Checking the Data Complexity of Ontology-Mediated Queries: a Case Study with Non-Uniform CSPs and Polyanna

O. Gerasimova[1], S. Kikot[2], and M. Zakharyaschev[3]

[1] National Research University Higher School of Economics, Moscow, Russia
[2] University of Oxford, U.K.
[3] Birkbeck, University of London, U.K.

Abstract. It has recently been shown that first-order- and datalog-rewritability of ontology-mediated queries (OMQs) with expressive ontologies can be checked in NEXPTIME using a reduction to CSPs. In this paper, we present a case study for OMQs with Boolean conjunctive queries and a fixed ontology consisting of a single covering axiom $A \sqsubseteq F \sqcup T$, possibly supplemented with a disjointness axiom for T and F. The ultimate aim is to classify such OMQs according to their data complexity: AC^0, L, NL, P or CONP. We report on our experience with trying to distinguish between OMQs in P and CONP using the reduction to CSPs and the Polyanna software for finding polymorphisms.

1 Introduction

Description logics (DLs) [4] have been tailored—by carefully picking and restricting various constructs that are relevant to intended applications—to make sure that reasoning with all ontologies in a given DL can uniformly be done in a given complexity class. For example, concept subsumption can be checked in EXPTIME for all $\mathcal{ALC}$-ontologies, in P for all $\mathcal{EL}$-ontologies, and in NL for all *DL-Lite*-ontologies.

In ontology-mediated query (OMQ) answering, a typical reasoning problem is to check whether a Boolean query $\boldsymbol{q}$ holds in every model of an ontology $\mathcal{T}$ and a data instance $\mathcal{D}$. In the context of ontology-based data access (OBDA) and management [28, 35], this problem is solved by reducing answering the OMQ $\boldsymbol{Q} = (\mathcal{T}, \boldsymbol{q})$ over $\mathcal{D}$ to standard database query evaluation over $\mathcal{D}$. If the target database query language is first-order logic, then such a reduction is possible for $\boldsymbol{Q}$ just in case answering it can be done in AC^0 for data complexity. If a reduction to first-order queries with (deterministic) transitive closure is acceptable, then answering $\boldsymbol{Q}$ should be done in NL (respectively, L) for data complexity. In terms of the data complexity measure, OMQ answering with conjunctive queries (CQs) can uniformly be done in CONP for all $\mathcal{ALC}$-ontologies, in P for all $\mathcal{EL}$-ontologies, and in AC^0 for all *DL-Lite*-ontologies.

In OBDA practice, an ontology $\mathcal{T}$ is designed by a domain expert to capture the natural vocabulary of the intended end-users, who formulate their queries in terms of that vocabulary and execute them using an OBDA system such as Mastro [10] or Ontop [30, 11]. Thus, from the user's point of view, the ontology $\mathcal{T}$ is fixed. Moreover, the class of queries the user is interested in could also be limited. For example, the NPD

FactPages ontology[4], used for testing OBDA in industry [22, 20], contains covering axioms of the form $A \sqsubseteq B_1 \sqcup \cdots \sqcup B_n$, which are not allowed in *DL-Lite* as there exist coNP-hard OMQs with such axioms. However, all of the practically important OMQs with the NPD FactPages ontology we know of can be answered in AC^0. Also answering CQs mediated by covering axioms can model some attacks in the setting of sensitive information disclosure [7].

These observations have lead to the following non-uniform problems: (i) What is the worst-case data complexity of answering OMQs with a fixed ontology and arbitrary CQs? (ii) What is the data complexity of answering a given single OMQ?

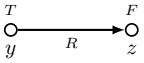
A systematic investigation of these problems was launched in [24, 8, 25]. In particular, [8] discovered a remarkable connection between OMQs and constraint satisfaction problems (CSPs) and used it to show that deciding FO-rewritability and datalog-rewritability of OMQs with $\mathcal{SHIU}$ ontologies is NExpTime-complete.

In this article, we are concerned with a very special case of the non-uniform problem (ii) above: classify the OMQs with the fixed ontology $\mathcal{C}ov_A = \{A \sqsubseteq F \sqcup T\}$ and arbitrary CQs according to their data complexity. We also consider three variants of $\mathcal{C}ov_A$, namely, $\mathcal{C}ov_\top = \{\top \sqsubseteq F \sqcup T\}$, $\mathcal{C}ov_\top^\bot = \{\top \sqsubseteq F \sqcup T,\ \ F \sqcap T \sqsubseteq \bot\}$ and $\mathcal{C}ov_A^\bot = \{A \sqsubseteq F \sqcup T,\ \ F \sqcap T \sqsubseteq \bot\}$ with top $\top$ and bottom $\bot$ concepts. It turns out that a single covering axiom, possibly supplemented with a disjointness axiom, gives rise to a surprisingly non-trivial and diverse class of OMQs. To illustrate, we show and discuss a few simple examples.

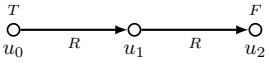
Appetisers

Suppose we are interested in querying digraphs of social network users, in which only some of the users have specified their gender. Let F mean 'female', T 'male' and R the 'follows' relation.

Example 1. Our first OMQ $\boldsymbol{Q} = (\mathcal{C}ov_\top, \boldsymbol{q})$ with $\boldsymbol{q} = \exists y, z\, (T(y) \land R(y, z) \land F(z))$ is supposed to check whether one can claim with certainty that, in given a data instance, there is always a man who follows a woman. We draw the CQ $\boldsymbol{q}$ as the labelled digraph



Now, consider the data instance $\mathcal{D} = \{T(u_0), R(u_0, u_1), R(u_1, u_2), F(u_2)\}$, which can be depicted as the labelled digraph



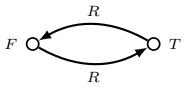
In every model of $\mathcal{C}ov_\top$ extending $\mathcal{D}$, we must have $T(u_1)$ or $F(u_1)$. In the former case, $\boldsymbol{q}$ is satisfied by the assignment $y \mapsto u_1, z \mapsto u_2$, in the latter one by $y \mapsto u_0$, $z \mapsto u_1$. It follows that the certain answer to $\boldsymbol{Q}$ over $\mathcal{D}$ is yes.

More generally, it is readily seen that the certain answer to $\boldsymbol{Q}$ over any given $\mathcal{D}$ is yes iff $\mathcal{D}$ contains an R-path from a T-vertex to an F-vertex. As known from the

[4] http://sws.ifi.uio.no/project/npd-v2/

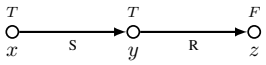
basic computational complexity theory [3], the reachability problem in digraphs is NL-complete (NL stands for Nondeterministic Logarithmic-space), and so answering $\boldsymbol{Q}$ is NL-complete for data complexity.

Example 2. Consider next the OMQ $\boldsymbol{Q} = (\mathcal{C}ov_\top, \boldsymbol{q})$ with $\boldsymbol{q}$ given by the digraph

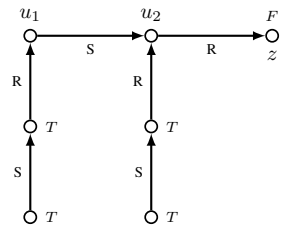


which checks if one can claim with certainty that there are a man and a woman who follow each other. In this case, answering $\boldsymbol{Q}$ is L-complete (L stands for deterministic Logarithmic-space) for data complexity, that is, as complex as reachability in undirected graphs. Indeed, to show that $\boldsymbol{Q}$ can be answered in L, with each data instance $\mathcal{D}$ we associate the undirected graph G with the same vertices as $\mathcal{D}$ connecting u and v by an edge iff $R(u, v)$ and $R(v, u)$ are both in $\mathcal{D}$. Then the answer to $\boldsymbol{Q}$ over $\mathcal{D}$ is yes iff G contains a path from a T-vertex to an F-vertex. To prove L-hardness, with any undirected graph G and a pair s, t of its vertices we associate a data instance $\mathcal{D}$ obtained by replacing each edge (u, v) in G by $R(u, v)$ and $R(v, u)$ and adding atoms $T(s)$ and $F(t)$. It is readily seen that the certain answer to $\boldsymbol{Q}$ over $\mathcal{D}$ is yes iff t is reachable from s in G.

Example 3. Now suppose $\boldsymbol{Q} = (\mathcal{C}ov_\top, \boldsymbol{q})$ and $\boldsymbol{q}$ looks as in the picture below



where S is another binary relation between the users. Consider the following data instance $\mathcal{D}$:



The certain answer to $\boldsymbol{Q}$ over $\mathcal{D}$ is yes. Indeed, in any model of $\mathcal{C}ov_\top$ based on $\mathcal{D}$, we have either $T(u_i)$ or $F(u_i)$, $i = 1, 2$. If $F(u_1)$ holds, then $\boldsymbol{q}$ maps onto the left vertical section of the model. Similarly, if $F(u_2)$ holds, then $\boldsymbol{q}$ maps onto the right vertical section. Otherwise, we have $T(u_1)$ and $T(u_2)$, in which case $\boldsymbol{q}$ maps onto the horizontal section.

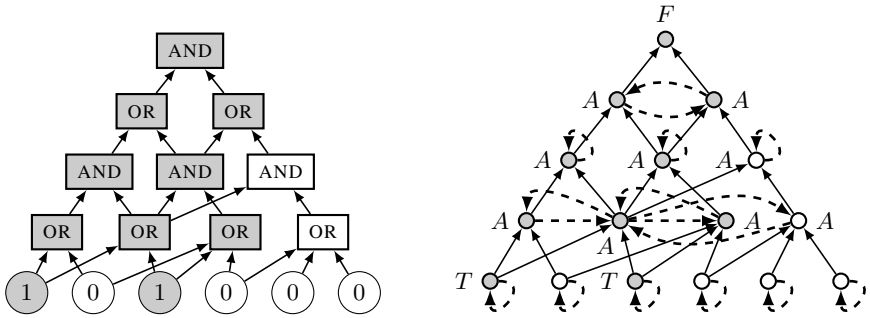
In general, for any data instance $\mathcal{D}$, the certain answer to $\boldsymbol{Q}$ over $\mathcal{D}$ is yes iff the following monadic datalog query with goal G, encoding our argument above, returns the answer yes over $\mathcal{D}$:

$$
\begin{aligned}
P(x) &\leftarrow T(x) \\
P(z) &\leftarrow P(x) \land S(x, y) \land P(y) \land R(y, z) \\
G &\leftarrow P(x) \land S(x, y) \land P(y) \land R(y, z) \land F(z)
\end{aligned}
$$

It follows that answering $\boldsymbol{Q}$ can be done in P (Polynomial time). It is not hard to show that this OMQ is P-hard. The proof is by reduction of the monotone circuit evaluation problem, which is known to be P-complete [27]. Without any loss of generality we assume that AND-nodes have two inputs and that the circuit consists of alternating layers of OR- and AND-nodes with an AND node at the top; an example of such a circuit is shown in the picture below. Now, given such a circuit $\mathcal{C}$ and an input $\boldsymbol{\alpha}$ for it, we define a data instance $\mathcal{D}_{\mathcal{C}}^{\boldsymbol{\alpha}}$ as the set of the following atoms:

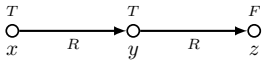
- $R(g, h)$, if a gate g is an input of a gate h;
- $S(g, h)$, if g and h are distinct inputs of some AND-gate;
- $S(g, g)$, if g is an input gate or a non-output AND-gate;
- $T(g)$, if g is an input gate with 1 under $\boldsymbol{\alpha}$;
- $F(g)$, for the only output gate g;
- $A(g)$, for those g that are neither inputs nor the output.

To illustrate, the picture below shows a monotone circuit $\mathcal{C}$, an input $\boldsymbol{\alpha}$ for it, and the data instance $\mathcal{D}_{\mathcal{C}}^{\boldsymbol{\alpha}}$, where the solid arrows represent R and the dashed ones S:



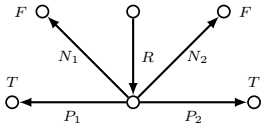
The reader can check that $\mathcal{C}(\boldsymbol{\alpha}) = 1$ iff the answer to $\boldsymbol{Q}$ over $\mathcal{D}_{\mathcal{C}}^{\boldsymbol{\alpha}}$ is yes. Indeed, the datalog program above computes the value of the circuit by placing P on those nodes which are evaluated into 1.

Example 4. Curiously enough, the OMQ $\boldsymbol{Q} = (\mathcal{C}ov_{\top}, \boldsymbol{q}')$ with $\boldsymbol{q}'$ obtained from $\boldsymbol{q}$ in the previous example by changing S to R

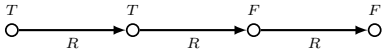


is NL-complete for data complexity, showing which is an instructive exercise. (Hint: prove that a data instance validates $\boldsymbol{Q}$ iff it has a path that starts with T, which is followed by T, and ends with F).

Example 5. It has been known since Schaerf's paper [32] that answering the OMQ $\boldsymbol{Q} = (\mathcal{C}ov_{\top}, \boldsymbol{q})$ with $\boldsymbol{q}$ below is coNP-complete for data complexity.



Schaerf showed coNP-hardness by encoding the satisfiability problem for 2+2-CNFs that consist of clauses of the form $P_1 \vee P_2 \vee \neg N_1 \vee \neg N_2$. The reader may find entertaining the task of showing that the OMQ $(\mathcal{C}ov_A, \boldsymbol{q})$ with $\boldsymbol{q}$ below is also coNP-complete.



One possible solution will be given in Section 6 below.

The remainder of this article is organised as follows. In the next section, we provide definitions of the basic notions we require later on. Then, in Section 3, we give a brief survey of related work. In Section 4, we explain by means of a simple example how detecting tractability of an path-OMQ can be reduced to checking tractability of a CSP. Then, in Section 5, we discuss how the program Polyanna [13], which was designed to check tractability of CSPs, can be used in the context of our case study for detecting whether answering a given OMQ with a 4-variable path CQ can be done in P or is coNP-hard. In Section 6, we sketch direct proofs of coNP-hardness using a reduction of 3SAT. In Section 7, we show how Polyanna can be used for constructing monadic datalog rewritings of tractable OMQs. Finally, in the appendix, we summarise what we know about the data complexity of answering the OMQs in the framework of our case study.

2 Preliminaries

In this paper, a *Boolean conjunctive query* (*CQ*) is any first-order (FO) sentence of the form $\boldsymbol{q} = \exists \boldsymbol{x}\, \varphi(\boldsymbol{x})$, where φ is a conjunction of unary or binary atoms whose variables are all among $\boldsymbol{x}$. We often regard CQs as *sets* of their atoms, depict them as labelled digraphs, and assume that all of our CQs are *connected* as graphs. By a *solitary occurrence* of F in a CQ $\boldsymbol{q}$ we mean any occurrence of $F(x)$ in $\boldsymbol{q}$, for some variable x, such that $T(x) \notin \boldsymbol{q}$; likewise, a *solitary occurrence* of T in $\boldsymbol{q}$ is any occurrence $T(x) \in \boldsymbol{q}$ such that $F(x) \notin \boldsymbol{q}$. We say that $\boldsymbol{q}$ is a *path CQ* if all the variables $x_0, \dots, x_n$ in $\boldsymbol{q}$ are ordered so that

- the binary atoms in $\boldsymbol{q}$ form a chain $R_1(x_0, x_1), \dots, R_n(x_{n-1}, x_n)$;
- the unary atoms in $\boldsymbol{q}$ are of the form $T(x_i)$ and $F(x_j)$, for some i and j with $0 \leq i, j \leq n$.

By *answering* an *ontology-mediated query* (OMQ) $\boldsymbol{Q} = (\mathcal{T}, \boldsymbol{q})$, where $\mathcal{T}$ is one of the following ontologies (or TBoxes)

$$\mathcal{C}ov_A = \{A \sqsubseteq F \sqcup T\}, \qquad \mathcal{C}ov_A^{\perp} = \{A \sqsubseteq F \sqcup T,\ \ F \sqcap T \sqsubseteq \bot\},$$
$$\mathcal{C}ov_{\top} = \{\top \sqsubseteq F \sqcup T\}, \qquad \mathcal{C}ov_{\top}^{\perp} = \{\top \sqsubseteq F \sqcup T,\ \ F \sqcap T \sqsubseteq \bot\},$$

we understand the problem of checking, given a data instance (or ABox) $\mathcal{A}$, whether $\boldsymbol{q}$ holds in every model of $\mathcal{T} \cup \mathcal{A}$, in which case we write $\mathcal{T}, \mathcal{A} \models \boldsymbol{q}$. For every $\boldsymbol{Q}$, this problem is clearly in coNP for data complexity. It is in the complexity class AC^0 if there is an FO-sentence $\boldsymbol{q}'$, called an *FO-rewriting* of $\boldsymbol{Q}$, such that $\mathcal{T}, \mathcal{A} \models \boldsymbol{q}$ iff $\mathcal{A} \models \boldsymbol{q}'$, for any ABox $\mathcal{A}$.

A *datalog program*, Π, is a finite set of *rules* $\forall \boldsymbol{x}\,(\gamma_0 \leftarrow \gamma_1 \wedge \dots \wedge \gamma_m)$, where each γ_i is an atom $P(\boldsymbol{y})$ with $\boldsymbol{y} \subseteq \boldsymbol{x}$. (As usual, we omit $\forall \boldsymbol{x}$.) The atom γ_0 is the *head* of the rule, and $\gamma_1, \dots, \gamma_m$ its *body*. All the variables in the head must occur in the body. The predicates in the head of rules are *IDB predicates*, the rest *EDB predicates* [1].

A *datalog query* is a pair (Π, G), where Π is a datalog program and G an 0-ary atom, the *goal*. The *answer* to (Π, G) over an ABox $\mathcal{A}$ is 'yes' if G holds in the FO-structure obtained by closing $\mathcal{A}$ under Π, in which case we write $\Pi, \mathcal{A} \models G$. A datalog query (Π, G) is a *datalog rewriting* of an OMQ $\boldsymbol{Q} = (\mathcal{T}, \boldsymbol{q})$ in case $\mathcal{T}, \mathcal{A} \models \boldsymbol{q}$ iff $\Pi, \mathcal{A} \models G$, for any ABox $\mathcal{A}$. The *answering problem* for (Π, G)—i.e., checking, given an ABox $\mathcal{A}$, whether $\Pi, \mathcal{A} \models G$—is clearly in P. Answering a datalog query with a *linear* program, whose rules have at most one IDB predicate in the body, can be done in NL. A datalog query is *monadic* if all of its IDB predicates are of arity at most 1.

3 Related Work

We begin by putting our case study problem into the context of more general investigations of (i) boundedness (i.e., equivalence to an FO-query) and linearisability of datalog programs and (ii) the data complexity of answering OMQs with expressive ontologies.

The decision problem whether a given datalog program is bounded (equivalent to an FO-query) has been a hot research topic in database theory since the late 1980s. Thus, it was shown that boundedness is undecidable already for linear datalog programs with binary IDB predicates [34] and single rule programs (aka *sirups*) [26]. On the other hand, deciding boundedness is 2ExpTime-complete for *monadic* datalog programs [12, 6] and PSpace-complete for linear monadic programs [12]; for linear sirups, it is even NP-complete [34].

The last two results are relevant to deciding FO-rewritability of OMQs $(\mathcal{C}ov_A, \boldsymbol{q})$, where $\boldsymbol{q}$ has a single solitary F (see Section 2) and is called a 1-*CQ*. Indeed, suppose that $F(x)$ and $T(y_1), \dots, T(y_n)$ are all the solitary occurrences of F and T in $\boldsymbol{q}$. Let $\Pi_{\boldsymbol{q}}$ be a monadic datalog program with three rules

$$G \leftarrow F(x), \boldsymbol{q}', P(y_1), \dots, P(y_n), \tag{1}$$

$$P(x) \leftarrow T(x), \tag{2}$$

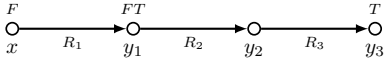
$$P(x) \leftarrow A(x), \boldsymbol{q}', P(y_1), \dots, P(y_n), \tag{3}$$

where $\boldsymbol{q}' = \boldsymbol{q} \setminus \{F(x), T(y_1), \dots, T(y_n)\}$ and P is a fresh predicate symbol that never occurs in our ABoxes. Then, for any ABox $\mathcal{A}$, we have $\mathcal{C}ov_A, \mathcal{A} \models \boldsymbol{q}$ iff $\Pi_{\boldsymbol{q}}, \mathcal{A} \models G$. Thus, FO-rewritability of $(\mathcal{C}ov_A, \boldsymbol{q})$ is clearly related to boundedness of the sirup (3).

The problem of linearising datalog programs, that is, transforming them into equivalent linear datalog programs, which are known to be in NL for data complexity, has also attracted much attention [29, 31, 36, 2] after the Ullman and van Gelder pioneering paper [33]. Here, Example 4 is very instructive: it is easy to construct $\Pi_{\boldsymbol{q}}$ automatically (either directly or by using the *markability* technique from [19] for disjunctive datalog), but clearly some additional artificial intelligence is required to notice the Hint and use it to produce a linear program. In Section 7, we show a datalog program for a similar query, which is produced automatically from an arc consistency procedure; again, this program is not linear but linearisable.

By establishing a remarkable connection to CSPs, it was shown in [8] that deciding FO- and datalog-rewritability of OMQs with a $\mathcal{SHIU}$ ontology is NEXPTIME-complete. This result is obviously applicable to our case study, and we shall discuss it in detail in the next section.

An AC^0/NL/P trichotomy for the data complexity of answering OMQs with an $\mathcal{EL}$ ontology and atomic query, which can be checked in EXPTIME, was established in [23]. This result is applicable to OMQs $(\mathcal{C}ov_A, \boldsymbol{q})$, in which $\boldsymbol{q}$ is an *F-tree* having a single solitary $F(x)$ such that the binary atoms in $\boldsymbol{q}$ form a ditree with root x. Indeed, denote by $\mathcal{T}_{\boldsymbol{Q}}$ the $\mathcal{EL}$ TBox with concept inclusions $F \sqcap C_{\boldsymbol{q}} \sqsubseteq G'$, $T \sqsubseteq P$ and $A \sqcap C_{\boldsymbol{q}} \sqsubseteq P$, where $C_{\boldsymbol{q}}$ is an $\mathcal{EL}$-concept representing $\boldsymbol{q} \setminus \{F(x)\}$ with P for T (so for $\boldsymbol{q}$ of the form



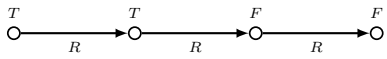
$C_{\boldsymbol{q}} = \exists R_1.(F \sqcap P \sqcap \exists R_2.\exists R_3.P)$). Then, for any ABox $\mathcal{A}$ that does not contain G', we have $\Pi_{\boldsymbol{Q}}, \mathcal{A} \models G$ iff $\mathcal{T}_{\boldsymbol{Q}}, \mathcal{A} \models \exists x\, G'(x)$.

Yet, despite all of these efforts and results (implying, in view of the recent positive solution to the Feder-Vardi conjecture [9, 37], that there is a P/CONP dichotomy for OMQs with $\mathcal{SHIU}$ ontologies, which is decidable in NEXPTIME), we are still lacking simple and transparent, in particular syntactic, conditions guaranteeing this or that data complexity or type of rewritability. Some results in this direction were obtained in [17, 19]. That a transparent classification of monadic sirups according to their data complexity has not been found so far and the close connection to CSPs indicate that this problem is extremely hard in general.

In the next section, we illustrate how OMQs of the form $(\mathcal{C}ov^{\perp}_{\top}, \boldsymbol{q})$ with a path CQ $\boldsymbol{q}$ can be reduced to CSPs.

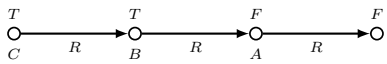
4 Converting Path OMQs to CSPs

In the context of our case study, we are interested in *non-uniform CSPs*. Let $\mathcal{B}$ be a fixed relational structure which in this setting is called a *template*. Each template $\mathcal{B}$ gives rise to the decision problem CSP($\mathcal{B}$) which is to decide, given an ABox $\mathcal{A}$, whether there is a homomorphism from $\mathcal{A}$ to $\mathcal{B}$, in which case we write $\mathcal{A} \to \mathcal{B}$. We show, following [8], how given an OMQ $\boldsymbol{Q} = (\mathcal{C}ov^{\perp}_{\top}, \boldsymbol{q})$ with a path CQ $\boldsymbol{q}$, one can construct a template $\mathcal{B}_{\boldsymbol{q}}$ such that, for any data instance $\mathcal{A}$, we have $\mathcal{A} \to \mathcal{B}_{\boldsymbol{q}}$ iff $\mathcal{C}ov^{\perp}_{\top}, \mathcal{A} \not\models \boldsymbol{q}$. We illustrate the construction of $\mathcal{B}_{\boldsymbol{q}}$ using the CQ $\boldsymbol{q}$ below:



The construction generalises to arbitrary path CQs in the obvious way and can be further extended to OMQs with tree-shaped CQs.

First, we assign labels A, B and C to the first three vertices of $\boldsymbol{q}$ in the following way:



Then we construct the following disjunctive datalog program Π such that, for any $\mathcal{A}$,

we have $\mathcal{C}ov^{\perp}_{\top}, \mathcal{A} \models \boldsymbol{q}$ iff $\mathcal{A}, \Pi \models \exists x\, C(x)$:

$$A(x) \leftarrow F(x), R(x,y), F(y) \tag{4}$$
$$B(x) \leftarrow T(x), R(x,y), F(y), A(y) \tag{5}$$
$$C(x) \leftarrow T(x), R(x,y), T(y), B(y) \tag{6}$$
$$T(x) \vee F(x) \leftarrow \tag{7}$$
$$\perp \leftarrow T(x), F(x) \tag{8}$$

(Informally, the labels A, B and C are used in Π to detect the query pattern in a data instance step-by-step.)

We now construct the CSP template $\mathcal{B}_{\boldsymbol{q}}$ using unary *types* for Π, which are sets t of unary predicates in Π such that t contains either T or F, but not both of them, and $C \notin t$. Thus in this example the types are 8 specific subsets of $\{A, B, C, T, F\}$ that are used as vertex labels in Fig. 1 (talking about types, we often omit curly brackets and commas, so, for example, FAB stands for $\{F, A, B\}$). In general, for a path CQ with n variables there are 2^{n-1} types. Two types t_1 and t_2 are called *R-compatible* if the data instance

$$\{R(t_1, t_2)\} \cup \{X(t_1) \mid X \in t_1\} \cup \{Y(t_2) \mid Y \in t_2\}$$

is a model of Π. The domain of the template $\mathcal{B}_{\boldsymbol{q}}$ consists of the types for Π, unary relations are interpreted in the natural way by $T^{\mathcal{B}_{\boldsymbol{q}}} = \{t \mid T \in t\}$, $F^{\mathcal{B}_{\boldsymbol{q}}} = \{t \mid F \in t\}$, and the binary relation is specified by $R^{\mathcal{B}_{\boldsymbol{q}}} = \{(t_1, t_2) \mid t_1 \text{ and } t_2 \text{ are } R\text{-compatible}\}$.

For the $\boldsymbol{q}$ above we obtain the template $\mathcal{B}_{\boldsymbol{q}}$ shown on the left-hand side of Fig. 1, where the vertices are labelled by the corresponding types. For example, there are no edges between F and FB since the data instances $\{F(x), R(x,y), F(y), B(y)\}$ and $\{F(x), B(x), R(x,y), F(y)\}$ do not satisfy the rule $A(x) \leftarrow F(x), R(x,y), F(y)$, but there is an edge from FA to F since the data instance $\{F(x), A(x), R(x,y), F(y)\}$ is a model of Π. One can see that $\mathcal{A}, \Pi \not\models \exists x\, C(x)$ iff $\mathcal{A} \to \mathcal{B}_{\boldsymbol{q}}$, and so $\mathcal{B}_{\boldsymbol{q}}$ is as required.

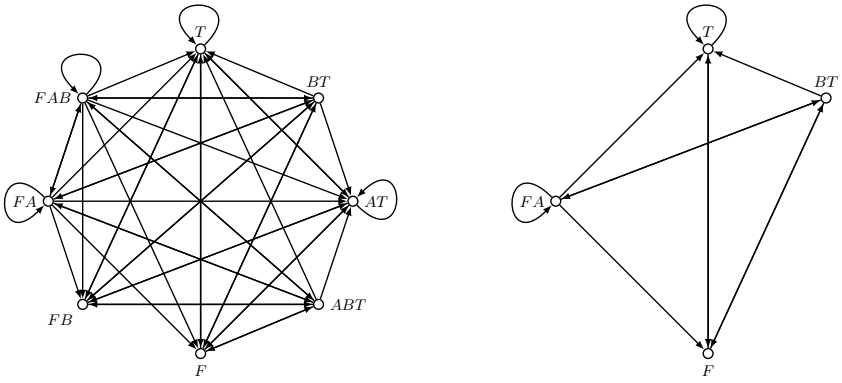


Fig. 1. CSP template $\mathcal{B}_{\boldsymbol{q}}$ (left) and its core (right).

5 Enter Polyanna

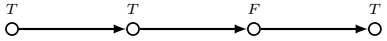
To check whether CSP($\mathcal{B}_{\boldsymbol{q}}$) is in P or coNP-hard, one can use the program Polyanna [13]. Polyanna proceeds in two stages. First, it finds a core of the template $\mathcal{B}_{\boldsymbol{q}}$. We remind the reader that a relational structure $\mathcal{D}'$ is a *core* of a relational structure $\mathcal{D}$ if $\mathcal{D}'$ is a minimal substructure of $\mathcal{D}$ that is homomorphically equivalent to $\mathcal{D}$ (in the sense that $\mathcal{D}' \to \mathcal{D}$ and $\mathcal{D} \to \mathcal{D}'$). The process of constructing such a core by Polyanna is called 'squashing'. For example, it squashes the 8-vertex template $\mathcal{B}_{\boldsymbol{q}}$ in Fig. 1 into the core template with 4 vertices shown on the right-hand side of the figure.

Then Polyanna decides tractability or coNP-hardness of CSP($\mathcal{B}_{\boldsymbol{q}}$) by checking whether the core template has polymorphisms of certain types by constructing and solving the corresponding 'indicator problems' [18]. While doing this, Polyanna uses different decomposition techniques to reduce computation in the case when the indicator problem has symmetries. The indicator problem for polymorphisms of arity k and cores with d vertices, for a signature Γ, has $k \cdot d^k$ variables and $\Sigma_{R \in \Gamma} |R|^k$ constraints. Given a core template of size d, Polyanna considers polymorphisms of arity up to $\max(3, d)$. In practice, for our use case, this implies that it can handle cores of size up to 4, but runs out of memory for some cores of size ≥ 5.

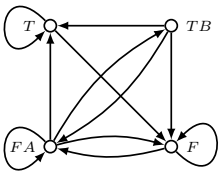
It has recently been shown that tractability of CSP can be determined by considering special polymorphisms of arity 4 that satisfy the identity $f(y, x, y, z) = f(x, y, z, x)$; they are called *Siggers polymorphisms* in [5]. However, we could not find any publicly available implementations for these polymorphisms, and so Polyanna can be considered as top-edge technology even if it is more than 15 years old. We conjecture that the range of its applicability can be significantly extended by enhancing it with the ability to search for Siggers polymorphisms.

We used Polyanna to classify the data complexity of OMQs $\boldsymbol{Q} = (\mathcal{C}ov^{\perp}_{\top}, \boldsymbol{q})$ with path CQs $\boldsymbol{q}$ of length up to 4 (that is, with at most 4 variables).[5]

She correctly determined that all of them but four OMQs with two T-nodes and two F-nodes are in P. For example, the OMQ with the CQ



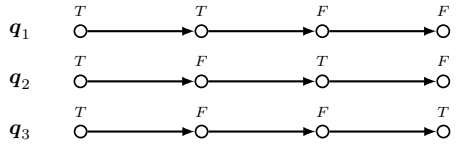
and the core CSP template $\mathcal{B}$ shown below was classified as tractable.



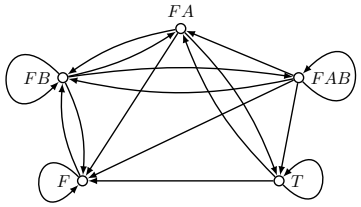
In Section 7, we illustrate how Polyanna's output can be used to construct a monadic datalog-rewritings of OMQs.

Polyanna also managed to determine that the CQs $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ below give coNP-hard OMQs:

[5] Path CQs of length 5 produce templates of size 16, and it takes over an hour to detect its core.



In fact, the CSP templates for $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ have a 4-vertex core, while CSP($\mathcal{B}_{\boldsymbol{q}_3}$) has the following core with five vertices:

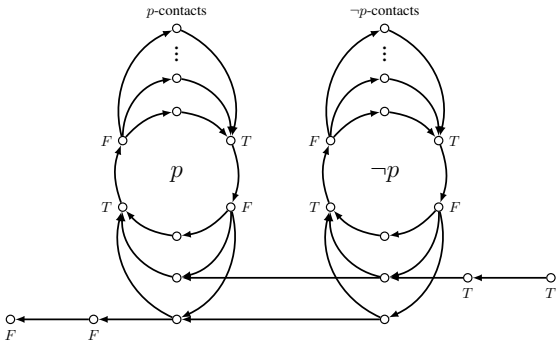


which turned out to be too hard for Polyanna.

In the next section, we sketch direct proofs of coNP-hardness of answering the three OMQs $\boldsymbol{Q}_i = (\mathcal{C}ov^{\perp}_{\top}, \boldsymbol{q}_i)$, for $i = 1, 2, 3$, by reduction of 3SAT. The conaisseurs might find it instructive to compare the gadgets used in those proofs with the general machinery of pp-interpretations coming from universal algebra.

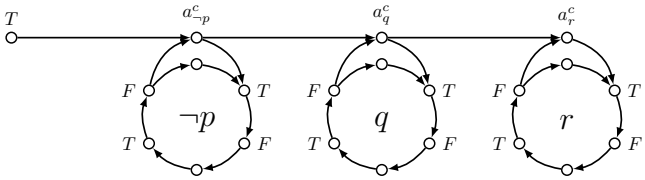
6 Three coNP-hard OMQs

Consider first the OMQ $\boldsymbol{Q}_1 = (\mathcal{C}ov^{\perp}_{\top}, \boldsymbol{q}_1)$. For every propositional variable p in a given 3CNF ψ, we construct a 'gadget' shown in the picture below, where the number of vertices above each of the circles matches the number of clauses in ψ; we refer to these vertices as *p-contacts* and, respectively, *¬p-contacts*:



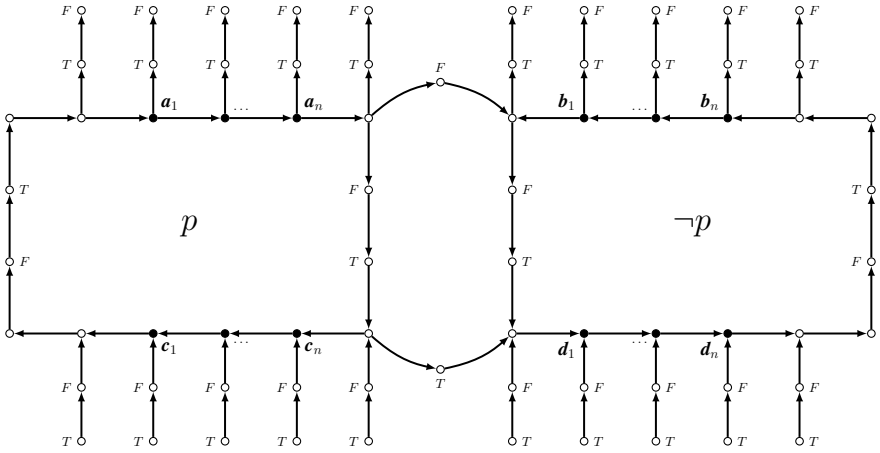
Observe that, for any model $\mathcal{I}$ of $\mathcal{C}ov^{\perp}_{\top}$ based on the constructed gadget for p, if $\mathcal{I} \not\models \boldsymbol{q}_1$ then either (i) the p-contacts are all in $F^{\mathcal{I}}$ and the $\neg p$-contacts are all in $T^{\mathcal{I}}$, or (ii) the p-contacts are all in $T^{\mathcal{I}}$ and the $\neg p$-contacts are all in $F^{\mathcal{I}}$.

Now, for every clause $c = (l_1 \vee l_2 \vee l_3)$ in ψ with literals l_i, we add to the constructed gadgets the atoms $T(c)$, $R(c, a^c_{\neg l_1})$, $R(a^c_{\neg l_1}, a^c_{l_2})$, $R(a^c_{l_2}, a^c_{l_3})$, where c is a new individual, $a^c_{\neg l_1}$ a fresh $\neg l_1$-contact, $a^c_{l_2}$ a fresh l_2-contact, and $a^c_{l_3}$ a fresh l_3-contact. For example, for the clause $c = (p \vee q \vee r)$, we obtain the fragment below:



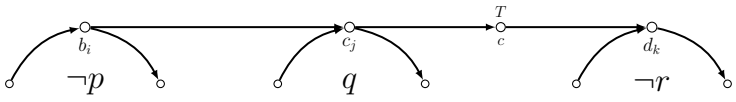
The resulting ABox is denoted by $\mathcal{A}_\psi$. The reader can check that ψ is satisfiable iff $\mathcal{C}ov^{\perp}_{\top}, \mathcal{A}_\psi \not\models \boldsymbol{q}_1$. It follows that answering $\boldsymbol{Q}_1$ is coNP-hard.

Next, consider the OMQ $\boldsymbol{Q}_2 = (\mathcal{C}ov^{\perp}_{\top}, \boldsymbol{q}_2)$. Similarly to the previous case, for every variable p occurring in ψ, we take the following *p-gadget*, where n is the number of clauses in ψ:



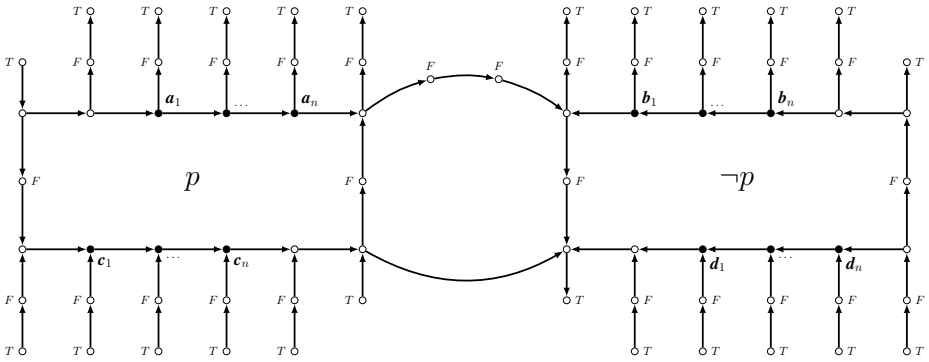
The key property of the p-gadget is that, for any model $\mathcal{I}$ of $\mathcal{C}ov^{\perp}_{\top}$ based on this gadget, if $\mathcal{I} \not\models \boldsymbol{q}_2$ then either the vertices without labels on left-hand side of the gadget are all in $T^{\mathcal{I}}$ and the vertices without labels on the right-hand side are all in $F^{\mathcal{I}}$, or the other way round. We refer to the a_i and b_i as $p^{\uparrow}$- and $\neg p^{\uparrow}$-*contacts*, and to the c_i and d_i as $p^{\downarrow}$- and $\neg p^{\downarrow}$-*contacts*, respectively.

Now, for every clause $c = (l_1 \vee l_2 \vee l_3)$ in ψ, we add to the constructed gadgets for the variables in ψ the atoms $R(u^c_{\neg l_1}, v^c_{l_2})$, $R(v^c_{l_2}, c)$, $T(c)$, $R(c, w^c_{l_3})$, where c is a new individual, $u^c_{\neg l_1}$ a fresh $\neg l_1^{\uparrow}$-contact, $v^c_{l_2}$ a fresh $l_2^{\downarrow}$-contact, and $w^c_{l_3}$ a fresh $l_3^{\downarrow}$-contact. For example, for the clause $c = (p \vee q \vee \neg r)$, we obtain the fragment below:



The resulting ABox $\mathcal{A}_\psi$ is such that ψ is satisfiable iff $\mathcal{C}ov^{\perp}_{\top}, \mathcal{A}_\psi \not\models \boldsymbol{q}_2$.

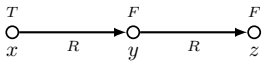
Finally, for the OMQ $\boldsymbol{Q}_3 = (\mathcal{C}ov^{\perp}_{\top}, \boldsymbol{q}_3)$, we use the following p-gadget:



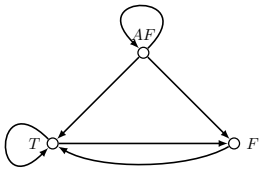
and a similar encoding of clauses. The only difference occurs at the individual c in the clause gadget where we place an F-atom instead of a T-atom.

7 Monadic Datalog Rewritings based on Arc Consistency

For tractable OMQs, Polyanna outputs semilattice functions, which guarantee that the classical arc-consistency check gives correct answers to these OMQs; see [21] for a survey of different arc-consistency routines. The classical arc-consistency procedure can be interpreted as a monadic datalog program, which gives a monadic datalog-rewriting of the OMQ in question. It may be instructive to have a look at the monadic datalog program for the CQ



which is dual to the one in Example 4. Below is a 3-vertex core of the template for this query:



The idea of arc-consistency is to introduce an IDB predicate P for each subset of vertices in the template, with the intuitive meaning that P should hold of a constant c in a given data instance iff it follows that c must be mapped by a homomorphism within the set corresponding to P. Using semilattice function, one can prove that the CSP does not have a solution iff we can deduce the predicate that corresponds to the empty set.

To make our monadic datalog program more readable, we associate predicate names P, Q, T', F', F_0 etc. to sets of types according to the table below. The table also shows the R-image of any set X (the set of all vertices that are R-accessible from X) and its R-pre-image (the set of all vertices from which X is R-accessible) encoded as predicate

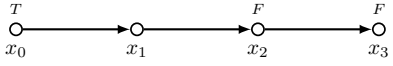
names:

predicate	subset of vertices	R-image	R-pre-image
P	$\{T, F\}$	P	1
T'	$\{T\}$	P	1
F'	$\{AF, F\}$	1	Q
Q	$\{T, AF\}$	1	1
F_0	$\{F\}$	T	Q
F_1	$\{AF\}$	1	F_1
1	$\{AF, A, T\}$	1	1

Then we have the following program for arc-consistency. The first two rules say that T and F should be preserved. The next rules say 'if X is the R-pre-image of Y, then we should have the rule $X(x) \leftarrow R(x, y), Y(y)$' (second group), 'if Z is the R-image of Y, then we should have the rule $X(z) \leftarrow Y(y), R(y, z)$ (third group). We also have rules for Boolean reasoning: if $X \subseteq Y$, then we add $Y(z) \leftarrow X(z)$; and if $Z = X \cap Y$, we have the rule $Z(z) \leftarrow X(z), Y(z)$. Finally, we have rules with the goal predicate in the head for disjoint X and Y. Here goes the program:

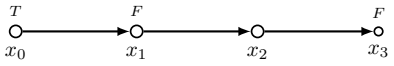
$$T'(x) \leftarrow T(x) \tag{9}$$
$$F'(x) \leftarrow F(x) \tag{10}$$
$$P(y) \leftarrow T'(x), R(x, y) \tag{11}$$
$$Q(x) \leftarrow R(x, y), F_0(y) \tag{12}$$
$$F_1(x) \leftarrow R(x, y), F_1(y) \tag{13}$$
$$P(y) \leftarrow P(x), R(x, y) \tag{14}$$
$$Q(x) \leftarrow R(x, y), F'(y) \tag{15}$$
$$F'(x) \leftarrow F_1(x) \tag{16}$$
$$F'(x) \leftarrow F_0(x) \tag{17}$$
$$Q(x) \leftarrow F_1(x) \tag{18}$$
$$P(x) \leftarrow F_0(x) \tag{19}$$
$$Q(x) \leftarrow T'(x) \tag{20}$$
$$P(x) \leftarrow T'(x) \tag{21}$$
$$T'(x) \leftarrow Q(x), P(x) \tag{22}$$
$$F_0(x) \leftarrow P(x), F'(x) \tag{23}$$
$$F_1(x) \leftarrow Q(x), F'(x) \tag{24}$$
$$G \leftarrow F_1(x), P(x) \tag{25}$$
$$G \leftarrow T'(x), F'(x) \tag{26}$$
$$G \leftarrow Q(x), F_0(x) \tag{27}$$

To illustrate, consider the following data instance



Then rule (9) produces $T'(x_0)$, rule (11) produces $P(x_1)$, and rule (14) produces $P(x_2)$ and $P(x_3)$. On the other hand, rule (10) produces $F'(x_3)$ and $F'(x_2)$, and so rule (23) produces $F_0(x_2)$ and $F_0(x_3)$. Now rule (12) produces $Q(x_2)$, which gives us G by rule (27).

However, the program returns 'no' on the following data instance (it still produces $P(x_3)$ and $Q(x_2)$, but now instead of G it produces $T'(x_2)$):



8 Appendix

We conclude this article with a brief appendix that summarises what is known about the data complexity of answering the OMQs in the framework of our case study. For more details and proofs the reader is referred to [15, 14, 16].

We classify the OMQs according to the number of occurrences of solitary F in their CQs (the case of solitary T is symmetric).

8.1 0-CQs

By a 0-*CQ* we mean any CQ that does not contain a solitary F. A *twin* in a CQ $\boldsymbol{q}$ is any pair $F(x), T(x) \in \boldsymbol{q}$. Here is an encouraging syntactic criterion of FO-rewritability for OMQs of the form $(\mathcal{C}ov_A^{\bot}, \boldsymbol{q})$:

Theorem 1. (i) *If $\boldsymbol{q}$ is a 0-CQ, then answering both $(\mathcal{C}ov_A^{\bot}, \boldsymbol{q})$ and $(\mathcal{C}ov_A, \boldsymbol{q})$ is in* AC^0*, with $\boldsymbol{q}$ being an FO-rewriting of these OMQs.*

(ii) *If $\boldsymbol{q}$ is not a 0-CQ and does not contain twins, then answering both $(\mathcal{C}ov_{\top}^{\bot}, \boldsymbol{q})$ and $(\mathcal{C}ov_{\top}, \boldsymbol{q})$ is* L*-hard.*

Corollary 1. *An OMQ $(\mathcal{C}ov_A^{\bot}, \boldsymbol{q})$ is in* AC^0 *iff $\boldsymbol{q}$ is a 0-CQ, which can be decided in linear time.*

Theorem 1 (i) generalises to OMQs with ontologies $\mathcal{C}ov_n = \{A \sqsubseteq B_1 \sqcup \dots \sqcup B_n\}$, for $n \geq 2$:

Theorem 2. *Suppose $\boldsymbol{q}$ is any CQ that does not contain an occurrence of B_i, for some i $(1 \leq i \leq n)$. Then answering the OMQ $\boldsymbol{Q} = (\mathcal{C}ov_n, \boldsymbol{q})$ is in* AC^0.

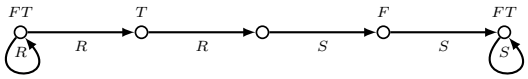
Thus, only those CQs can 'feel' $\mathcal{C}ov_n$ as far as FO-rewritability is concerned that contain all the B_n (which makes them quite complex in practice).

If twins can occur in CQs (that is, F and T are not necessarily disjoint), the picture becomes more complex. On the one hand, we have the following criterion for OMQs $(\mathcal{C}ov_A, \boldsymbol{q})$ with a *path CQ* $\boldsymbol{q}$ whose variables $x_0, \dots, x_n$ in $\boldsymbol{q}$ are ordered so that the binary atoms in $\boldsymbol{q}$ form a chain $R_1(x_0, x_1), \dots, R_n(x_{n-1}, x_n)$.

Theorem 3. *An OMQ $(\mathcal{C}ov_A, \boldsymbol{q})$ with a path CQ $\boldsymbol{q}$ is in* AC^0 *iff $\boldsymbol{q}$ is a 0-CQ. If $\boldsymbol{q}$ contains both solitary F and T, then $(\mathcal{C}ov_A, \boldsymbol{q})$ is* NL*-hard.*

On the other hand, this AC^0/NL criterion collapses for path CQs with loops:

Proposition 1. *The OMQ $(\mathcal{C}ov_A, \boldsymbol{q})$, where $\boldsymbol{q}$ is shown below, is in* AC^0.



Note that the CQ $\boldsymbol{q}$ above is *minimal* (not equivalent to any of its proper sub-CQs). Note also that, if a minimal 1-CQ $\boldsymbol{q}$ contains both a solitary F and a solitary T, then FO-rewritability of $(\mathcal{C}ov_A, \boldsymbol{q})$ implies that $\boldsymbol{q}$ contains at least one twin (FT) and at least one y with $T(y) \notin \boldsymbol{q}$ and $F(y) \notin \boldsymbol{q}$ (which can be shown using Theorem 7 (i)).

8.2 1-CQs

1-CQs have exactly one solitary F. We now complement the sufficient conditions of L- and NL-hardness from Section 8.1 with sufficient conditions of OMQ answering in L- and NL.

A *1-CQ* $\boldsymbol{q}'(x, y)$ is *symmetric* if the CQs $\boldsymbol{q}'(x, y)$ and $\boldsymbol{q}'(y, x)$ are equivalent in the sense that $\boldsymbol{q}'(a, b)$ holds in $\mathcal{A}$ iff $\boldsymbol{q}'(b, a)$ holds in $\mathcal{A}$, for any ABox $\mathcal{A}$ and $a, b \in \mathsf{ind}(\mathcal{A})$.

Theorem 4. *Let $\boldsymbol{Q} = (\mathcal{C}ov_A, \boldsymbol{q})$ be any OMQ such that*

$$\boldsymbol{q} = \exists x, y\, (F(x) \land \boldsymbol{q}'_1(x) \land \boldsymbol{q}'(x, y) \land \boldsymbol{q}'_2(y) \land T(y)),$$

for some connected CQs $\boldsymbol{q}'(x, y)$, $\boldsymbol{q}'_1(x)$ and $\boldsymbol{q}'_2(y)$ that do not contain solitary T and F, and $\boldsymbol{q}'(x, y)$ is symmetric. Then answering $\boldsymbol{Q}$ can be done in L.

If we do not require $\boldsymbol{q}'(x, y)$ to be symmetric, the complexity upper bound increases to NL:

Theorem 5. *Let $\boldsymbol{Q} = (\mathcal{C}ov_A, \boldsymbol{q})$ be any OMQ such that*

$$\boldsymbol{q} = \exists x, y\, (F(x) \land T(y) \land \boldsymbol{q}'(x, y)),$$

for some connected CQ $\boldsymbol{q}'(x, y)$ without solitary occurrences of F and T. Then answering $\boldsymbol{Q}$ can be done in NL.

By an F-tree CQ we mean a CQ $\boldsymbol{q}$ having a single solitary $F(x)$ such that the binary atoms in $\boldsymbol{q}$ form a ditree with root x.

Theorem 6. (i) *Answering any OMQ $(\mathcal{C}ov_A, \boldsymbol{q})$ with a 1-CQ $\boldsymbol{q}$ can be done in* P.

(ii) *Answering any OMQ $(\mathcal{C}ov_A, \boldsymbol{q})$ with an F-tree $\boldsymbol{q}$ is either in* AC^0 *or* NL*-complete or* P*-complete. The trichotomy can be decided in* ExpTime.

Theorem 6 (ii) was proved by a reduction to the AC^0/NL/P-trichotomy of [23]. It is to be noted, however, that applying the algorithm from [23] in our case is tricky because the input ontology must first be converted to a normal form. As a result, we do not obtain transparent syntactic criteria on the shape of $\boldsymbol{q}$ that would guarantee that the OMQ $(\mathcal{C}ov_A, \boldsymbol{q})$ belongs to the desired complexity class.

We now give a semantic sufficient condition for an OMQ with a 1-CQ to lie in NL. This condition uses ideas and constructions from [12, 23]. Let $\boldsymbol{Q} = (\mathcal{C}ov_A, \boldsymbol{q})$ be an OMQ with a 1-CQ $\boldsymbol{q}$ having a solitary $F(x)$. Define by induction a class $\mathfrak{K}_{\boldsymbol{Q}}$ of ABoxes that will be called *cactuses for* $\boldsymbol{Q}$. We start by setting $\mathfrak{K}_{\boldsymbol{Q}} := \{\boldsymbol{q}\}$, regarding $\boldsymbol{q}$ as an ABox, and then recursively apply to $\mathfrak{K}_{\boldsymbol{Q}}$ the following two rules:

(bud) if $T(y) \in \mathcal{A} \in \mathfrak{K}_{\boldsymbol{Q}}$ with solitary $T(y)$, then we add to $\mathfrak{K}_{\boldsymbol{Q}}$ the ABox obtained by replacing $T(y)$ in $\mathcal{A}$ with $(\boldsymbol{q} \setminus F(x)) \cup \{A(x)\}$, in which x is renamed to y and all of the other variables are given *fresh* names;

(prune) if $\mathcal{C}ov_A, \mathcal{A}' \models \boldsymbol{Q}$, where $\mathcal{A}' = \mathcal{A} \setminus \{T(y)\}$ and $T(y)$ is solitary, we add to $\mathfrak{K}_{\boldsymbol{Q}}$ the ABox obtained by removing $T(y)$ from $\mathcal{A} \in \mathfrak{K}_{\boldsymbol{Q}}$.

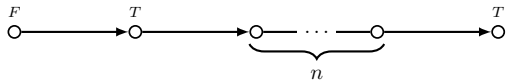
It is readily seen that, for any ABox $\mathcal{A}'$, we have $\mathcal{C}ov_A, \mathcal{A}' \models \boldsymbol{Q}$ iff there exist $\mathcal{A} \in \mathfrak{K}_{\boldsymbol{Q}}$ and a homomorphism $h\colon \mathcal{A} \to \mathcal{A}'$. Denote by $\mathfrak{K}^{\dagger}_{\boldsymbol{Q}}$ the set of minimal cactuses in $\mathfrak{K}_{\boldsymbol{Q}}$ (that have no proper sub-cactuses in $\mathfrak{K}_{\boldsymbol{Q}}$).

For a cactus $\mathcal{C} \in \mathfrak{K}_{\boldsymbol{Q}}$, we refer to the copies of (maximal subsets of) $\boldsymbol{q}$ that comprise $\mathcal{C}$ as *segments*. The *skeleton* $\mathcal{C}^s$ of $\mathcal{C}$ is the ditree whose nodes are the segments $\mathfrak{s}$ of $\mathcal{C}$ and edges $(\mathfrak{s}, \mathfrak{s}')$ mean that $\mathfrak{s}'$ was attached to $\mathfrak{s}$ by budding. The atoms $T(y) \in \mathfrak{s}$ are called the *buds* of $\mathfrak{s}$. The *rank* $\boldsymbol{r}(\mathfrak{s})$ of $\mathfrak{s}$ is defined by induction: if $\mathfrak{s}$ is a leaf, then $\boldsymbol{r}(\mathfrak{s}) = 0$; for non-leaf $\mathfrak{s}$, we compute the maximal rank m of its children and then set

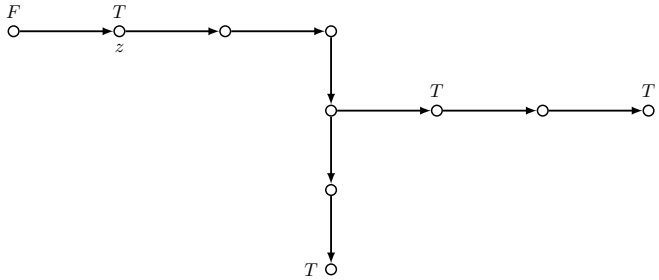
$$\boldsymbol{r}(\mathfrak{s}) = \begin{cases} m+1, & \text{if } \mathfrak{s} \text{ has} \geq 2 \text{ children of rank } m; \\ m, & \text{otherwise.} \end{cases}$$

The *width* of $\mathcal{C}$ and $\mathcal{C}^s$ is the rank of the root in $\mathcal{C}^s$. We say that $\mathfrak{K}^{\dagger}_{\boldsymbol{Q}}$ is of *width* k if it contains a cactus of width k but no cactus of greater width. The *depth* of $\mathcal{C}$ and $\mathcal{C}^s$ is the number of edges in the longest branch in $\mathcal{C}^s$.

We illustrate the definition by an example. Denote by $\boldsymbol{q}_{TnT}$, for $n \geq 0$, the 1-CQ shown below, where all the binary predicates are R and the n variables without labels do not occur in F- or T-atoms:



Example 6. Let $\boldsymbol{Q} = (\mathcal{C}ov_{\top}, \boldsymbol{q}_{T1T})$. In the picture below, we show a cactus $\mathcal{C}$ obtained by applying **(bud)** twice to $\boldsymbol{q}_{T1T}$ (with $A = \top$ omitted):



One can check that $\mathcal{C}ov_{\top}, \mathcal{C} \setminus \{T(z)\} \models \boldsymbol{q}_{T1T}$, and so an application of **(prune)** will remove $T(z)$ from $\mathcal{C}$. Using this observation, one can show that $\mathfrak{K}^{\dagger}_{\boldsymbol{Q}}$ is of width 1. On

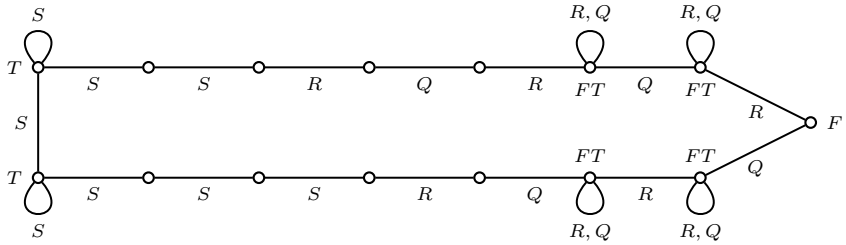
the other hand, if $\boldsymbol{Q} = (\mathcal{C}ov_A, \boldsymbol{q}_{T1T})$ then $\mathfrak{K}^{\dagger}_{\boldsymbol{Q}}$ is of unbounded width as follows from Theorem 9 below.

Theorem 7. *Let $\boldsymbol{Q} = (\mathcal{C}ov_A, \boldsymbol{q})$ be an OMQ with a 1-CQ $\boldsymbol{q}$. Then*

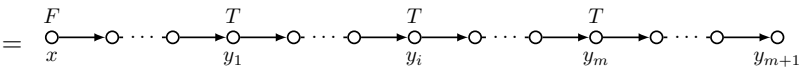
(i) $\boldsymbol{Q}$ is in AC^0 *iff for every $\mathcal{C} \in \mathfrak{K}^{\dagger}_{\boldsymbol{Q}}$, there is a homomorphism $h\colon \boldsymbol{q} \to \mathcal{C}$;*

(ii) $\boldsymbol{Q}$ is rewritable in linear datalog, and so is in NL*, if $\mathfrak{K}^{\dagger}_{\boldsymbol{Q}}$ is of bounded width.*

It is worth noting that, for $\boldsymbol{Q} = (\mathcal{C}ov_\top, \boldsymbol{q})$ with $\boldsymbol{q}$ from Proposition 1, $\mathfrak{K}^{\dagger}_{\boldsymbol{Q}}$ consists of $\boldsymbol{q}$ and the cactus of depth 1, in which the only solitary T is removed by **(prune)**. Clearly, there is a homomorphism from $\boldsymbol{q}$ into this cactus, and so $\boldsymbol{Q}$ is FO-rewritable. However, for the 1-CQ $\boldsymbol{q}$ in the picture below (where all edges are bidirectional), $(\mathcal{C}ov_\top, \boldsymbol{q})$ is not FO-rewritable, but there is a homomorphism from $\boldsymbol{q}$ to both cactuses of depth 1. We do not know whether, in general, there is an upper bound $N_{\boldsymbol{q}}$ such that the existence of homomorphisms $h\colon \boldsymbol{q} \to \mathcal{C}$, for all $\mathcal{C} \in \mathfrak{K}_{\boldsymbol{Q}}$ of depth $N_{\boldsymbol{q}}$, would ensure FO-rewritability of $(\mathcal{C}ov_A, \boldsymbol{q})$. For 1-CQs $\boldsymbol{q}$ with a single solitary T, one can take $N_{\boldsymbol{q}} = |\boldsymbol{q}| + 1$. Neither do we know the exact complexity of deciding FO-rewritability of OMQs with 1-CQs. As mentioned in Section 3, this problem is reducible to the boundedness problem for monadic datalog programs, which is known to be in 2EXPTIME.



Theorem 7 (ii) allows us to obtain a sufficient condition for linear-datalog rewritability of OMQs $(\mathcal{C}ov_A, \boldsymbol{q})$ with an F-path CQ $\boldsymbol{q}$, that is, a path CQ with a single solitary F at its root. We represent such a $\boldsymbol{q}$ as shown in the picture below, which indicates *all* the solitary occurrences of F and T:



We require the following sub-CQs of $\boldsymbol{q}$:

- $\boldsymbol{q}_i$ is the suffix of $\boldsymbol{q}$ that starts at y_i, but without $T(y_i)$, for $1 \leq i \leq m$;
- $\boldsymbol{q}^*_i$ is the prefix of $\boldsymbol{q}$ that ends at y_i, but without $F(x)$ and $T(y_i)$, for $1 \leq i \leq m$;
- $\boldsymbol{q}^*_{m+1}$ is $\boldsymbol{q}$ without $F(x)$,

and write $f_i\colon \boldsymbol{q}_i \twoheadrightarrow \boldsymbol{q}$ if f_i is a homomorphism from $\boldsymbol{q}_i$ into $\boldsymbol{q}$ with $f_i(y_i) = x$.

Theorem 8. *If for each $1 \leq i \leq m$ there exist $f_i\colon \boldsymbol{q}_i \twoheadrightarrow \boldsymbol{q}$, then $(\mathcal{C}ov_A, \boldsymbol{q})$ is rewritable into a linear datalog program, and so is* NL*-complete.*

For F-path CQs $\boldsymbol{q}$ without twins, we extend Theorem 8 to a NL/P dichotomy (provided that NL $\neq$ P). Given such a CQ $\boldsymbol{q}$, we denote by $N_{\boldsymbol{q}}$ the set of the numbers indicating the length of the path from x to each of the y_i, $i = 1, \dots, m+1$.

Theorem 9. *Let $\boldsymbol{Q} = (\mathcal{C}ov_A, \boldsymbol{q})$ be an OMQ where $\boldsymbol{q}$ is an F-path CQ without twins having a single binary relation. The following are equivalent unless* NL = P:

(i) $\boldsymbol{Q}$ *is* NL*-complete*;
(ii) $\{0\} \cup N_{\boldsymbol{q}}$ *is an arithmetic progression*;
(iii) *there exist* $f_i \colon \boldsymbol{q}_i \twoheadrightarrow \boldsymbol{q}$ *for every* $i = 1, \dots, m$.

If these conditions do not hold, then $\boldsymbol{Q}$ is P*-complete.*

Note that the proof of P-hardness in Theorem 9 does not go through for $A = \top$. Thus, for $(\mathcal{C}ov_\top, \boldsymbol{q}_{T1T})$, we are in the framework of Example 6 and, by Theorem 7 (ii), this OMQ is in NL. In fact, we have the following NL/P dichotomy for the OMQs of the form $\boldsymbol{Q} = (\mathcal{C}ov_\top, \boldsymbol{q}_{TnT})$:

- either n is equal to 1, and answering $\boldsymbol{Q}$ is in NL,
- or $n \geq 2$, and answering $\boldsymbol{Q}$ is P-hard.

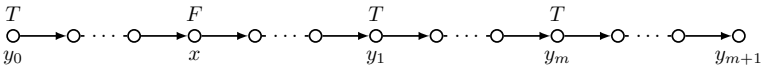
Proposition 2. *Answering the OMQ $(\mathcal{C}ov_\top, \boldsymbol{q}_{T1T})$ is* NL*-complete.*

Theorem 10. *The OMQs $(\mathcal{C}ov_\top, \boldsymbol{q}_{TnT})$ (and $(\mathcal{C}ov_A, \boldsymbol{q}_{TnT})$), for $n \geq 2$, are* P*-complete.*

On the other hand we have:

Proposition 3. *Answering the OMQ $(\mathcal{C}ov_A, \boldsymbol{q}_{T1T})$ is* P*-complete.*

We now apply Theorem 7 (ii) to the class of TF*-path CQs* of the form

$\boldsymbol{q}_{TF} =$ 

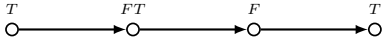
where the $T(y_i)$ and $F(x)$ are all the solitary occurrences of T and F in $\boldsymbol{q}_{TF}$. We represent this CQ as

$$\boldsymbol{q}_{TF} = \{T(y_0)\} \cup \boldsymbol{q}_0 \cup \boldsymbol{q},$$

where $\boldsymbol{q}_0$ is the sub-CQ of $\boldsymbol{q}_{TF}$ between y_0 and x with $T(y_0)$ removed and $\boldsymbol{q}$ is the same as in Theorem 8 (and $\boldsymbol{q}^*_{m+1}$ is $\boldsymbol{q}$ without $F(x)$).

Theorem 11. *If $\boldsymbol{q}$ satisfies the condition of Theorem 8 and there is a homomorphism $h \colon \boldsymbol{q}^*_{m+1} \to \boldsymbol{q}_0$ such that $h(x) = y_0$, then answering $(\mathcal{C}ov_A, \boldsymbol{q}_{TF})$ is* NL*-complete.*

For example, the OMQ $(\mathcal{C}ov_A, \boldsymbol{q})$ with $\boldsymbol{q}$ shown below is NL-complete:



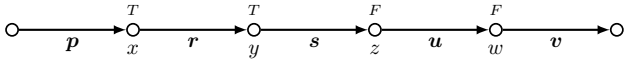
On the other hand, we have the following:

Proposition 4. *Answering the OMQs $(\mathcal{C}ov_A, \boldsymbol{q})$ and $(\mathcal{C}ov_\top, \boldsymbol{q})$ is* P*-complete for $\boldsymbol{q}$ of the forms*

8.3 2-CQs

A *2-CQ* has at least two solitary F and at least two solitary T. We have the following generalisations of the CONP-hardness results from Section 6 for the the OMQs $\boldsymbol{Q}_j = (\mathcal{C}ov^{\perp}_{\top}, \boldsymbol{q}_j)$, for $j \in \{1, 2, 3\}$:

Consider the *2-2-CQs*, which are path 2-CQs where all the F are located after all the T, and every occurrence of T or F is solitary. We represent any given 2-2-CQ $\boldsymbol{q}$ as shown below



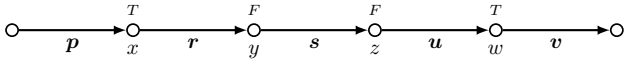
where $\boldsymbol{p}$, $\boldsymbol{r}$, $\boldsymbol{u}$ and $\boldsymbol{v}$ do not contain F and T, while $\boldsymbol{s}$ may contain solitary occurrences of both T and F (in other words, the T shown in the picture are the first two occurrences of T in $\boldsymbol{q}$ and the F are the last two occurrences of F in $\boldsymbol{q}$). Denote by $\boldsymbol{q_r}$ the suffix of $\boldsymbol{q}$ that starts from x but without $T(x)$; similarly, $\boldsymbol{q_u}$ is the suffix of $\boldsymbol{q}$ starting from z but without $F(z)$. Denote by $\boldsymbol{q}^-_{\boldsymbol{r}}$ the prefix of $\boldsymbol{q}$ that ends at y but without $T(y)$; similarly, $\boldsymbol{q}^-_{\boldsymbol{u}}$ is the prefix of $\boldsymbol{q}$ ending at w but without $F(w)$. Using the construction from [15], one can show the following:

Theorem 12. *Any OMQ $(\mathcal{C}ov_A, \boldsymbol{q})$ with a 2-2-CQ $\boldsymbol{q}$ is* CONP*-complete provided the following conditions are satisfied: (i) there is no homomorphism $h_1 \colon \boldsymbol{q_u} \to \boldsymbol{q_r}$ with $h_1(z) = x$, and (ii) there is no homomorphism $h_2 \colon \boldsymbol{q}^-_{\boldsymbol{r}} \to \boldsymbol{q}^-_{\boldsymbol{u}}$ with $h_2(y) = w$.*

We do not know yet whether this theorem holds for $\mathcal{C}ov_{\top}$ in place of $\mathcal{C}ov_A$.

In Theorem 13 and 14, we assume that $\boldsymbol{p}$ and $\boldsymbol{v}$ do not contain F and T, while $\boldsymbol{r}$ and $\boldsymbol{u}$ may only contain solitary occurrences of T ($F \notin \boldsymbol{r}, \boldsymbol{u}$), and $\boldsymbol{s}$ only solitary occurrences of F ($T \notin \boldsymbol{s}$).

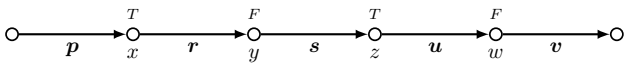
Theorem 13. *Any OMQ $(\mathcal{C}ov_A, \boldsymbol{q})$ with $\boldsymbol{q}$ of the form*



is CONP*-complete provided the following conditions are satisfied: (i) there is no homomorphism $h_1 \colon \boldsymbol{r_t} \to \boldsymbol{u}$ with $h_1(y) = w$, and (ii) there is no homomorphism $h_2 \colon \boldsymbol{u_t} \to \boldsymbol{r}$ with $h_2(z) = x$, where $\boldsymbol{r}$ is the sub-CQ of $\boldsymbol{q}$ between x and y without $T(x)$, $F(y)$, and similarly for $\boldsymbol{u}$, $\boldsymbol{r_t}$ is $\boldsymbol{r}$ with $T(x)$ and $\boldsymbol{u_t}$ is $\boldsymbol{u}$ with $T(w)$.*

In Theorem 14, we use $\boldsymbol{r}^{ext} = \boldsymbol{r}(x, y) \land T(y) \land \boldsymbol{s}_1(y, y_1) \land F(y_1)$, where $\boldsymbol{s_1}$ is the part of $\boldsymbol{s}$ such that $\boldsymbol{s}(y, z) = \boldsymbol{s}_1(y, y_1) \land F(y_1) \land \boldsymbol{s}_2(y_1, z)$ and $\boldsymbol{s}_1(y, y_1)$ does not contain any occurrences of F. In other words, the variable y_1 corresponds to the first appearance of F in $\boldsymbol{s}$, where $\boldsymbol{s}$ is the sub-CQ of $\boldsymbol{q}$ between y and z without $F(y), T(z)$.

Theorem 14. *Any OMQ $(\mathcal{C}ov_A, \boldsymbol{q})$ with $\boldsymbol{q}$ of the form*



is CONP*-complete provided the following conditions hold: (i) there is no homomorphism $g_1 \colon \boldsymbol{r_t} \to \boldsymbol{u}$ with $g_1(y) = w$, and (ii) there is no homomorphism $g_2 \colon \boldsymbol{u} \to \boldsymbol{r}^{ext}$ with $g_2(z) = x$ and $g_2(w) = y_1$.*

Acknowledgements. The work of O. Gerasimova and M. Zakharyaschev was carried out at the National Research University Higher School of Economics and supported by the Russian Science Foundation under grant 17-11-01294. We are grateful to Peter Jeavons and Standa Živný for helpful discussions of Polyanna and arc consistency.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley (1995)
2. Afrati, F.N., Gergatsoulis, M., Toni, F.: Linearisability on datalog programs. Theor. Comput. Sci. 308(1-3), 199–226 (2003), https://doi.org/10.1016/S0304-3975(02)00730-2
3. Arora, S., Barak, B.: Computational Complexity: A Modern Approach. Cambridge University Press, New York, NY, USA, 1st edn. (2009)
4. Baader, F., Horrocks, I., Lutz, C., Sattler, U.: An Introduction to Description Logic. Cambridge University Press (2017), http://www.cambridge.org/de/academic/subjects/computer-science/knowledge-management-databases-and-data-mining/introduction-description-logic?format=PB\#17zVGeWD2TZUeu6s.97
5. Barto, L., Krokhin, A., Willard, R.: Polymorphisms, and how to use them. In: Dagstuhl Follow-Ups. vol. 7. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2017)
6. Benedikt, M., ten Cate, B., Colcombet, T., Vanden Boom, M.: The complexity of boundedness for guarded logics. In: 30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2015, Kyoto, Japan, July 6-10, 2015. pp. 293–304. IEEE Computer Society (2015), https://doi.org/10.1109/LICS.2015.36
7. Benedikt, M., Grau, B.C., Kostylev, E.V.: Logical foundations of information disclosure in ontology-based data integration. Artif. Intell. 262, 52–95 (2018), https://doi.org/10.1016/j.artint.2018.06.002
8. Bienvenu, M., ten Cate, B., Lutz, C., Wolter, F.: Ontology-based data access: A study through disjunctive datalog, CSP, and MMSNP. ACM Transactions on Database Systems 39(4), 33:1–44 (2014)
9. Bulatov, A.A.: A dichotomy theorem for nonuniform CSPs. In: Umans, C. (ed.) 58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017. pp. 319–330. IEEE Computer Society (2017), https://doi.org/10.1109/FOCS.2017.37
10. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Poggi, A., Rodriguez-Muro, M., Rosati, R., Ruzzi, M., Savo, D.F.: The MASTRO system for ontology-based data access. Semantic Web 2(1), 43–53 (2011)
11. Calvanese, D., et al.: Ontop: Answering SPARQL queries over relational databases 8(3), 471–487 (2017)
12. Cosmadakis, S.S., Gaifman, H., Kanellakis, P.C., Vardi, M.Y.: Decidable optimization problems for database logic programs (preliminary report). In: STOC. pp. 477–490 (1988)
13. Gault, R., Jeavons, P.: Implementing a test for tractability. Constraints 9(2), 139–160 (2004)
14. Gerasimova, O., Kikot, S., Podolskii, V.V., Zakharyaschev, M.: More on the data complexity of answering ontology-mediated queries with a covering axiom. In: Rózewski, P., Lange, C. (eds.) Knowledge Engineering and Semantic Web - 8th International Conference, KESW 2017, Szczecin, Poland, November 8-10, 2017, Proceedings. Communications in Computer and Information Science, vol. 786, pp. 143–158. Springer (2017), https://doi.org/10.1007/978-3-319-69548-8_11

15. Gerasimova, O., Kikot, S., Podolskii, V.V., Zakharyaschev, M.: On the data complexity of ontology-mediated queries with a covering axiom. In: Artale, A., Glimm, B., Kontchakov, R. (eds.) Proceedings of the 30th International Workshop on Description Logics, Montpellier, France, July 18-21, 2017. CEUR Workshop Proceedings, vol. 1879. CEUR-WS.org (2017), http://ceur-ws.org/Vol-1879/paper39.pdf
16. Gerasimova, O., Kikot, S., Zakharyaschev, M.: Towards a data complexity classification of ontology-mediated queries with covering. In: Ortiz, M., Schneider, T. (eds.) Proceedings of the 31st International Workshop on Description Logics co-located with 16th International Conference on Principles of Knowledge Representation and Reasoning (KR 2018), Tempe, Arizona, US, October 27th - to - 29th, 2018. CEUR Workshop Proceedings, vol. 2211. CEUR-WS.org (2018), http://ceur-ws.org/Vol-2211/paper-36.pdf
17. Hernich, A., Lutz, C., Ozaki, A., Wolter, F.: Schema.org as a description logic. In: Calvanese, D., Konev, B. (eds.) Proceedings of the 28th International Workshop on Description Logics, Athens,Greece, June 7-10, 2015. CEUR Workshop Proceedings, vol. 1350. CEUR-WS.org (2015), http://ceur-ws.org/Vol-1350/paper-24.pdf
18. Jeavons, P., Cohen, D., Gyssens, M.: A test for tractability. In: International Conference on Principles and Practice of Constraint Programming. pp. 267–281. Springer (1996)
19. Kaminski, M., Nenov, Y., Grau, B.C.: Datalog rewritability of disjunctive datalog programs and non-Horn ontologies. Artif. Intell. 236, 90–118 (2016), http://dx.doi.org/10.1016/j.artint.2016.03.006
20. Kharlamov, E., Hovland, D., Skjæveland, M.G., Bilidas, D., Jiménez-Ruiz, E., Xiao, G., Soylu, A., Lanti, D., Rezk, M., Zheleznyakov, D., Giese, M., Lie, H., Ioannidis, Y.E., Kotidis, Y., Koubarakis, M., Waaler, A.: Ontology based data access in Statoil. J. Web Sem. 44, 3–36 (2017), https://doi.org/10.1016/j.websem.2017.05.005
21. Kozik, M.: Weak consistency notions for all the csps of bounded width. In: Grohe, M., Koskinen, E., Shankar, N. (eds.) Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016. pp. 633–641. ACM (2016), https://doi.org/10.1145/2933575.2934510
22. Lanti, D., Rezk, M., Xiao, G., Calvanese, D.: The NPD benchmark: Reality check for OBDA systems. In: Alonso, G., Geerts, F., Popa, L., Barceló, P., Teubner, J., Ugarte, M., den Bussche, J.V., Paredaens, J. (eds.) Proceedings of the 18th International Conference on Extending Database Technology, EDBT 2015, Brussels, Belgium, March 23-27, 2015. pp. 617–628. OpenProceedings.org (2015), https://doi.org/10.5441/002/edbt.2015.62
23. Lutz, C., Sabellek, L.: Ontology-mediated querying with the description logic EL: trichotomy and linear datalog rewritability. In: Sierra, C. (ed.) Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017. pp. 1181–1187. ijcai.org (2017), https://doi.org/10.24963/ijcai.2017/164
24. Lutz, C., Wolter, F.: Non-uniform data complexity of query answering in description logics. In: Brewka, G., Eiter, T., McIlraith, S.A. (eds.) Principles of Knowledge Representation and Reasoning: Proceedings of the Thirteenth International Conference, KR 2012, Rome, Italy, June 10-14, 2012. AAAI Press (2012), http://www.aaai.org/ocs/index.php/KR/KR12/paper/view/4533
25. Lutz, C., Wolter, F.: The data complexity of description logic ontologies. Logical Methods in Computer Science 13(4) (2017), https://doi.org/10.23638/LMCS-13(4:7)2017
26. Marcinkowski, J.: DATALOG sirups uniform boundedness is undecidable. In: Proceedings, 11th Annual IEEE Symposium on Logic in Computer Science, New Brunswick, New Jersey, USA, July 27-30, 1996. pp. 13–24. IEEE Computer Society (1996), https://doi.org/10.1109/LICS.1996.561299

27. Papadimitriou, C.: Computational Complexity. Addison-Wesley (1994)
28. Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Linking data to ontologies. Journal on Data Semantics X, 133–173 (2008)
29. Ramakrishnan, R., Sagiv, Y., Ullman, J.D., Vardi, M.Y.: Proof-tree transformation theorems and their applications. In: Proceedings of the eighth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems. pp. 172–181. ACM (1989)
30. Rodriguez-Muro, M., Kontchakov, R., Zakharyaschev, M.: Ontology-based data access: Ontop of databases. In: Alani, H., Kagal, L., Fokoue, A., Groth, P.T., Biemann, C., Parreira, J.X., Aroyo, L., Noy, N.F., Welty, C., Janowicz, K. (eds.) The Semantic Web - ISWC 2013 - 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part I. Lecture Notes in Computer Science, vol. 8218, pp. 558–573. Springer (2013), https://doi.org/10.1007/978-3-642-41335-3_35
31. Saraiya, Y.P.: Linearizing nonlinear recursions in polynomial time. In: Silberschatz, A. (ed.) Proceedings of the Eighth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, March 29-31, 1989, Philadelphia, Pennsylvania, USA. pp. 182–189. ACM Press (1989), http://doi.acm.org/10.1145/73721.73740
32. Schaerf, A.: On the complexity of the instance checking problem in concept languages with existential quantification. J. of Intelligent Information Systems 2, 265–278 (1993)
33. Ullman, J.D., Gelder, A.V.: Parallel complexity of logical query programs. Algorithmica 3, 5–42 (1988), https://doi.org/10.1007/BF01762108
34. Vardi, M.Y.: Decidability and undecidability results for boundedness of linear recursive queries. In: Edmondson-Yurkanan, C., Yannakakis, M. (eds.) Proceedings of the Seventh ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, March 21-23, 1988, Austin, Texas, USA. pp. 341–351. ACM (1988), http://doi.acm.org/10.1145/308386.308470
35. Xiao, G., Calvanese, D., Kontchakov, R., Lembo, D., Poggi, A., Rosati, R., Zakharyaschev, M.: Ontology-based data access: A survey. In: Lang, J. (ed.) Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden. pp. 5511–5519. ijcai.org (2018), https://doi.org/10.24963/ijcai.2018/777
36. Zhang, W., Yu, C.T., Troy, D.: Necessary and sufficient conditions to linearize double recursive programs in logic databases. ACM Trans. Database Syst. 15(3), 459–482 (1990), http://doi.acm.org/10.1145/88636.89237
37. Zhuk, D.: A proof of CSP dichotomy conjecture. In: Umans, C. (ed.) 58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017. pp. 331–342. IEEE Computer Society (2017), https://doi.org/10.1109/FOCS.2017.38