More on the Data Complexity of Answering Ontology-Mediated Queries with a Covering Axiom

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

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

Abstract. We report on our recent results in the ongoing attempts to classify conjunctive queries (CQs) $\boldsymbol{q}$ according to the data complexity of answering ontology-mediated queries of the form $(\{A \sqsubseteq F \sqcup T\}, \boldsymbol{q})$. In particular, we present new families of path CQs for which this problem is NL-, P- or coNP-complete.

1 Introduction

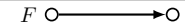
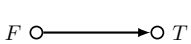
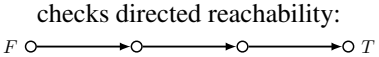
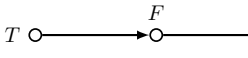
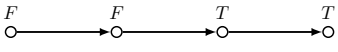
Ontology-based query answering [19, 15, 5, 6] is a way of organising access to data where, instead of the schemas of data sources, the user is provided with an ontology that serves two purposes: (i) it gives a familiar and convenient vocabulary for formulating end-user queries (e.g., standard geological terms for geologists who want to query a company's databases) and (ii) enriches the data with background knowledge. The key notion in this case is *ontology-mediated query* (OMQ), a pair of the form $\boldsymbol{Q} = (\mathcal{T}, \boldsymbol{q}(\boldsymbol{x}))$, where $\mathcal{T}$ is an ontology and $\boldsymbol{q}(\boldsymbol{x})$ a query. The schema of the data is related to the terms in $\mathcal{T}$ by means of mappings, $\mathcal{M}$ (say, in R2RML). Now, given a data instance $\mathcal{A}$, we say that a tuple $\boldsymbol{a}$ of constants from $\mathcal{A}$ is a *certain answer* to $\boldsymbol{Q}$ over $\mathcal{A}$ if $\boldsymbol{q}(\boldsymbol{a})$ holds true in every model of $\mathcal{T}$ and $\mathcal{M}(\mathcal{A})$. Whether finding certain answers to OMQs is feasible in practice depends on the languages of $\mathcal{T}$ and $\boldsymbol{q}$. Thus, if $\mathcal{T}$ is an *OWL 2 QL*[4] ontology and $\boldsymbol{q}$ a conjunctive query (CQ), then answering $\boldsymbol{Q}$ can be done in AC^0 for data complexity; in other words, there is a first-order query $\Phi(\boldsymbol{x})$, called an *FO-rewriting* of $\boldsymbol{Q}$, answers to which over $\mathcal{A}$ are precisely the certain answers to $\boldsymbol{Q}$ over $\mathcal{A}$ [7, 3]. Classifying OMQs according to data complexity has become one of the hottest topics in the area of ontology-based data access [17, 20, 8, 12, 14].

A systematic investigation of this problem was launched in [5], which, in particular, connected it to constraint satisfaction problems. As shown in [13], answering CQs with basic schema.org ontologies and CQs of qvar-size ≤ 2 is in P for combined complexity, where $\boldsymbol{q}$ is of *qvar-size* n if the restriction of $\boldsymbol{q}$ to its quantified variables is a disjoint union of CQs with at most n variables each. Moreover, FO- and datalog-rewritability of OMQs of the form $(\mathcal{T}, \boldsymbol{u})$, where $\mathcal{T}$ is a schema.org ontology and $\boldsymbol{u}$ is a UCQ, are decidable in NExpTime. It has also been recently established in [9] that checking FO-rewritability of OMQs with ontologies formulated in any description logic between

[4] https://www.w3.org/TR/owl2-profiles/

$\mathcal{ALCI}$ and $\mathcal{SHI}$ is 2NEXPTIME-complete. Datalog rewritability of OMQs with ontologies given in disjunctive datalog has been investigated in [14]. An AC^0/NL/P trichotomy of OMQs with $\mathcal{EL}$ ontologies and atomic queries has been established in [18].

In this paper, we report on our ongoing attempts to obtain a complete classification of OMQs of the form $\boldsymbol{Q} = (\mathcal{D}is_A, \boldsymbol{q})$, where $\mathcal{D}is_A = \{A \sqsubseteq F \sqcup T\}$ and $\boldsymbol{q}$ is a CQ. Ontologies with *covering axioms* such as $A \sqsubseteq F \sqcup T$ (saying that, in every model of $\mathcal{D}is_A$, the class A is covered by the union of the classes F and T) are very common in practice: for example, *Animal* $\sqsubseteq$ *Male* $\sqcup$ *Female*. The simple examples collected in the table below show how minor tweaks to $\boldsymbol{q}$ can drastically affect the complexity of $\boldsymbol{Q} = (\mathcal{D}is_A, \boldsymbol{q})$ [10]. In the table and elsewhere in the paper, we represent CQs by diagrams. For example, the first CQ below represents $\exists x, y\, (F(x) \land R(x, y))$ and the second one $\exists x, y\, (F(x) \land R(x, y) \land R(y, x) \land T(y))$.[5] (Binary predicates different from R will be shown in diagrams explicitly.)

Complexity	CQ $\boldsymbol{q}$	Explanation
AC^0		if $\boldsymbol{q}$ has only F but no T, then the F can be ignored
L		checks undirected reachability: the answer to $\boldsymbol{Q}$ is 'yes'
NL		checks directed reachability:  the answer to $\boldsymbol{Q}$ is 'yes'
P		evaluates monotone circuits
coNP		checks CNF satisfiability

The plan of the paper is as follows. Having introduced in Section 2 the basic notions we need in what follows, in Section 3 we use the AC^0/NL/P trichotomy from [18] to establish a similar trichotomy for the OMQs $(\mathcal{D}is_A, \boldsymbol{q})$ whose CQ $\boldsymbol{q}$ is tree-shaped and the only solitary F-atom in it is at the root. In Section 4, we show that the AC^0-criterion for path CQs from [10] collapses for CQs with loops. In Section 5, we present a few classes of path CQs $\boldsymbol{q}$ with a single solitary F, for which answering $(\mathcal{D}is_A, \boldsymbol{q})$ is NL-complete and P-complete. Finally, in Section 6, we give a class of path CQs for which this problem is CONP-complete.

[5] The OMQ $\boldsymbol{Q} = (\mathcal{D}is_\top, \boldsymbol{q})$ with this CQ $\boldsymbol{q}$ can be interpreted as follows, assuming that F stands for 'female', T for 'male', $\top$ for all the individuals of the domain in question, and R for the 'follows' relation: given a graph of Twitter users, in which the gender may be specified for some nodes and missing for the other ones, check whether there certainly exist two people (nodes) in the graph of different gender who follow each other.

2 Preliminaries

By a *conjunctive query* (*CQ*) we mean in this paper any FO-formula $\boldsymbol{q}(\boldsymbol{x}) = \exists \boldsymbol{y}\, \varphi(\boldsymbol{x}, \boldsymbol{y})$, where φ is a conjunction of unary or binary atoms $P(\boldsymbol{z})$ with $\boldsymbol{z} \subseteq \boldsymbol{x} \cup \boldsymbol{y}$. Given a data instance—or an *ABox*, in the description logic parlance—$\mathcal{A}$, we denote by $\mathsf{ind}(\mathcal{A})$ the set of individual names that occur in $\mathcal{A}$. A tuple $\boldsymbol{a} \subseteq \mathsf{ind}(\mathcal{A})$ is a *certain answer* to the OMQ $\boldsymbol{Q} = (\mathcal{D}is_A, \boldsymbol{q}(\boldsymbol{x}))$ over $\mathcal{A}$ if $\mathcal{I} \models \boldsymbol{q}(\boldsymbol{a})$, for every model $\mathcal{I}$ of $\mathcal{D}is_A \cup \mathcal{A}$; in this case we write $\mathcal{D}is_A, \mathcal{A} \models \boldsymbol{q}(\boldsymbol{a})$. If the set $\boldsymbol{x}$ of *answer variables* is empty, a *certain answer* to $\boldsymbol{Q}$ over $\mathcal{D}$ is 'yes' if $\mathcal{I} \models \boldsymbol{q}$, for every model $\mathcal{I}$ of $\mathcal{D}is_A \cup \mathcal{A}$, and 'no' otherwise. OMQs and CQs without answer variables $\boldsymbol{x}$ are called *Boolean*. We often regard CQs as *sets* of their atoms. For the purposes of this paper, it is enough to assume that all CQs $\boldsymbol{q}$ are *Boolean* and *connected* (in the sense that any two distinct variables in $\boldsymbol{q}$ are connected by a not necessarily directed path of binary atoms from $\boldsymbol{q}$).

By *answering* a given OMQ $\boldsymbol{Q} = (\mathcal{D}is_A, \boldsymbol{q}(\boldsymbol{x}))$, we understand the problem of checking, given an ABox $\mathcal{A}$ and a tuple $\boldsymbol{a} \subseteq \mathsf{ind}(\mathcal{A})$, whether $\mathcal{D}is_A, \mathcal{A} \models \boldsymbol{q}(\boldsymbol{a})$. It is easy to see that this problem is always in coNP. It is in the complexity class AC^0 if there is an FO-formula $\boldsymbol{q}'(\boldsymbol{x})$, called an *FO-rewriting* of $\boldsymbol{Q}$, such that $\mathcal{D}is_A, \mathcal{A} \models \boldsymbol{q}(\boldsymbol{a})$ iff $\boldsymbol{q}'(\boldsymbol{a})$ holds in the interpretation given by $\mathcal{A}$, for any ABox $\mathcal{A}$ and any $\boldsymbol{a} \subseteq \mathsf{ind}(\mathcal{A})$.

A *datalog program*, Π, is a finite set of *rules* $\forall \boldsymbol{z}\, (\gamma_0 \leftarrow \gamma_1 \wedge \dots \wedge \gamma_m)$, where each γ_i is an atom $Q(\boldsymbol{y})$ with $\boldsymbol{y} \subseteq \boldsymbol{z}$ or an equality $(z = z')$ with $z, z' \in \boldsymbol{z}$. (As usual, we omit the prefix $\forall \boldsymbol{z}$.) 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, and $=$ can only occur in the body. The predicates in the head of rules are *IDB predicates*, the rest *EDB predicates*.

A *datalog query* is a pair $(\Pi, G(\boldsymbol{x}))$, where Π is a datalog program and $G(\boldsymbol{x})$ an atom. A tuple $\boldsymbol{a} \subseteq \mathsf{ind}(\mathcal{A})$ is an *answer to* $(\Pi, G(\boldsymbol{x}))$ *over* an ABox $\mathcal{A}$ if $G(\boldsymbol{a})$ holds in the FO-structure with domain $\mathsf{ind}(\mathcal{A})$ obtained by closing $\mathcal{A}$ under the rules in Π, in which case we write $\Pi, \mathcal{A} \models G(\boldsymbol{a})$. A datalog query $(\Pi, G(\boldsymbol{x}))$ is a *datalog rewriting* of an OMQ $\boldsymbol{Q} = (\mathcal{D}is, \boldsymbol{q}(\boldsymbol{x}))$ in case $\mathcal{D}is, \mathcal{A} \models \boldsymbol{q}(\boldsymbol{a})$ iff $\Pi, \mathcal{A} \models G(\boldsymbol{a})$, for any ABox $\mathcal{A}$ and any $\boldsymbol{a} \subseteq \mathsf{ind}(\mathcal{A})$. The *evaluation problem* for $(\Pi, G(\boldsymbol{x}))$—i.e., checking, given an ABox $\mathcal{A}$ and a tuple $\boldsymbol{a} \subseteq \mathsf{ind}(\mathcal{A})$, whether $\Pi, \mathcal{A} \models G(\boldsymbol{a})$—is known to be in P. Evaluation of a datalog query with a *linear* program, where the body of any rule has at most one IDB predicate, can be done in NL; see [11] and references therein. The NL upper bound also holds for datalog queries with linear-stratified programs that are defined as follows. A *stratified* program [1] is a sequence $\Pi = (\Pi_0, \dots, \Pi_n)$ of datalog programs, called the *strata* of Π, such that each predicate in Π can occur in the head of a rule only in one stratum Π_i and can occur in the body of a rule only in strata Π_j with $j \geq i$. If, additionally, the body of each rule in Π contains at most one occurrence of a head predicate from the same stratum, we call Π *linear-stratified*. It is shown in [2] that every linear-stratified program (called there piecewise linear) can be converted in an equivalent linear datalog program.

3 AC^0/NL/P Trichotomy for F-Tree OMQs

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 $F(x) \notin \boldsymbol{q}$. An *$F$-tree CQ* is a CQ $\boldsymbol{q}$ with a single solitary $F(x)$ such that the binary atoms in $\boldsymbol{q}$ form a directed tree with root x.

Our first observation is that answering any OMQ $\boldsymbol{Q} = (\mathcal{D}is_A, \boldsymbol{q})$ with an F-tree CQ $\boldsymbol{q}$ is either in AC^0 or NL-complete or P-complete. We obtain this trichotomy using a recent result of Lutz and Sabellek [18] establishing such a trichotomy for OMQs of the form $(\mathcal{T}, G(x))$, where $\mathcal{T}$ is an ontology formulated in the description logic $\mathcal{EL}$ [4] and G is a concept name (unary predicate).

Theorem 1. *Answering any OMQ $\boldsymbol{Q} = (\mathcal{D}is_A, \boldsymbol{q})$ with an F-tree CQ $\boldsymbol{q}$ is either in* AC^0 *or* NL*-complete or* P*-complete.*

Proof. Let $\Pi_{\boldsymbol{Q}}$ be the datalog program with the following rules:

$$\begin{aligned} G &\leftarrow F(x) \land \tilde{\boldsymbol{q}}'(x, y_1, \dots, y_n) \land P(y_1) \land \dots \land P(y_n), \\ P(x) &\leftarrow T(x), \\ P(x) &\leftarrow A(x) \land \tilde{\boldsymbol{q}}'(x, y_1, \dots, y_n) \land P(y_1) \land \dots \land P(y_n), \end{aligned}$$

where $\boldsymbol{q}'$ is obtained from $\boldsymbol{q}$ by removing all of its solitary occurrences of T- and F-atoms and $\tilde{\boldsymbol{q}}'$ is the result of omitting all the $\exists$ from $\boldsymbol{q}'$. As shown in [10, Theorem 7], for any ABox $\mathcal{A}$, we have $\mathcal{D}is_A, \mathcal{A} \models \boldsymbol{q}$ iff $\Pi_{\boldsymbol{Q}}, \mathcal{A} \models G$.

Denote by $\mathcal{T}_{\boldsymbol{Q}}$ the $\mathcal{EL}$ TBox with two concept inclusions:

$$T \sqsubseteq P, \qquad A \sqcap C_{\boldsymbol{q}} \sqsubseteq P,$$

where $C_{\boldsymbol{q}}$ is an $\mathcal{EL}$-concept representing $\boldsymbol{q} \setminus \{F(x)\}$. For example, for

$$\boldsymbol{q} = F(x) \land R_1(x, y_1) \land F(y_1) \land T(y_1) \land R_2(x, y_2) \land R_3(y_2, y_3) \land T(y_3),$$

we have

$$C_{\boldsymbol{q}} = \exists R_1.(F \sqcap T) \sqcap \exists R_2.\exists R_3.T.$$

It is readily seen that, for any ABox $\mathcal{A}$ and any $a \in \mathsf{ind}(\mathcal{A})$, we have $\Pi_{\boldsymbol{Q}}, \mathcal{A} \models P(a)$ iff $\mathcal{T}_{\boldsymbol{Q}}, \mathcal{A} \models P(a)$.

Finally, we observe that (i) answering $\boldsymbol{Q}$ is in AC^0 iff answering $(\mathcal{T}_{\boldsymbol{Q}}, P(x))$ is in AC^0; (ii) answering $\boldsymbol{Q}$ is NL-complete iff answering $(\mathcal{T}_{\boldsymbol{Q}}, P(x))$ is NL-complete; (iii) answering $\boldsymbol{Q}$ is P-complete iff answering $(\mathcal{T}_{\boldsymbol{Q}}, P(x))$ is P-complete.

Note that [18] gives an ExpTime algorithm for checking which of the three complexity classes a given $\mathcal{EL}$-OMQ of the form $(\mathcal{T}, G(x))$ falls into. However, applying this algorithm in our case is tricky because the input ontology $\mathcal{T}_{\boldsymbol{Q}}$ must first be converted to a normal form. In particular, it does not give clear syntactic criteria on the shape of the CQ $\boldsymbol{q}$ that would guarantee that the OMQ $(\mathcal{D}is_A, \boldsymbol{q})$ belongs to the desired complexity class (see examples below). Note also that the reduction in the proof above does not work for CQs that are not F-trees.

4 AC^0

As shown in [10], answering any CQ $\boldsymbol{Q} = (\mathcal{D}is_A, \boldsymbol{q})$ is in AC^0 if the CQ $\boldsymbol{q}$ does not have solitary occurrences of F (or T). This sufficient condition becomes also a necessary one if $\boldsymbol{q}$ is a *path CQ*, that is, the variables $x_0, \dots, x_n$ in $\boldsymbol{q}$ are ordered so that

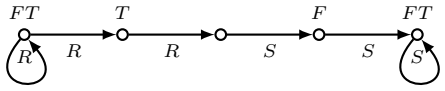
– the binary atoms in $\boldsymbol{q}$ form a chain $R_1(x_0, x_1), \dots, R_n(x_{n-1}, x_n)$;
– the unary atoms in $\boldsymbol{q}$ are of the form $T(x_i)$ and $F(x_j)$, for some i and j with $0 \leq i, j \leq n$.

In fact, we have the following AC^0/NL-dichotomy for OMQs $\boldsymbol{Q} = (\mathcal{D}is_A, \boldsymbol{q})$ with a path CQ $\boldsymbol{q}$ [10]:

– either $\boldsymbol{q}$ does not contain a solitary F or a solitary T, and answering $\boldsymbol{Q}$ is in AC^0,
– or $\boldsymbol{q}$ contains both solitary F and T, and answering $\boldsymbol{Q}$ is NL-hard.

Here, we give an example showing that this dichotomy collapses for path CQs with loops of the form $R(x, x)$.

Proposition 1. *Answering the OMQ $(\mathcal{D}is_A, \boldsymbol{q})$, where $\boldsymbol{q}$ is the CQ with a solitary F and a solitary T shown in the picture below, is in AC^0 for data complexity.*



Proof. It suffices to show that $\mathcal{D}is_A, \mathcal{A} \models \boldsymbol{q}$ iff $\mathcal{A} \models \boldsymbol{q}$. The implication ($\Leftarrow$) is trivial.

($\Rightarrow$) Suppose $\mathcal{A} \not\models \boldsymbol{q}$. Let $x_1, \dots, x_5$ be the consecutive variables in $\boldsymbol{q}$. We construct a model $\mathcal{I}$ of $\mathcal{D}is_A$ with $\mathcal{I} \not\models \boldsymbol{q}$. Consider the following subsets of $\mathsf{ind}(\mathcal{A})$:

$$
\begin{aligned}
B_R &= \{a \in \mathsf{ind}(\mathcal{A}) \mid R(a,a) \in \mathcal{A},\ F(a) \in \mathcal{A},\ T(a) \in \mathcal{A}\},\\
B_S &= \{a \in \mathsf{ind}(\mathcal{A}) \mid S(a,a) \in \mathcal{A},\ F(a) \in \mathcal{A},\ T(a) \in \mathcal{A}\},\\
X &= \{a \in \mathsf{ind}(\mathcal{A}) \mid R(b,a) \text{ for some } b \in B_R\},\\
Y &= \{a \in \mathsf{ind}(\mathcal{A}) \mid S(a,b) \text{ for some } b \in B_S\}.
\end{aligned}
$$

Note that since $\mathcal{A} \not\models \boldsymbol{q}$, the sets X and Y do not intersect. Indeed, if $b \in X \cap Y$, then B_R contains some element a such that $R(a, b) \in \mathcal{A}$, B_S contains some c with $S(b, c) \in \mathcal{A}$, and the map h given by $h(x_1) = h(x_2) = a$, $h(x_3) = b$ and $h(x_4) = h(x_5) = c$ is a homomorphism from $\boldsymbol{q}$ to $\mathcal{A}$. Define a model $\mathcal{I}$ of $\mathcal{D}is_A$ by extending $\mathcal{A}$ with

– $F(a)$, for all $a \in X$;
– $T(a)$, for all $a \in \mathsf{ind}(\mathcal{A}) \setminus X$.

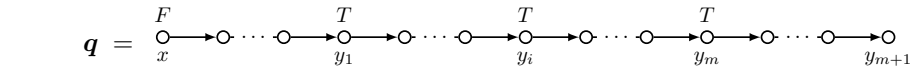
We claim that $\mathcal{I} \not\models \boldsymbol{q}$. Indeed, suppose there is a homomorphism $h\colon \boldsymbol{q} \to \mathcal{I}$. Clearly, $h(x_1) \in B_R$ and $h(x_5) \in B_S$. It follows that $h(x_2) \in X$ and $h(x_4) \in Y$. Since $T(x_2) \in \boldsymbol{q}$, we have $h(x_2) \in T^{\mathcal{I}}$, and so $T(h(x_2)) \in \mathcal{A}$. Similarly, $F(x_4) \in \mathcal{A}$. It follows that h is a homomorphism from $\boldsymbol{q}$ to $\mathcal{A}$, contrary to our assumption.

Note that the CQ $\boldsymbol{q}$ above is *minimal* (not equivalent to any of its proper sub-CQs).

5 NL vs. P

A path CQ $\boldsymbol{q}$ is called an *F-path CQ* if $\boldsymbol{q}$ has a single solitary occurrence of F at its root; in other words, $\boldsymbol{q}$ is both a path CQ and an F-tree CQ. We represent such a $\boldsymbol{q}$ as shown in the picture below, which indicates *all* the solitary occurrences of F and T:

$\boldsymbol{q}$ =



We know from [10] that

- answering OMQs $(\mathcal{D}is_A, \boldsymbol{q})$ with F-path CQs can be done in P;
- if $x, y_1, \dots, y_m$ are all the variables in $\boldsymbol{q}$, then answering $(\mathcal{D}is_A, \boldsymbol{q})$ is NL-complete.

There is also a table in [10] with quite a few odd examples of CQs of both kinds. Our next result sheds some light on the left column of this table.

We require the following sub-CQs of the F-path CQ $\boldsymbol{q}$ shown above:

- $\boldsymbol{q}_i$ 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 $F(x)$.

We 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 2. *If there exist $f_i \colon \boldsymbol{q}_i \twoheadrightarrow \boldsymbol{q}$, for $1 \leq i \leq m$, then $(\mathcal{D}is_A, \boldsymbol{q})$ is* NL*-complete.*

Proof. Let Π be a linear datalog program with the following rules:

$$G \leftarrow F(x) \wedge \tilde{\boldsymbol{q}}_{m+1}^*, \tag{$r1$}$$

$$G \leftarrow F(x) \wedge \tilde{\boldsymbol{q}}_i^* \wedge P(y_i), \quad \text{for } 1 \leq i \leq m, \tag{$r2$}$$

$$P(x) \leftarrow A(x) \wedge \tilde{\boldsymbol{q}}_{m+1}^*, \tag{$r3$}$$

$$P(x) \leftarrow A(x) \wedge \tilde{\boldsymbol{q}}_i^* \wedge P(y_i), \quad \text{for } 1 \leq i \leq m. \tag{$r4$}$$

It suffices to show that, for any ABox $\mathcal{A}$, we have $\mathcal{D}is_A, \mathcal{A} \models \boldsymbol{q}$ iff $\Pi, \mathcal{A} \models G$.

($\Rightarrow$) Suppose $\mathcal{D}is_A, \mathcal{A} \models \boldsymbol{q}$. Let $V_P = \{a \in \mathsf{ind}(\mathcal{A}) \mid \Pi, \mathcal{A} \models P(a)\}$. Define an interpretation $\mathcal{I}$ with domain $\mathsf{ind}(\mathcal{A})$ by taking

$$T^{\mathcal{I}} = \{a \mid T(a) \in \mathcal{A}\} \cup \{a \in V_P \mid A(a) \in \mathcal{A},\ F(a) \notin \mathcal{A}\},$$
$$F^{\mathcal{I}} = \{a \mid F(a) \in \mathcal{A}\} \cup \{a \notin V_P \mid A(a) \in \mathcal{A},\ T(a) \notin \mathcal{A}\}.$$

Clearly, $\mathcal{I} \models \mathcal{D}is_A$, and so there is a homomorphism $h \colon \boldsymbol{q} \to \mathcal{I}$. We show now that $\Pi, \mathcal{A} \models G$. Note that we have both $a \in F^{\mathcal{I}}$ and $a \in T^{\mathcal{I}}$ only if $F(a), T(a) \in \mathcal{A}$.

Case 1: $T(h(y_i)) \in \mathcal{A}$, for $1 \leq i \leq m$. Then $\Pi, \mathcal{A} \models G$ by $(r1)$ since $h(x) \in F^{\mathcal{I}}$ can only be because $F(h(x)) \in \mathcal{A}$ (if this is not the case, then $A(h(x)) \in \mathcal{A}$ and we have $h(x) \in V_P$ by $(r3)$, which is a contradiction).

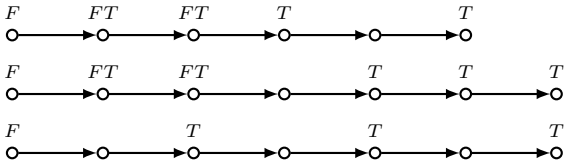
Case 2: $T(h(y_i)) \notin \mathcal{A}$, for some i ($1 \leq i \leq m$). Let i be minimal with this property. By the definition of $\mathcal{I}$, we then have $h(y_i) \in V_P$ and $A(h(y_i)) \in \mathcal{A}$. Then $\Pi, \mathcal{A} \models G$ by $(r2)$ since $h(x) \in F^{\mathcal{I}}$ can only be because $F(h(x)) \in \mathcal{A}$ (if this is not the case, then $A(h(x)) \in \mathcal{A}$ and we have $h(x) \in V_P$ by $(r4)$, which is a contradiction).

($\Leftarrow$) Suppose there is a derivation of G from Π and $\mathcal{A}$. Then there exist a sequence of homomorphisms

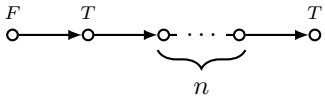
$$h_1 \colon \boldsymbol{q}^*_{m+1} \to \mathcal{A}, \quad h_2 \colon \boldsymbol{q}^*_i \to \mathcal{A}, \quad \dots, \quad h_k \colon \boldsymbol{q}^*_j \to \mathcal{A},$$

for some $i, \dots, j \leq m$, with $h_1(x) = h_2(y_i), \dots, h_{k-1}(x) = h_2(y_j)$, $F(h_k(x)) \in \mathcal{A}$ and $A(h_n(x)) \in \mathcal{A}$, for $1 \leq n \leq k-1$. Now, consider any model $\mathcal{I}$ of $\mathcal{D}is_A$ extending $\mathcal{A}$ and show that $\mathcal{I} \models \boldsymbol{q}$. If $h_1(x) \in F^{\mathcal{I}}$, then h_1 is a homomorphism from $\boldsymbol{q}$ to $\mathcal{I}$. So, let $h_1(x) \in T^{\mathcal{I}}$. Then the homomorphisms h_2 and f_i give us a homomorphism $h'_2 \colon \boldsymbol{q}^*_{m+1} \to \mathcal{I}$ such that $h'_2(x) = h_3(y_l)$. Again, if $h'_2(x) \in F^{\mathcal{I}}$, then h'_2 is a homomorphism from $\boldsymbol{q}$ to $\mathcal{I}$. Otherwise, we combine h'_2 with f_l, and so on. As $h_k(x) \in F^{\mathcal{I}}$, sooner or later we must obtain a homomorphism from $\boldsymbol{q}$ to $\mathcal{I}$.

Example 1. By Theorem 2, the following CQs $\boldsymbol{q}$ give NL-complete OMQs $(\mathcal{D}is_A, \boldsymbol{q})$:



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



Clearly, Theorem 2 only applies to $\boldsymbol{q}_{T0T}$. Our next results show that, surprisingly, $(\mathcal{D}is_\top, \boldsymbol{q}_{T1T})$ is NL-complete, $(\mathcal{D}is_A, \boldsymbol{q}_{T1T})$ is P-complete, and $(\mathcal{D}is_\top, \boldsymbol{q}_{TnT})$ is P-complete, for every $n \geq 2$ (where, as usual, $\top$ denotes the class of all domain individuals).

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

Proof. The NL-hardness follows from [10, Theorem 4]. To establish the matching upper bound, consider the datalog program Π' with the following rules:

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

As shown in [10, Theorem 7], $\mathcal{D}is_\top, \mathcal{A} \models \boldsymbol{q}_{T1T}$ iff $\Pi', \mathcal{A} \models G$. Now, consider a program Π with the single rule

$$T(x) \leftarrow R(x,y) \land T(y) \land R(y,z) \land R(z,u) \land T(u). \tag{r}$$

It is not hard to see that if checking whether $\Pi, \mathcal{A} \models T(a)$, for any given $a \in \mathsf{ind}(\mathcal{A})$, can be done in NL, then checking whether $\Pi', \mathcal{A} \models G$ can also be done in NL. Thus, it suffices to show that checking whether $\Pi, \mathcal{A} \models T(a)$ can be done in NL.

Let $\Pi^\dagger$ be the *linear stratified* datalog program with the following rules:

$$P(x) \leftarrow R(x,y) \land T(y) \land R(y,z) \land R(z,v) \land T(v), \tag{$r1$}$$
$$P(x) \leftarrow R(x,y) \land T(y) \land R(y,z) \land R(z,v) \land P(v), \tag{$r1'$}$$
$$Q(x) \leftarrow R(x,y) \land P(y) \land R(y,z) \land R(z,v) \land T(v), \tag{$r2$}$$
$$Q(x) \leftarrow R(x,y) \land P(y) \land R(y,z) \land R(z,v) \land P(v), \tag{$r2'$}$$
$$Q(x) \leftarrow R(x,y) \land Q(y), \tag{$r3$}$$
$$G(x) \leftarrow T(x), \tag{$r4$}$$
$$G(x) \leftarrow P(x), \tag{$r5$}$$
$$G(x) \leftarrow Q(x). \tag{$r6$}$$

Checking whether $\Pi^\dagger, \mathcal{A} \models G(a)$ can be done in NL. We claim that $\Pi^\dagger, \mathcal{A} \models G(a)$ iff $\Pi, \mathcal{A} \models T(a)$, for any ABox $\mathcal{A}$ and any $a \in \mathsf{ind}(\mathcal{A})$.

($\Rightarrow$) Suppose $\Pi^\dagger, \mathcal{A} \models G(a)$. By $(r4)$–$(r6)$, we have one of the following cases:

Case 1: $\Pi^\dagger, \mathcal{A} \models T(a)$. Then trivially $\Pi, \mathcal{A} \models T(a)$.
Case 2: $\Pi^\dagger, \mathcal{A} \models P(a)$. Then $\Pi, \mathcal{A} \models T(a)$ by $(r1)$ and $(r2)$.
Case 3: $\Pi^\dagger, \mathcal{A} \models Q(a)$. Then, by $(r3)$–$(r4)$, there are $a_0, a_1, \ldots, a_n, a_{n+1}$ such that
- $a = a_0$;
- $R(a_i, a_{i+1}) \in \mathcal{A}$, for $0 \leq i \leq n$;
- $\Pi^\dagger, \mathcal{A} \models P(a_{n+1})$;
- there are $z', v' \in \mathsf{ind}(\mathcal{A})$ with $R(a_{n+1}, z'), R(z', v') \in \mathcal{A}$ and $\Pi^\dagger, \mathcal{A} \models P(v')$.

As in case 2, $\Pi, \mathcal{A} \models T(a_{n+1})$ and $\Pi, \mathcal{A} \models T(v')$, from which $\Pi, \mathcal{A} \models T(a_n)$. As $\Pi^\dagger, \mathcal{A} \models P(a_{n+1})$, there is an R-successor a_{n+2} of a_{n+1} with $\Pi, \mathcal{A} \models T(a_{n+2})$. But then (r) is applicable at a_{n-1} (with y being a_n, z being a_{n+1} and v being a_{n+2}). By iteratively applying (r) for $i = n-1, n-2, \ldots, 0$, we conclude that $\Pi, \mathcal{A} \models T(a_0)$.

($\Leftarrow$) Suppose $\Pi, \mathcal{A} \models T(r)$. Then there is a finite 2-ary (derivation) tree $\mathfrak{T}$ such that

- the vertices v of $\mathfrak{T}$ are some elements from $\mathsf{ind}(\mathcal{A})$;
- r is the root of $\mathfrak{T}$;
- any vertex v of $\mathfrak{T}$ either is a leaf or has 2 successors: 'left' v_1 and 'right' v_2 such that $\mathcal{A} \models R(v, v_1) \land R(v_1, w) \land R(w, v_2)$, for some $w \in \mathsf{ind}(\mathcal{A})$;
- if v is a leaf, then $T(v) \in \mathcal{A}$.

We prove that $\Pi^\dagger, \mathcal{A} \models G(r)$ by induction on the depth of $\mathfrak{T}$. The basis of induction ($\mathfrak{T}$ of depth 0) is trivial. For the induction step, we define inductively a finite sequence $u_0, d_0, u_1, d_1, \ldots, d_{n-1}, u_n$, where the u_i are vertices of $\mathfrak{T}$ and $d_i \in \{\mathfrak{r}, \mathfrak{l}\}$). First, we set $u_0 = r$. Now, suppose u_i has been defined. If u_i is a leaf of $\mathfrak{T}$, we stop and set $n = i$. Otherwise, let v_1 and v_2 be, respectively, the left and right successors of u_i in $\mathfrak{T}$. If v_1 is not a leaf, we set $d_i = \mathfrak{l}$ and $u_{i+1} = v_1$. Otherwise, we set $d_i = \mathfrak{r}$ and $u_{i+1} = v_2$. Note that

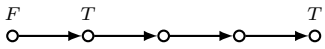
- $d_{n-1} = \mathfrak{r}$, if $n \geq 1$;
- if $d_i = \mathfrak{r}$, there are $y, w \in \mathsf{ind}(\mathcal{A})$ with $R(u_i, y), T(y), R(y, w), R(w, u_{i+1}) \in \mathcal{A}$.

Now, we have two cases depending on the sequence $\mathfrak{dir} = d_0, d_1, \ldots, d_{n-1}$.

Case 1: $\mathfrak{dir}$ does not contain $\mathfrak{l}$. Then we can show by induction on i from $n-1$ to 0 using (ii) that $\Pi^\dagger, \mathcal{A} \models P(u_i)$, for $0 \leq i \leq n-1$. It follows that $\Pi^\dagger, \mathcal{A} \models G(r)$.

Case 2: $\mathfrak{dir}$ contains at least one $\mathfrak{l}$. Let k be such that the last occurrence of $\mathfrak{l}$ in $\mathfrak{dir}$ is between u_k and u_{k+1}. By (i), $k+1 < n$, and so u_{k+2} is well defined. The argument from case 1 shows that $\Pi^\dagger, \mathcal{A} \models P(u_{k+1})$. By IH, $\Pi^\dagger, \mathcal{A} \models G(y)$ for the right successor y of u_k. This means that either $\Pi^\dagger, \mathcal{A} \models Q(y)$ or $\Pi^\dagger, \mathcal{A} \models P(y)$ or $\Pi^\dagger, \mathcal{A} \models T(y)$. In the first case, we obtain $\Pi^\dagger, \mathcal{A} \models Q(r)$ using $(r3)$ and the fact that y is accessible from r via R in $\mathcal{A}$. In last two cases (using $(r2)$ or $(r2')$), we have $\Pi^\dagger, \mathcal{A} \models Q(u_k)$. By construction, u_k is accessible from r via R in $\mathcal{A}$, and so $\Pi^\dagger, \mathcal{A} \models Q(r)$. It follows that $\Pi^\dagger, \mathcal{A} \models G(r)$.

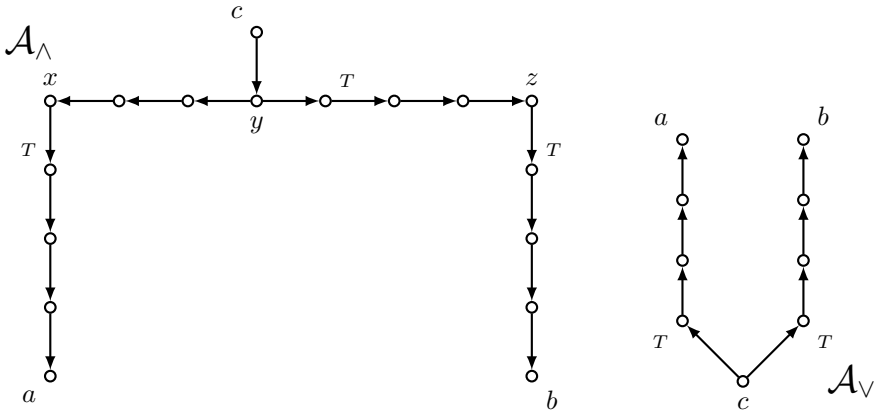
Theorem 3. *Answering any OMQ $(\mathcal{D}is_\top, \boldsymbol{q}_{TnT})$, for $n \geq 2$, is* P*-complete.*



Proof. We sketch a proof for $\boldsymbol{q}_{T2T}$ shown in the picture above and leave the general case to the reader. Let Π be the program with the single rule

$$T(x) \leftarrow R(x,y) \wedge T(y) \wedge R(y,z) \wedge R(z,u) \wedge R(z,v) \wedge T(v).$$

It suffices to show that checking whether $\Pi, \mathcal{A} \models T(a)$, for $\mathcal{A}$ and $a \in \mathsf{ind}(\mathcal{A})$, is P-hard. Consider the following two ABoxes:



It is routine to verify the following properties of these ABoxes:

$\wedge$-gadget

$$\Pi, \mathcal{A}_\wedge \cup \{T(a), T(b)\} \models T(c),$$
$$\Pi, \mathcal{A}_\wedge \cup \{T(a)\} \not\models T(c),$$
$$\Pi, \mathcal{A}_\wedge \cup \{T(b)\} \not\models T(c),$$
$$\Pi, \mathcal{A}_\wedge \cup \{T(a), T(b), R(c', c)\} \not\models T(c');$$

$\vee$-gadget

$$\Pi, \mathcal{A}_\vee \cup \{T(a)\} \models T(c),$$
$$\Pi, \mathcal{A}_\vee \cup \{T(b)\} \models T(c),$$
$$\Pi, \mathcal{A}_\vee \not\models T(c),$$
$$\Pi, \mathcal{A}_\vee \cup \{T(a), T(b), R(c', c)\} \not\models T(c').$$

Now, with any monotone Boolean circuit $\mathcal{C}$ with an output o and all gates having exactly two inputs, we associate an ABox $\mathcal{A}_\mathcal{C}$ by replacing every AND-gate in $\mathcal{C}$ with inputs a and b and output c by a fresh copy of $\mathcal{A}_\wedge$, and every OR-gate with inputs a and b and output c by a fresh copy of $\mathcal{A}_\vee$. Given an input $\boldsymbol{\alpha}$ for $\mathcal{C}$, we place atoms $T(a)$ on the input gates a (which are also individuals of $\mathcal{A}_\mathcal{C}$) with $\boldsymbol{\alpha}(a) = 1$, and denote the resulting ABox by $\mathcal{A}^{\boldsymbol{\alpha}}_\mathcal{C}$. We claim that $\mathcal{C}$ outputs 1 under $\boldsymbol{\alpha}$ iff $\mathcal{A}^{\boldsymbol{\alpha}}_\mathcal{C}, \Pi \models T(o)$.

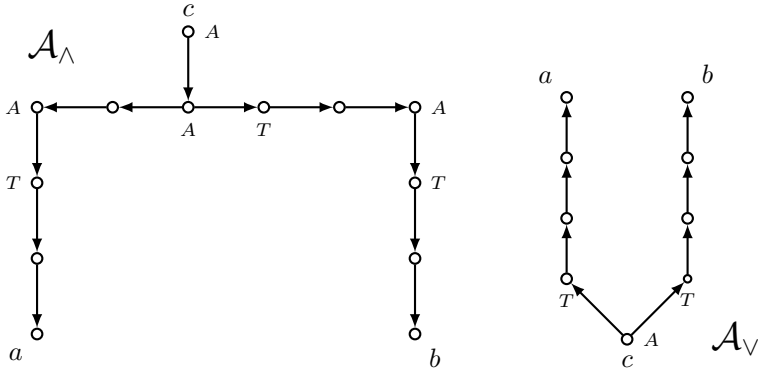
The implication ($\Rightarrow$) is proved by induction, using the properties of $\mathcal{A}_\wedge$ and $\mathcal{A}_\vee$, that if a gate g of $\mathcal{C}$ outputs 1 under $\boldsymbol{\alpha}$, then $\Pi, \mathcal{A}^{\boldsymbol{\alpha}}_\mathcal{C} \models T(g)$.

($\Leftarrow$) Suppose $\mathcal{C}$ outputs 0. Define an ABox $\mathcal{A}$ by extending $\mathcal{A}^{\boldsymbol{\alpha}}_\mathcal{C}$ as follows. We add atoms $T(c)$ for all gates g that output 1 under $\boldsymbol{\alpha}$, atoms $T(x)$ for those copies of $\mathcal{A}_\wedge$ that correspond to an AND-gate having 1 as its left input, and atoms $T(y)$ and $T(z)$ for those copies of $\mathcal{A}_\wedge$ that correspond to an AND-gate having 1 as its right input. It is readily checked that no rule in Π can be applied to $\mathcal{A}$. Since $\mathcal{C}$ outputs 0, it follows that $\Pi, \mathcal{A}^{\boldsymbol{\alpha}}_\mathcal{C} \not\models T(o)$.

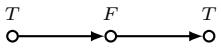
(The reader may want to figure out which part of the proof goes wrong for $n = 1$.) On the other hand we have:

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

The proof is similar to that of Theorem 3 and uses the following gadgets $\mathcal{A}_\wedge$, $\mathcal{A}_\vee$:

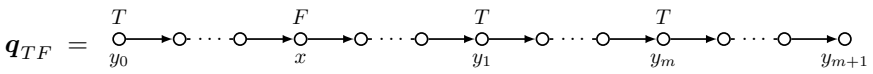


So far in this section we have considered OMQs with F-path CQs, thus excluding path CQs such as $\boldsymbol{q}$ in the picture below



As shown in [10], answering the OMQ $(\mathcal{D}is_A, \boldsymbol{q})$ with this $\boldsymbol{q}$ is P-complete; in fact, it follows from the proof that $(\mathcal{D}is_\top, \boldsymbol{q})$ is P-complete, too. This tempted us to conjecture that having a solitary F in the middle of a path CQ with solitary T's on both sides ensures P-hardness. To our surprise, there is a family of path CQs of this shape that are NL-complete.

A path CQ $\boldsymbol{q}_{TF}$ is called a *TF-path CQ* if it is of the form



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 2 (and $\boldsymbol{q}^*_{m+1}$ is $\boldsymbol{q}$ without $F(x)$).

Theorem 4. *If $\boldsymbol{q}$ satisfies the condition of Theorem 2 and there is a homomorphism $h\colon \boldsymbol{q}^*_{m+1} \to \boldsymbol{q}_0$ such that $h(x) = y_0$, then answering $(\mathcal{D}is_A, \boldsymbol{q}_{TF})$ is* NL*-complete.*

Proof. We use the notations introduced for Theorem 2. Let Π be the following linear-stratified datalog program:

$$G \leftarrow F(x) \wedge P(x) \wedge Q(x), \tag{$r1$}$$
$$G \leftarrow F(x) \wedge Q(x) \wedge \tilde{\boldsymbol{q}}^*_{m+1}, \tag{$r2$}$$
$$P(x) \leftarrow A(x) \wedge \tilde{\boldsymbol{q}}^*_{m+1}, \tag{$r3$}$$
$$P(x) \leftarrow A(x) \wedge \tilde{\boldsymbol{q}}^*_i \wedge P(y_i) \wedge Q(y_i), \tag{$r4$}$$
$$Q(x) \leftarrow T(y_0) \wedge \tilde{\boldsymbol{q}}_0(y_0, x), \tag{$r5$}$$
$$Q(x) \leftarrow A(y_0) \wedge Q(y_0) \wedge \tilde{\boldsymbol{q}}_0(y_0, x). \tag{$r6$}$$

It suffices to prove that $\Pi, \mathcal{A} \models G$ iff $\mathcal{D}is_A, \mathcal{A} \models \boldsymbol{q}_{TF}$, for all ABoxes $\mathcal{A}$.

($\Leftarrow$) Suppose $\mathcal{D}is_A, \mathcal{A} \models \boldsymbol{q}_{TF}$. Let $V_P = \{a \in \mathsf{ind}(\mathcal{A}) \mid \Pi, \mathcal{A} \models P(a)\}$ and $V_Q = \{a \in \mathsf{ind}(\mathcal{A}) \mid \Pi, \mathcal{A} \models Q(a)\}$. Define an interpretation $\mathcal{I}$ with domain $\mathsf{ind}(\mathcal{A})$ by taking

$$T^{\mathcal{I}} = \{a \mid T(a) \in \mathcal{A}\} \cup \{a \in V_P \text{ or } a \in V_Q \mid F(a) \notin \mathcal{A}\},$$
$$F^{\mathcal{I}} = \{a \mid F(a) \in \mathcal{A}\} \cup \{a \notin V_P \text{ and } a \notin V_Q \mid T(a) \notin \mathcal{A}\}.$$

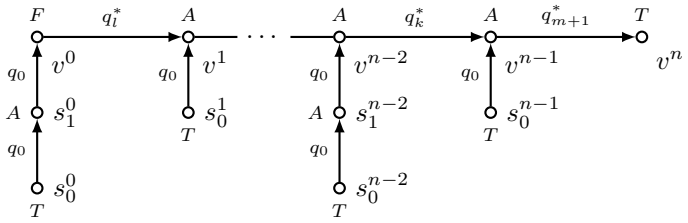
Note that we have both $a \in F^{\mathcal{I}}$ and $a \in T^{\mathcal{I}}$ only if $F(a), T(a) \in \mathcal{A}$. Clearly, $\mathcal{I}$ is a model of $(\mathcal{D}is_A, \mathcal{A})$, and so there is a homomorphism $f\colon \boldsymbol{q}_{TF} \to \mathcal{I}$. We show now that $\Pi, \mathcal{A} \models G$. First, we have $\Pi, \mathcal{A} \models Q(f(x))$. Indeed, if $T(f(y_0)) \in \mathcal{A}$, then we can use $(r5)$. If $A(f(y_0)) \in \mathcal{A}$, then $f(y_0) \in V_Q$ (using $r6$) and $f(y_0) \in T^{\mathcal{I}}$ follows from the definition of $\mathcal{I}$. So, $f(x) \in V_Q$ is again obtained by $(r5)$. Second, there are two similar cases. If $T(f(y_i)) \in \mathcal{A}$, for $1 \leq i \leq m$, then $\Pi, \mathcal{A} \models G$ by $(r2)$. Otherwise, we take the smallest i such that $T(f(y_i)) \notin \mathcal{A}$. Then $A(f(y_i)) \in \mathcal{A}$ and, by the definition of $\mathcal{I}$, we have $f(y_i) \in V_P$ (using $(r3)$ or $(r4)$) and $f(y_i) \in T^{\mathcal{I}}$, and so again $\Pi, \mathcal{A} \models G$ by $(r1)$.

($\Rightarrow$) Suppose there is a derivation of G from Π and $\mathcal{A}$. Then $\mathcal{D}is_A, \mathcal{A} \models \boldsymbol{q}$ and there exists a sequence $v^0, v^1, \dots, v^n \in \mathsf{ind}(\mathcal{A})$ such that:

- $F(v^0) \in \mathcal{A}$;
- $A(v^i) \in \mathcal{A}$ and $v^i \in V_P$, for $1 \leq i < n$;
- for each i ($0 \leq i < n$), we have $\boldsymbol{q}^*_j(v^i, v^{i+1})$, for some $j \in \{1, \dots, m\}$;
- $\boldsymbol{q}^*_{m+1}(v^{n-1}, v^n) \wedge T(v^n) \in \mathcal{A}$.

Moreover, there are also paths $s^i_0, s^i_1, \dots, s^i_{k_i}$, where $v^i = s^i_{k_i}$ and $0 \leq i \leq n$, such that

– $T(s^i_0) \in \mathcal{A}$;
– $A(s^i_j) \in \mathcal{A}$ and $s^i_j \in V_Q$, for $1 \leq j \leq k_i$;
– for each j ($0 \leq j < k_i$), we have $\boldsymbol{q}_0(s^i_j, s^i_{j+1}) \in \mathcal{A}$;
– $A(s^i_{k_i}) \in \mathcal{A}$ and $s^i_{k_i} \in V_P$ or, if $i = 0$, then $F(s^0_{k_0}) \in \mathcal{A}$.

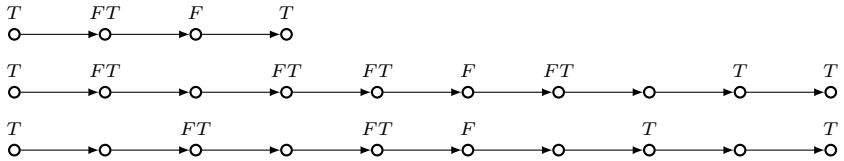


Let $\mathcal{I}$ be any interpretation based on $\mathcal{A}$. Let i be the maximal number such that $v_i \in F^{\mathcal{I}}$.

Case 1: $s^i_l \in T^{\mathcal{I}}$, for $0 \leq l < k_i$. In this case, there exists a homomorphism h_1 from $\boldsymbol{q}_{TF}$ to $\mathcal{I}$ such that $h_1(y_0) = s^i_{k_i-1}$, $h_1(x) = v_i$ and $h_1(y_j) = v_{i+1}$, where j is maximal with $T(y_j) \notin \mathcal{A}$. Then $\mathcal{D}is_A, \mathcal{A} \models \boldsymbol{q}_{TF}$, because $\boldsymbol{q}$ satisfies Theorem 2.

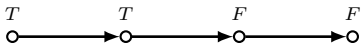
Case 2: otherwise. Let j be minimal with $s^i_j \in V_Q$ and $s^i_j \in F^{\mathcal{I}}$. Then there is a homomorphism h_2 from $\boldsymbol{q}_{TF}$ to $\mathcal{I}$ such that $h_2(y_0) = s^i_{j-1}$ and $h_2(x) = s^i_j$. We obtain $\mathcal{D}is_A, \mathcal{A} \models \boldsymbol{q}_{TF}$ using the homomorphism h.

Example 2. By Theorem 4, the following CQs $\boldsymbol{q}$ give NL-complete OMQs $(\mathcal{D}is_A, \boldsymbol{q})$:

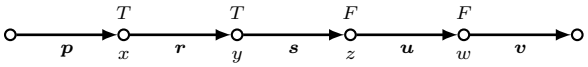


6 coNP

As shown in [10] answering $(\mathcal{D}is_A, \boldsymbol{q})$ with the CQ $\boldsymbol{q}$



is coNP-complete. Here, we generalise this observation. We say that a path CQ $\boldsymbol{q}$ is a *2-2-CQ* if it has at least two solitary T, at least two solitary F all of which are located after all the T, and every occurrence of T or F in $\boldsymbol{q}$ is solitary. We represent any given 2-2-CQ $\boldsymbol{q}$ as shown below

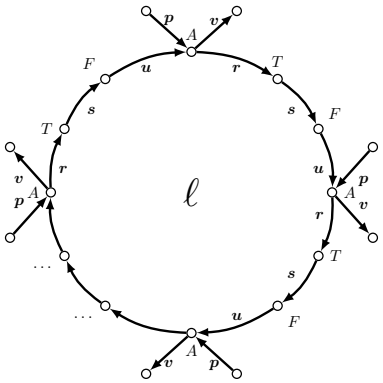


where $\boldsymbol{p}$, $\boldsymbol{r}$, $\boldsymbol{u}$ and $\boldsymbol{v}$ do not contain F and T, while $\boldsymbol{s}$ may contain solitary occurrences of both T and F (in other words, the T shown in the picture are the first two occurrences of T in $\boldsymbol{q}$ and the F are the last two occurrences of F in $\boldsymbol{q}$). Denote by $\boldsymbol{q}_{\boldsymbol{r}}$ the suffix of $\boldsymbol{q}$ that starts from x but without $T(x)$; similarly, $\boldsymbol{q}_{\boldsymbol{u}}$ 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 $F(w)$.

Theorem 5. *Answering any OMQ* $(\mathcal{D}is_A, \boldsymbol{q})$ *with a 2-2-CQ* $\boldsymbol{q}$ *is* coNP*-complete provided the following conditions are satisfied:*

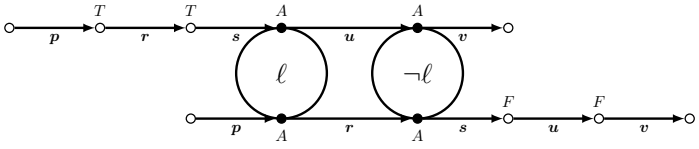
- *there is no homomorphism* $h_1 \colon \boldsymbol{q_u} \to \boldsymbol{q_r}$ *with* $h_1(z) = x$*;*
- *there is no homomorphism* $h_2 \colon \boldsymbol{q_r^-} \to \boldsymbol{q_u^-}$ *with* $h_2(y) = w$*.*

Proof. We prove coNP-hardness by reduction of the NP-complete 3SAT. Given a 3CNF ψ, we construct an ABox $\mathcal{A}_\psi$ as follows. First, for every literal ℓ whose propositional variable is present in ψ, we take the following ℓ-gadget that contains sufficiently many occurrences of A:



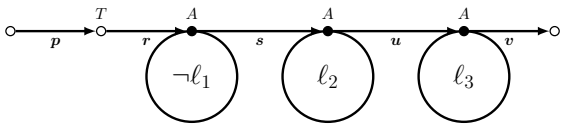
One can show that, for every model of $\mathcal{D}is_A$ extending this ℓ-gadget, we have $\mathcal{I} \not\models \boldsymbol{q}$ iff the A-points in the gadget are all in $T^{\mathcal{I}}$ or are all in $F^{\mathcal{I}}$.

Next, for every pair ℓ and $\neg\ell$ of literals as above, we connect the corresponding gadgets following the pattern in the picture below:



Now, one can show that, for every model of $\mathcal{D}is_A$ extending this new gadget, we have $\mathcal{I} \not\models \boldsymbol{q}$ iff either all A-points in the ℓ-gadget are in $T^{\mathcal{I}}$ and all A-points in the $\neg\ell$-gadget are in $F^{\mathcal{I}}$ or the other way round.

Finally, for every clause $c = (\ell_1 \vee \ell_2 \vee \ell_3)$ in ψ, we connect the $\neg\ell_1$-, ℓ_2- and ℓ_3-gadgets as shown below, always taking fresh A-points (by the construction, we have a sufficient supply of them):



Denote the resulting structure by $\mathcal{A}_\psi$. We leave it to the reader to verify, using the properties of the gadgets mentioned above, that ψ is satisfiable iff $\mathcal{D}is_A, \mathcal{A}_\psi \not\models \boldsymbol{q}$.

We do not know yet whether this theorem holds for $\mathcal{D}is_\top$ in place of $\mathcal{D}is_A$.

7 Conclusion

In this paper, we have obtained a few new results on the data complexity of answering a given ontology-mediated query (OMQ) that consists of a conjunctive query (CQ) and a covering axiom similar to the one used in the variant [16, Example 7] of the well-known 'Andrea example' [21]. We have observed that answering such OMQs is often tractable, with the respective OMQs being rewritable into standard datalog queries over the data. Sometimes we can even achieve rewritability into linear datalog, which guarantees OMQ answering in NL. We have given a few necessary and sufficient conditions for these phenomena. We have also discovered a few interesting counterexamples, in particular, a minimal CQ with solitary occurrences of both T and F that is FO-rewritable, a path CQ that is NL-complete for $\mathcal{D}is_{\top}$ but P-complete for $\mathcal{D}is_A$, and a path CQ with a solitary F in the middle and solitary Ts on either side of it that is NL-complete.

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; the work of V. Podolskii was supported by the Russian Academic Excellence Project '5-100' and by grant MK-7312.2016.1.

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. Artale, A., Calvanese, D., Kontchakov, R., Zakharyaschev, M.: The DL-Lite family and relations. Journal of Artificial Intelligence Research (JAIR) 36, 1–69 (2009)
4. Baader, F., Horrocks, I., Lutz, C., Sattler, U.: An Introduction to Description Logic. Cambridge University Press (2017)
5. 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)
6. Bienvenu, M., Ortiz, M.: Ontology-mediated query answering with data-tractable description logics. In: Reasoning Web. Web Logic Rules - 11th International Summer School 2015, Berlin, Germany, July 31 - August 4, 2015, Tutorial Lectures. pp. 218–307 (2015), https://doi.org/10.1007/978-3-319-21768-0_9
7. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: the *DL-Lite* family. Journal of Automated Reasoning 39(3), 385–429 (2007)
8. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Data complexity of query answering in description logics. Artif. Intell. 195, 335–360 (2013), https://doi.org/10.1016/j.artint.2012.10.003
9. Feier, C., Kuusisto, A., Lutz, C.: Rewritability in monadic disjunctive datalog, MMSNP, and expressive description logics. CoRR abs/1701.02231 (2017), http://arxiv.org/abs/1701.02231

10. Gerasimova, O., Kikot, S., Podolskii, V., Zakharyaschev, M.: On the data complexity of ontology-mediated queries with a covering axiom. In: Proceedings of the 30th International Workshop on Description Logics (2017)
11. Gottlob, G., Papadimitriou, C.H.: On the complexity of single-rule datalog queries. Inf. Comput. 183(1), 104–122 (2003), http://dx.doi.org/10.1016/S0890-5401(03)00012-9
12. Grau, B.C., Motik, B., Stoilos, G., Horrocks, I.: Computing datalog rewritings beyond Horn ontologies. In: Rossi, F. (ed.) IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013. pp. 832–838. IJCAI/AAAI (2013), http://www.aaai.org/ocs/index.php/IJCAI/IJCAI13/paper/view/6318
13. 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
14. 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
15. Kontchakov, R., Rodriguez-Muro, M., Zakharyaschev, M.: Ontology-based data access with databases: A short course. In: Reasoning Web. Semantic Technologies for Intelligent Data Access - 9th International Summer School 2013, Mannheim, Germany, July 30 - August 2, 2013. Proceedings. pp. 194–229 (2013), https://doi.org/10.1007/978-3-642-39784-4_5
16. Kontchakov, R., Zakharyaschev, M.: An introduction to description logics and query rewriting. In: Reasoning Web. Reasoning on the Web in the Big Data Era - 10th International Summer School 2014, Athens, Greece, September 8-13, 2014. Proceedings. pp. 195–244 (2014), https://doi.org/10.1007/978-3-319-10587-1_5
17. Krisnadhi, A., Lutz, C.: Data complexity in the *EL* family of description logics. In: Logic for Programming, Artificial Intelligence, and Reasoning, 14th International Conference, LPAR 2007, Yerevan, Armenia, October 15-19, 2007, Proceedings. pp. 333–347 (2007), https://doi.org/10.1007/978-3-540-75560-9_25
18. Lutz, C., Sabellek, L.: Ontology-mediated querying with $\mathcal{EL}$: Trichotomy and linear datalog rewritabvility. In: Proceedings of the 30th International Workshop on Description Logics (2017)
19. 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)
20. Rosati, R.: The limits of querying ontologies. In: Database Theory - ICDT 2007, 11th International Conference, Barcelona, Spain, January 10-12, 2007, Proceedings. pp. 164–178 (2007), https://doi.org/10.1007/11965893_12
21. 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)