V. 1995. *Foundations of Databases*. Addison-Wesley.
[2] Alon, N. and Boppana, R. 1987. The monotone circuit complexity of Boolean functions. *Combinatorica 7,* 1, 1–22.
[3] Arora, S. and Barak, B. 2009. *Computational Complexity: A Modern Approach* 1st Ed. Cambridge University Press, New York, NY, USA.
[4] Artale, A., Calvanese, D., Kontchakov, R., and Zakharyaschev, M. 2009. The DL-Lite family and relations. *Journal of Artificial Intelligence Research (JAIR) 36*, 1–69.
[5] Avigad, J. 2003. Eliminating definitions and Skolem functions in first-order logic. *ACM Transactions on Computational Logic 4,* 3, 402–415.

[6] Baget, J.-F., Leclère, M., Mugnier, M.-L., and Salvat, E. 2009. Extending decidable cases for rules with existential variables. In *Proc. of the 21st Int. Joint Conf. on Artificial Intelligence (IJCAI 2009)*. IJCAI, 677–682.

[7] Baget, J.-F., Leclère, M., Mugnier, M.-L., and Salvat, E. 2011. On rules with existential variables: Walking the decidability line. *Artificial Intelligence 175,* 9–10, 1620–1654.

[8] Bienvenu, M., Ortiz, M., Simkus, M., and Xiao, G. 2013a. Tractable queries for lightweight description logics. In *Proc. of the 23rd Int. Joint Conf. on Artificial Intelligence (IJCAI 2013)*. AAAI Press/IJCAI, 768–774.

[9] Bienvenu, M., ten Cate, B., Lutz, C., and Wolter, F. 2013b. Ontology-based data access: a study through disjunctive datalog, CSP, and MMSNP. In *Proc. of the 32nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS 2013)*. ACM, 213–224.

[10] Borodin, A., von zur Gathen, J., and Hopcroft, J. E. 1982. Fast parallel matrix and gcd computations. In *Proc. of the 23rd Annual Symposium on Foundations of Computer Science (FOCS'82)*. IEEE Computer Society, 65–71.

[11] Calì, A., Gottlob, G., and Lukasiewicz, T. 2012a. A general datalog-based framework for tractable query answering over ontologies. *Journal of Web Semantics 14*, 57–83.

[12] Calì, A., Gottlob, G., and Pieris, A. 2010. Advanced processing for ontological queries. *PVLDB 3,* 1, 554–565.

[13] Calì, A., Gottlob, G., and Pieris, A. 2012b. Towards more expressive ontology languages: The query answering problem. *Artificial Intelligence 193*, 87–128.

[14] Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., and Rosati, R. 2005. *DL-Lite*: Tractable description logics for ontologies. In *Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI 2005)*. AAAI Press, 602–607.

[15] Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., and Rosati, R. 2006. Data complexity of query answering in description logics. In *Proc. of the 10th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2006)*. AAAI Press, 260–270.

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

[17] Chekuri, C. and Rajaraman, A. 2000. Conjunctive query containment revisited. *Theoretical Computer Science 239,* 2, 211–229.

[18] Chortaras, A., Trivela, D., and Stamou, G. 2011. Optimized query rewriting for OWL 2 QL. In *Proc. of the 23rd Int. Conf. on Automated Deduction (CADE-23)*. Lecture Notes in Computer Science Series, vol. 6803. Springer, 192–206.

[19] Dolby, J., Fokoue, A., Kalyanpur, A., Ma, L., Schonberg, E., Srinivas, K., and Sun, X. 2008. Scalable grounded conjunctive query evaluation over large and expressive knowledge bases. In *Proc. of the 7th Int. Semantic Web Conf. (ISWC 2008)*. Lecture Notes in Computer Science Series, vol. 5318. Springer, 403–418.

[20] Eiter, T., Ortiz, M., Šimkus, M., Tran, T.-K., and Xiao, G. 2012. Query rewriting for Horn-SHIQ plus rules. In *Proc. of the 26th AAAI Conf. on Artificial Intelligence (AAAI 2012)*. AAAI Press.

[21] Flum, J. and Grohe, M. 2006. *Parameterized Complexity Theory*. EATCS Series: Texts in Theoretical Computer Science. Springer.

[22] Garey, M. and Johnson, D. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA.

[23] Gottlob, G., Manna, M., and Pieris, A. 2014. Polynomial combined rewritings for existential rules. In *Proc. of the 14th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2014)*. AAAI Press.

[24] Gottlob, G., Orsi, G., and Pieris, A. Query rewriting and optimization for ontological databases. To Appear.

[25] Gottlob, G., Orsi, G., and Pieris, A. 2011. Ontological queries: Rewriting and optimization. In *Proc. of the 27th Int. Conf. on Data Engineering (ICDE 2011)*. IEEE Computer Society, 2–13.

[26] Gottlob, G. and Schwentick, T. 2012. Rewriting ontological queries into small nonrecursive datalog programs. In *Proc. of the 13th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2012)*. AAAI Press, 254–263.

[27] Grohe, M., Schwentick, T., and Segoufin, L. 2001. When is the evaluation of conjunctive queries tractable? In *Proc. of the 33rd ACM SIGACT Symposium on Theory of Computing (STOC'01)*. ACM, 657–666.

[28] Heymans, S., Ma, L., Anicic, D., Ma, Z., Steinmetz, N., Pan, Y., Mei, J., Fokoue, A., Kalyanpur, A., Kershenbaum, A., Schonberg, E., Srinivas, K., Feier, C., Hench, G., Wetzstein, B., and Keller, U. 2008. Ontology reasoning with large data repositories. In *Ontology Management, Semantic Web, Semantic Web Services, and Business Applications*. Semantic Web and Beyond Series, vol. 7. Springer, 89–128.

[29] Jukna, S. 2012. *Boolean Function Complexity: Advances and Frontiers*. Springer.

[30] Kikot, S., Kontchakov, R., Podolskii, V., and Zakharyaschev, M. 2013. Query rewriting over shallow ontologies. In *Proc. of the 26th Int. Workshop on Description Logics (DL 2013)*. Vol. 1014. CEUR-WS, 316–327.

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

[32] Kikot, S., Kontchakov, R., Podolskii, V. V., and Zakharyaschev, M. 2012a. Exponential lower bounds and separation for query rewriting. In *Proc. of the 39th Int. Colloquium on Automata, Languages, and Programming (ICALP 2012), Part II*. Lecture Notes in Computer Science Series, vol. 7392. Springer, 263–274.

[33] Kikot, S., Kontchakov, R., and Zakharyaschev, M. 2012b. Conjunctive query answering with OWL 2 QL. In *Proc. of the 13th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2012)*. AAAI Press, 275–285.

[34] Kolaitis, P. G. and Vardi, M. Y. 1998. Conjunctive-query containment and constraint satisfaction. In *Proc. of the 17th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'98)*. ACM Press, 205–213.

[35] König, M., Leclère, M., Mugnier, M.-L., and Thomazo, M. 2012. A sound and complete backward chaining algorithm for existential rules. In *Proc. of the 6th Int. Conf. on Web Reasoning and Rule Systems (RR 2012)*. Lecture Notes in Computer Science Series, vol. 7497. Springer, 122–138.

[36] König, M., Leclère, M., Mugnier, M.-L., and Thomazo, M. 2013. On the exploration of the query rewriting space with existential rules. In *Proc. of the 7th Int. Conf. on Web Reasoning and Rule Systems (RR 2013)*. Lecture Notes in Computer Science Series, vol. 7994. Springer, 123–137.

[37] Kontchakov, R., Lutz, C., Toman, D., Wolter, F., and Zakharyaschev, M. 2010. The combined approach to query answering in DL-Lite. In *Proc. of the 10th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2010)*. AAAI Press.

[38] Kontchakov, R., Lutz, C., Toman, D., Wolter, F., and Zakharyaschev, M. 2011. The combined approach to ontology-based data access. In

Proc. of the 22nd Int. Joint Conf. on Artificial Intelligence (IJCAI 2011). AAAI Press, 2656–2661.

[39] Lutz, C., Seylan, I., Toman, D., and Wolter, F. 2013. The combined approach to OBDA: Taming role hierarchies using filters. In *Proc. of the 12th Int. Semantic Web Conf. (ISWC 2013)*. Lecture Notes in Computer Science Series, vol. 8218. Springer, 314–330.

[40] Lutz, C., Toman, D., and Wolter, F. 2009. 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)*. IJCAI, 2070–2075.

[41] McMahan, B. J., Pan, G., Porter, P., and Vardi, M. Y. 2004. Projection pushing revisited. In *Proc. of the 9th Int. Conf. on Extending Database Technology (EDBT)*. Lecture Notes in Computer Science Series, vol. 2992. Springer, 441–458.

[42] Meyer, A. R. and Fischer, M. J. 1971. Economy of description by automata, grammars, and formal systems. In *Proc. of the 12th Annual Symposium on Switching and Automata Theory (SWAT/FOCS'71)*. IEEE Computer Society, 188–191.

[43] Orsi, G. and Pieris, A. 2011. Optimizing query answering under ontological constraints. *PVLDB 4,* 11, 1004–1015.

[44] Ortiz, M., Rudolph, S., and Simkus, M. 2011. Query answering in the Horn fragments of the description logics SHOIQ and SROIQ. In *Proc. of the 22nd Int. Joint Conf. on Artificial Intelligence (IJCAI 2011)*. IJCAI/AAAI, 1039–1044.

[45] Pérez-Urbina, H., Motik, B., and Horrocks, I. 2009. A comparison of query rewriting techniques for DL-Lite. In *Proc. of the 22nd Int. Workshop on Description Logics (DL 2009)*. Vol. 477. CEUR-WS.

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

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

[48] Raz, R. and McKenzie, P. 1997. Separation of the monotone NC hierarchy. In *Proc. of the 38th Annual Symposium on Foundations of Computer Science (FOCS'97)*. IEEE Computer Society, 234–243.

[49] Raz, R. and Wigderson, A. 1992. Monotone circuits for matching require linear depth. *Journal of the ACM 39,* 3, 736–744.

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

[51] Rodríguez-Muro, M., Kontchakov, R., and Zakharyaschev, M. 2013a. Ontology-based data access: Ontop of databases. In *Proc. of the 12th Int. Semantic Web Conf. (ISWC 2013)*. Lecture Notes in Computer Science Series, vol. 8218. Springer, 558–573.

[52] Rodríguez-Muro, M., Kontchakov, R., and Zakharyaschev, M. 2013b. Ontop at work. In *Proc. of the 10th Int. Workshop on OWL: Experiences and Directions (OWLED 2013)*. Vol. 1080. CEUR-WS.

[53] Rosati, R. 2007. On conjunctive query answering in EL. In *Proc. of the 2007 Int. Workshop on Description Logics (DL 2007)*. Vol. 250. CEUR-WS.

[54] Rosati, R. 2012. Prexto: Query rewriting under extensional constraints in DL-Lite. In *Proc. of the 9th Extended Semantic Web Conf. (EWSC 2012)*. Lecture Notes in Computer Science Series, vol. 7295. Springer, 360–374.

[55] Rosati, R. and Almatelli, A. 2010. Improving query answering over DL-Lite ontologies. In *Proc. of the 10th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2010)*. AAAI Press, 290–300.

[56] Rossman, B. 2008. Homomorphism preservation theorems. *Journal of the ACM 55,* 3.

[57] Salvat, E. and Mugnier, M.-L. 1996. Sound and complete forward and backward chaining of graph rules. In *Proc. of the 4th Int. Conf. on Conceptual Structures (ICCS'96)*. Lecture Notes in Computer Science Series, vol. 1115. Springer, 248–262.

[58] Stewart, I. A. 1994. Logical description of monotone NP problems. *Journal of Logic and Computation 4,* 4, 337–357.

[59] Tseitin, G. 1983. On the complexity of derivation in propositional calculus. In *Automation of Reasoning 2: Classical Papers on Computational Logic 1967–1970*. Springer, 466–483.

[60] Yannakakis, M. 1981. Algorithms for acyclic database schemes. In *Proc. of the 7th Int. Conf. on Very Large Data Bases (VLDB'81)*. IEEE Computer Society, 82–94.