Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning (KR 2020)
Main Track

Explanations for Negative Query Answers under Existential Rules

İsmail İlkan Ceylan[1] , **Thomas Lukasiewicz**[1] , **Enrico Malizia**[2] , **Cristian Molinaro**[3] , **Andrius Vaicenavičius**[1]
[1]University of Oxford, Department of Computer Science, UK
[2]King's College London, Department of Informatics, UK
[3]University of Calabria, DIMES Department, Italy
{ismail.ceylan, thomas.lukasiewicz, andrius.vaicenavicius}@cs.ox.ac.uk,
enrico.malizia@kcl.ac.uk, cmolinaro@dimes.unical.it

Abstract

Ontology-mediated query answering is an extensively studied paradigm, where the conceptual knowledge provided by an ontology is leveraged towards more enhanced querying of data sources. A major advantage of ontological reasoning is its interpretability, which allows one to derive explanations for query answers. Indeed, explanations have a long history in knowledge representation, and have also been investigated for ontology languages based on description logics and existential rules. Existing works on existential rules, however, merely focus on understanding why a query is entailed, i.e., explaining positive query answers. In this paper, we continue this line of research and address another important problem, namely, explaining why a query is *not* entailed under existential rules, i.e., explaining negative query answers. We consider various problems related to explaining non-entailments from the abduction literature, and also introduce new problems. For all considered problems, we give a detailed complexity analysis for a wide range of existential rule languages and complexity measures.

1 Introduction

Ontology-based query answering aims to enhance querying of data sources with an ontology that encodes domain knowledge. The idea is to view the ontology and the user query as a composite query, called *ontology-mediated query (OMQ)*, and the task of evaluating such queries is then called *ontology-mediated query answering (OMQA)* (Bienvenu et al. 2014). OMQA is an important paradigm in knowledge representation with many application areas. Description logics (DLs) (Baader et al. 2007) and existential rules (Calì, Gottlob, and Kifer 2013; Calì, Gottlob, and Lukasiewicz 2012) are two families of languages, which are commonly used to formulate ontologies.

With the increasing demand for more explainable and interpretable systems, providing explanations for OMQA has recently seen a surge in interest. The most basic problem is that of explaining why a query is entailed, i.e., explaining *positive* query answers. This problem has been studied for ontology languages based on DLs (Borgida, Calvanese, and Rodriguez-Muro 2008; Ceylan et al. 2020) and existential rules (Ceylan et al. 2019). The main idea is to view an explanation as a set of facts from the database, which together with the ontology are sufficient to entail the query.

The complementary problem of explaining why a query is *not* entailed, i.e., explaining *negative* query answers, has only been studied in the context of DLs (Calvanese et al. 2013), where the problem is modeled as an abduction task. Intuitively, a (minimal) explanation for a non-entailment can be seen as a (minimal) set of facts such that, if the database were extended with this set, the query would be entailed, while the knowledge base would remain consistent with the added information. This is a natural choice for explaining non-entailments of monotonic queries, as a non-entailment can be attributed to missing facts.

In this paper, we continue this line of research, and address the problem of explaining negative answers in OMQA based on existential rules (rather than DLs) as underlying ontology languages. While DLs are popular formalisms for modeling ontologies, it is generally agreed that existential rules are well-suited for data-intensive applications, since they allow us to conveniently deal with higher-arity relations, which naturally occur in standard relational databases.

We consider the following computational problems of explaining negative query answers in OMQA with existential rules: (i) deciding whether a given set of facts is a minimal explanation, (ii) deciding whether there exists a minimal explanation at all, (iii) deciding whether a given fact is relevant or necessary, and (iv) deciding whether a given set of facts contains exactly all relevant or exactly all necessary facts. While the problems in (i) to (iii) have been considered before for DLs (Calvanese et al. 2013), they have not been considered yet for existential rules, and the problems in (iv) are considered for the first time in this paper. We give an intuitive example to illustrate the above concepts.

Example 1. We encode the functionality of a mechanical system that may experience certain issues as an OMQA problem. The system consists of four parts (p_1, p_2, p_3, and p_4), and each may cause some issues (among $overheat$, $leakage$, $high\_pressure$, non_start, $burnt\_smell$, $flow\_blockage$, and $pipes\_rattling$) when faulty. For example, (1) if p_1 is faulty, then the issues $overheat$ and $leakage$ may occur, (2) if p_2 is faulty, then the issues $high\_pressure$ and non_start may occur, and (3) if p_3 is faulty, then the issues $leakage$ and $burnt\_smell$ may occur. This functionality of the system is encoded via facts in a database D and ontological axioms (i.e., existential rules) in a program Σ.

When issues occur in the system, we would like to under-

stand which parts may be causing the issues, so that we can restrict the search for the causes to a limited number of parts. This is a problem of *fault diagnosis*, which can be modeled via explanations of negative query answers as follows.

In our example, the issues observed in the mechanical system (e.g., $leakage$ and non_start) can be encoded as queries. Such queries are not entailed by the database D alone. However, by adding to D facts about the potential faultiness of the four parts (forming the set of *hypotheses* H), the query can be derived, and hence an explanation built from H can be interpreted as a possible set of causes for the observed issues.

For example, under the above specification, the observed issues $leakage$ and non_start have the *explanations* of (a) p_1 and p_2 being faulty or (b) p_2 and p_3 being faulty. These are also *minimal* explanations, and they are the only minimal ones. Hence, p_1, p_2, and p_3 are all *relevant* facts (which belong to some minimal explanation), while p_2 is the only *necessary* fact (which is in every minimal explanation).

The precise definitions and details for this example will be given later in the paper. ◁

In this paper, we provide a precise picture of the complexity of the following computational tasks around explanations for negative answers in OMQA for a wide range of existential rule languages and under different complexity measures:

- Deciding the existence of a minimal explanation, and deciding whether a given set of facts is an explanation.
- Deciding whether a given fact is relevant (i.e., belongs to at least one minimal explanation), and deciding whether a given set of facts contains exactly all relevant facts.
- Deciding whether a given fact is necessary (i.e., belongs to every minimal explanation), and deciding whether a given set of facts contains exactly all necessary facts.

2 Preliminaries

In this section, we recall some basics on existential rules (Calì, Gottlob, and Pieris 2012; Calì, Gottlob, and Kifer 2013; Calì, Gottlob, and Lukasiewicz 2012) and the paradigm of ontology-mediated query answering. We also give some complexity-theoretic background relevant to our study.

2.1 General

We consider a relational vocabulary consisting of mutually disjoint, possibly infinite sets $\mathbf{R}$, $\mathbf{C}$, $\mathbf{N}$, and $\mathbf{V}$ of *predicates*, *constants*, *nulls*, and *variables*, respectively. Each predicate is associated with an arity, i.e., a non-negative integer. A *term* is a constant, a null, or a variable. An *atom* is an expression of the form $P(t_1, \ldots, t_n)$, where P is an n-ary predicate, and $t_1, \ldots, t_n$ are terms. A *ground atom* (or *fact*) has only constants as terms. Conjunctions of atoms are often identified with the sets of their atoms.

An *instance* I is a (possibly infinite) set of atoms containing constants and nulls only. A *database* D is a finite instance that contains only constants. A *homomorphism* is a mapping $h\colon \mathbf{C} \cup \mathbf{N} \cup \mathbf{V} \to \mathbf{C} \cup \mathbf{N} \cup \mathbf{V}$ that is the identity on $\mathbf{C}$ and maps $\mathbf{N}$ to $\mathbf{C} \cup \mathbf{N}$. With a slight abuse of notation, homomorphisms are applied also to (sets of) atoms.

A *conjunctive query (CQ)* $Q(\mathbf{X})$ is a first-order formula of the form $\exists \mathbf{Y} \phi(\mathbf{X}, \mathbf{Y})$, where $\phi(\mathbf{X}, \mathbf{Y})$ is a conjunction of atoms without nulls. The *answer* to $Q(\mathbf{X})$ over an instance I, denoted $Q(I)$, is the set of all tuples $\mathbf{t} \in \mathbf{C}^{|\mathbf{X}|}$ for which there is a homomorphism h such that $h(\phi(\mathbf{X}, \mathbf{Y})) \subseteq I$ and $h(\mathbf{X}) = \mathbf{t}$. A *union of conjunctive queries (UCQ)* $Q(\mathbf{X})$ has the form $Q_1(\mathbf{X}) \vee \cdots \vee Q_n(\mathbf{X})$, where each $Q_i(\mathbf{X})$ is a CQ. The evaluation of $Q(\mathbf{X})$ over an instance I, denoted $Q(I)$, is defined as the set of tuples $\bigcup_{1 \leq i \leq n} Q_i(I)$. A *Boolean* UCQ Q is a UCQ where all variables are existentially quantified. We say that Q is *true* over I, denoted $I \models Q$, if $Q(I) \neq \emptyset$. In the rest of the paper, we only focus on Boolean UCQs.

2.2 Existential Rules

A *tuple-generating dependency (TGD)* σ is a first-order formula of the form $\forall \mathbf{X} \forall \mathbf{Y}\, \Phi(\mathbf{X}, \mathbf{Y}) \rightarrow \exists \mathbf{Z}\, \Psi(\mathbf{X}, \mathbf{Z})$, where $\Phi(\mathbf{X}, \mathbf{Y})$ is a conjunction of atoms, called the *body* of the TGD and denoted $body(\sigma)$, and $\Psi(\mathbf{X}, \mathbf{Z})$ is a conjunction of atoms, called the *head* of the TGD and denoted $head(\sigma)$, all without nulls. Classes of TGDs are also known as *existential rules*, or *Datalog$^{\pm}$ languages* in the literature. An instance I satisfies σ, written $I \models \sigma$, if the following holds: whenever there exists a homomorphism h such that $h(\Phi(\mathbf{X}, \mathbf{Y})) \subseteq I$, then there exists $h' \supseteq h|_{\mathbf{X}}$, where $h|_{\mathbf{X}}$ is the restriction of h on $\mathbf{X}$, such that $h'(\Psi(\mathbf{X}, \mathbf{Z})) \subseteq I$.

A *negative constraint (NC)* ν is a first-order formula of the form $\forall \mathbf{X}\, \Phi(\mathbf{X}) \rightarrow \bot$, where $\Phi(\mathbf{X})$ is a conjunction of atoms without nulls, called the *body* of ν and denoted $body(\nu)$, and $\bot$ denotes the truth constant $false$. An instance I satisfies ν, written $I \models \nu$, if there is no homomorphism h such that $h(\Phi(\mathbf{X})) \subseteq I$.

A *program* (or *ontology*) is a finite set Σ of TGDs and NCs. We denote by Σ_T and Σ_{NC} the subsets of Σ containing the TGDs and NCs of Σ, respectively. An instance I satisfies Σ, written $I \models \Sigma$, if I satisfies each TGD and NC of Σ. For brevity, we omit the universal quantifiers in front of TGDs and NCs, and use the comma (instead of $\wedge$) for conjoining atoms. Given a class of TGDs $\mathbb{C}$, we denote by $\mathbb{C}_{\bot}$ the formalism obtained by combining $\mathbb{C}$ with arbitrary NCs. For a program Σ, $\Sigma \in \mathbb{C}$ (resp., $\Sigma \in \mathbb{C}_{\bot}$) denotes that all TGDs of Σ belong to $\mathbb{C}$.

The Datalog$^{\pm}$ languages that we consider to guarantee decidability are among the most frequently analyzed in the literature, namely, linear (L) (Calì, Gottlob, and Lukasiewicz 2012), guarded (G) (Calì, Gottlob, and Kifer 2013), sticky (S) (Calì, Gottlob, and Pieris 2012), and acyclic TGDs (A), along with the "weak" (proper) generalizations weakly sticky (WS) (Calì, Gottlob, and Pieris 2012) and weakly acyclic TGDs (WA) (Fagin et al. 2005), as well as their "full" (i.e., existential-free) proper restrictions linear full (LF), guarded full (GF), sticky full (SF), and acyclic full TGDs (AF), respectively, and full TGDs (F) in general. We also recall the following further inclusions: $\mathsf{L} \subset \mathsf{G}$ and $\mathsf{F} \subset \mathsf{WA} \subset \mathsf{WS}$.

2.3 Ontology-Mediated Query Answering

The paradigm of ontology-mediated query answering generalizes query answering over databases by incorporating additional background knowledge in terms of an ontology.

$\mathcal{L}$	**Data**	fp**-comb.**	ba**-comb.**	**Comb.**
L, LF, AF	$\leq$ AC^0	NP	NP	PSPACE
S, SF	$\leq$ AC^0	NP	NP	EXP
A	$\leq$ AC^0	NP	NEXP	NEXP
G	P	NP	EXP	2EXP
F, GF	P	NP	NP	EXP
WS, WA	P	NP	2EXP	2EXP

Table 1: Complexity of OMQA under existential rules.

Formally, an *ontology-mediated query (OMQ)* is a pair (Q, Σ), where Q is a query, and Σ is an ontology. Let $\mathcal{L}$ be an ontology language. If $\Sigma \in \mathcal{L}$, then we say that (Q, Σ) is an $\mathcal{L}$-OMQ. Consider a database D and an OMQ (Q, Σ). The set of *models* of (D, Σ), denoted $mods(D, \Sigma)$, is the set of instances $\{I \mid I \supseteq D \wedge I \models \Sigma\}$. We say that D *entails* (Q, Σ), denoted $D \models (Q, \Sigma)$, if $I \models Q$ for every $I \in mods(D, \Sigma)$. *Ontology-mediated query answering (OMQA)* is the task of deciding whether $D \models (Q, \Sigma)$ for a given database D and an OMQ (Q, Σ). Table 1 summarizes the known complexity results for OMQA in the different classes of TGDs that we consider. We denote by OMQA($\mathcal{L}$) the problem of OMQA when restricted over ontologies belonging to $\mathcal{L}$. We say that (D, Σ) is *consistent* if $mods(D, \Sigma) \neq \emptyset$, otherwise (D, Σ) is *inconsistent*. When OMQA($\mathcal{L}$) is restricted to the case where (D, Σ) is consistent, we talk of consistent-OMQA($\mathcal{L}$).

A key paradigm in OMQA is the *FO-rewritability* of queries: an OMQ (Q, Σ) is *FO-rewritable*, if there exists a query Q_Σ such that, for all databases D that are consistent relative to Σ, we have that $D \models (Q, \Sigma)$ iff $D \models Q_\Sigma$. In this case, Q_Σ is called an *FO-rewriting* of (Q, Σ). A class of programs $\mathcal{L}$ is *FO-rewritable*, if it admits an FO-rewriting for every query and program in $\mathcal{L}$. All languages from Table 1 with AC^0 data complexity are FO-rewritable.

2.4 Computational Complexity

In our complexity analysis, we make the standard assumptions (Vardi 1982): the *combined complexity* of query answering is calculated by considering all the components (i.e., the database, the program, and the query) as part of the input. The *bounded-arity combined complexity* (or simply *ba-combined complexity*) assumes that the maximum arity of the predicates in $\mathbf{R}$ is bounded by a constant integer. The *fixed-program combined complexity* (or simply *fp-combined complexity*) is calculated by considering the ontology as fixed. Finally, the *data complexity* is calculated by considering the database as the input, i.e., everything else is fixed. This paper's most relevant complexity classes and their relations are as follows:

AC$^0 \subseteq$ P $\subseteq$ NP, coNP $\subseteq$ D$^{\text{P}} \subseteq \Sigma_2^{\text{P}}, \Pi_2^{\text{P}} \subseteq$ D$_2^{\text{P}} \subseteq$ PSPACE $\subseteq$ EXP $\subseteq$ NEXP, coNEXP $\subseteq$ D$^{\text{EXP}} \subseteq$ P$^{\text{NEXP}} \subseteq$ 2EXP.

3 Explanations for Negative Query Answers

In this section, we formally define explanations and minimal explanations for negative query answers in OMQA along with several important computational problems for them.

Definition 2. Let D be a database, let (Q, Σ) be an OMQ, with $D \not\models (Q, \Sigma)$, and let H be a finite set of facts. An *explanation* for $D \not\models (Q, \Sigma)$ w.r.t. H is a subset E of H such that $(D \cup E, \Sigma)$ is consistent and $D \cup E \models (Q, \Sigma)$. A *minimal explanation* (or *MinEX*) for $D \not\models (Q, \Sigma)$ w.r.t. H is an explanation E for $D \not\models (Q, \Sigma)$ w.r.t. H that is inclusion-minimal, i.e., no set $E' \subsetneq E$ is an explanation for $D \not\models (Q, \Sigma)$ w.r.t. H.

Notice that, in order for $D \not\models (Q, \Sigma)$ to hold, (D, Σ) must be consistent.

Example 3. Consider again the mechanical system in Example 1. We can encode the fault diagnosis problem as follows. In the database D, we store the relationships between the faulty parts and the caused issues. The binary predicate $FaultIssue(X, Y)$ describes that when the part X is faulty, we may observe the issue Y. For example, $FaultIssue(p_1, overheat)$ states that when p_1 is faulty, we may observe an overheat of the system. Consider the following database:

$$D = \{FaultIssue(p_1, overheat), FaultIssue(p_1, leakage), FaultIssue(p_2, high\_pressure), FaultIssue(p_2, non\_start), FaultIssue(p_3, leakage), FaultIssue(p_3, burnt\_smell), FaultIssue(p_4, flow\_blockage)\}.$$

The set H of the possible causes of the observed issues is:

$$H = \{Fault(p_1), Fault(p_2), Fault(p_3), Fault(p_4)\}.$$

OMQs (Q, Σ) then encode the issues that we observe (in Q) and our additional knowledge about the functioning of the system (in Σ). Consider the following program:

$$\Sigma = \{Fault(p_1), Fault(p_4) \rightarrow \bot; Fault(p_1) \rightarrow Fault(p_3); Fault(p_1), Fault(p_2) \rightarrow Issue(pipes\_rattling); FaultIssue(X, Y), Fault(X) \rightarrow Issue(Y)\}.$$

The first rule in Σ (an NC) says that the parts p_1 and p_4 cannot be faulty at the same time. The second one encodes that if p_1 is faulty, then so is p_3. The third one says that when p_1 and p_2 are both faulty, then we observe rattling pipes. The last one encodes that if X is a part causing the issue Y, and X is faulty, then we observe the issue Y.

Consider now the following OMQ:

$$(Q = Issue(leakage) \wedge Issue(non\_start), \Sigma),$$

where Σ is the program defined above. In the query, we state that we observe a leakage and that the system does not start. The OMQ is not entailed by the database D alone. However, by adding to D facts taken from H, the OMQ can be derived, and hence an explanation built from H can be interpreted as a possible set of causes for the issues observed. $\lhd$

The first computational problem for explanations for negative query answers in OMQA is deciding whether a set of facts is a minimal explanation for a negative answer.

Problem: IS-MINEX$^{\not\models}(\mathcal{L})$.
Input: A database D, an $\mathcal{L}$-OMQ (Q, Σ), where $D \not\models (Q, \Sigma)$, a finite set of facts H, and $E \subseteq H$.
Question: Is E a MinEX for $D \not\models (Q, \Sigma)$ w.r.t. H?

Example 4. Consider again the database D, the program Σ, and the set H of Example 3. Then, for the OMQ

$$(Q = \mathit{Issue}(\mathit{burnt\_smell}) \wedge \mathit{Issue}(\mathit{leakage}), \Sigma),$$

the sets $E_1 = \{\mathit{Fault}(p_1)\}$ and $E_2 = \{\mathit{Fault}(p_3)\}$ are minimal explanations, while $E' = \{\mathit{Fault}(p_1), \mathit{Fault}(p_3)\}$ is an explanation, but not a minimal one. $\lhd$

Another computational problem is deciding whether there exists a minimal explanation at all for a given OMQ.

Problem: MINEX-EXISTS$^{\not\models}(\mathcal{L})$.
Input: A database D, an $\mathcal{L}$-OMQ (Q, Σ), where $D \not\models (Q, \Sigma)$, and a finite set of facts H.
Question: Is there a MinEX for $D \not\models (Q, \Sigma)$ w.r.t. H?

Example 5. Consider again the database D, the program Σ, and the set H of Example 3. Then, for the OMQ

$$(Q = \mathit{Issue}(\mathit{flow\_blockage}) \wedge \mathit{Issue}(\mathit{burnt\_smell}), \Sigma),$$

the set $E = \{\mathit{Fault}(p_3), \mathit{Fault}(p_4)\}$ is an explanation, but not $E' = \{\mathit{Fault}(p_1), \mathit{Fault}(p_4)\}$, as the simultaneous presence of $\mathit{Fault}(p_1)$ and $\mathit{Fault}(p_4)$ is not admitted by the NC. On the other hand, for the OMQ

$$(Q = \mathit{Issue}(\mathit{flow\_blockage}) \wedge \mathit{Issue}(\mathit{overheat}), \Sigma),$$

there is no explanation. $\lhd$

Two other computational problems are recognizing relevant facts and recognizing necessary facts. A fact ψ is *relevant* (resp., *necessary*) for $D \not\models (Q, \Sigma)$ w.r.t. H iff ψ appears in at least one (resp., in every) MinEX for $D \not\models (Q, \Sigma)$ w.r.t. H.

Problem: MINEX-REL$^{\not\models}(\mathcal{L})$.
Input: A database D, an $\mathcal{L}$-OMQ (Q, Σ), where $D \not\models (Q, \Sigma)$, a finite set of facts H, and a fact ψ.
Question: Is ψ relevant for $D \not\models (Q, \Sigma)$ w.r.t. H?

Problem: MINEX-NEC$^{\not\models}(\mathcal{L})$.
Input: A database D, an $\mathcal{L}$-OMQ (Q, Σ), where $D \not\models (Q, \Sigma)$, a finite set of facts H, and a fact ψ.
Question: Is ψ necessary for $D \not\models (Q, \Sigma)$ w.r.t. H?

Example 6. Consider again the database D, the program Σ, and the set H of Example 3. Then, for the OMQ

$$(Q = \mathit{Issue}(\mathit{non\_start}) \wedge \mathit{Issue}(\mathit{leakage}), \Sigma),$$

the sets $E_1 = \{\mathit{Fault}(p_1), \mathit{Fault}(p_2)\}$ and $E_2 = \{\mathit{Fault}(p_2), \mathit{Fault}(p_3)\}$ are all the minimal explanations for the OMQ. Hence, the facts $\mathit{Fault}(p_1)$, $\mathit{Fault}(p_2)$, and $\mathit{Fault}(p_3)$ are relevant, while the only necessary fact is $\mathit{Fault}(p_3)$. $\lhd$

The computational problems introduced so far are those commonly addressed in the context of abductive reasoning and negative answer explanations. In addition, we introduce two novel important problems. The first one asks whether a set H' of facts contains exactly all the relevant facts, i.e., H' is the union of all MinEXs. The problem is particularly interesting, as H' can be seen as a minimal over-approximation of all MinEXs, i.e., for every MinEX E, it holds that $E \subseteq H'$, and H' is the smallest set enjoying this property.

Problem: MINEX-ALLREL$^{\not\models}(\mathcal{L})$.
Input: A database D, an $\mathcal{L}$-OMQ (Q, Σ), where $D \not\models (Q, \Sigma)$, a finite set of facts H, and a set $H' \subseteq H$.
Question: Does H' contain exactly all the relevant facts for $D \not\models (Q, \Sigma)$ w.r.t. H?

The second novel computational problem that we consider asks whether a set H' contains exactly all the necessary facts, i.e., H' is the intersection of all MinEXs. Interestingly, H' can be seen as a maximal under-approximation of all MinEXs, i.e., for every MinEX E, it holds that $H' \subseteq E$, and H' is the biggest set enjoying this property.

Problem: MINEX-ALLNEC$^{\not\models}(\mathcal{L})$.
Input: A database D, an $\mathcal{L}$-OMQ (Q, Σ), where $D \not\models (Q, \Sigma)$, a finite set of facts H, and a set $H' \subseteq H$.
Question: Does H' contain exactly all the necessary facts for $D \not\models (Q, \Sigma)$ w.r.t. H?

Example 7. For the OMQ in Example 6

$$(Q = \mathit{Issue}(\mathit{non\_start}) \wedge \mathit{Issue}(\mathit{leakage}), \Sigma),$$

the sets of all relevant and necessary facts are $\{\mathit{Fault}(p_1), \mathit{Fault}(p_2), \mathit{Fault}(p_3)\}$ and $\{\mathit{Fault}(p_3)\}$, respectively. $\lhd$

4 Positive vs. Negative Explanations

Explaining positive query answers, i.e., why a database D entails an OMQ (Q, Σ), where Σ is a set of TGDs, has been recently investigated by Ceylan et al. (2019). Specifically, an *explanation* for $D \models (Q, \Sigma)$ is a set $E \subseteq D$ such that $E \models (Q, \Sigma)$. A *minimal explanation* for $D \models (Q, \Sigma)$ is an inclusion-minimal explanation for $D \models (Q, \Sigma)$.

In this paper, we deal with "negative" explanations. One may nevertheless wonder whether there are easy reductions mapping one problem into the other and covering the problems that are considered. It is possible, as we show, to leverage a reduction from the positive to the negative setting to derive the lower bounds for the problems IS-MINEX$^{\not\models}$ and MINEX-REL$^{\not\models}$. For the remaining problems, however, there is no straightforward correspondence between positive and negative explanation problems, and hence, they required novel proofs.

One important difference between the positive and negative setting is that positive explanations are subsets of the database, while negative explanations are subsets of a set H of abducible facts and must be added to the entire database. This makes simple reductions from the negative to the positive setting fail. As an example, suppose in our setting that we are given a database D, an OMQ (Q, Σ), and a finite set of facts H. One might try to derive an instance of the positive explanation problem where the database is $D \cup H$ and the OMQ remains (Q, Σ). However, a set E that is a minimal explanation for $D \cup H \models (Q, \Sigma)$ might not be a minimal explanation for $D \not\models (Q, \Sigma)$ w.r.t. H (because E is not minimal or it is not an explanation at all). As an example, consider the database $D = \{R(a)\}$, the OMQ (Q, Σ) with $Q = \exists X, Y\, R(X), S(Y)$ and $\Sigma = \emptyset$, and the set of abducible facts $H = \{R(b), S(b)\}$. Then, $E = \{R(b), S(b)\}$ is a minimal explanation for $D \cup H \models (Q, \Sigma)$, but it is not a minimal explanation for $D \not\models (Q, \Sigma)$ w.r.t. H (because it is not minimal).

5 Overview of Complexity Results

We give a precise picture of the complexity for all the explanation problems. An overview of our complexity results is given in Tables 2–7. They range from membership in P to 2EXP-completeness; all entries without '$\leq$' are completeness results. For all problems and complexity measures, H (as well as E in IS-MINEX$^{\models}$, ψ in MINEX-REL$^{\models}$and MINEX-NEC$^{\models}$, and H' in MINEX-ALLREL$^{\models}$and MINEX-ALLNEC$^{\models}$) is always part of the input, that is, it is treated like the database.

MINEX-REL$^{\models}$ and MINEX-NEC$^{\models}$ always have complementary complexity classes, except in the *fp*-combined complexity, where MINEX-REL$^{\models}$ is Σ_2^{P}-complete, while MINEX-NEC$^{\models}$ is coNP-complete. Intuitively, the reason is that to decide whether a fact ψ is *not* necessary, we can guess a subset E of H and check whether E is an explanation that does not include ψ, and verifying minimality of E can be avoided. In contrast, deciding relevance requires verifying minimality of a guessed set, which in turn requires verifying *non*-entailment. Such a check is in coNP in the *fp*-combined complexity and yields a membership in $\text{NP}^{\text{coNP}} = \Sigma_2^{\text{P}}$. This also explains the difference between MINEX-ALLREL$^{\models}$ and MINEX-ALLNEC$^{\models}$in the *fp*-combined complexity.

The discussion above is also related to why MINEX-EXISTS$^{\models}$ has a lower complexity than MINEX-REL$^{\models}$in the *fp*-combined complexity: a minimal explanation exists iff an explanation (not necessarily minimal) exists; we can then avoid verifying minimality of a guessed subset of H.

For the linear, full, and sticky languages, as well as for their sublanguages, OMQA is NP-complete both in the *fp*- and in the ba-combined complexity. One may expect that, for a fixed problem, the complexity does not change in such cases, as the complexity of OMQA remains the same across them. However, this is not the case for MINEX-EXISTS$^{\models}$, MINEX-NEC$^{\models}$, and MINEX-ALLNEC$^{\models}$, where the complexity goes from NP-, coNP-, and D^{P}-completeness in the *fp*-combined complexity to Σ_2^{P}-, Π_2^{P}-, and D_2^{P}-completeness in the ba-combined complexity, respectively. The reason is that for such problems, consistency checking can be done in P in the *fp*-combined complexity, as the program is fixed, and it is not required to check minimality of a guessed set. In contrast, in the ba-combined complexity, the program is not fixed anymore, and checking consistency requires a linear number of checks for *non*-entailment, whose cost increases the overall complexity by one level.

We analyze the complexity of IS-MINEX$^{\models}$ and MINEX-EXISTS$^{\models}$ in Section 6, of MINEX-REL$^{\models}$ and MINEX-ALLREL$^{\models}$ in Section 7, and of MINEX-NEC$^{\models}$ and MINEX-ALLNEC$^{\models}$ in Section 8. For each problem, we first discuss membership and then hardness results. We first prove general procedures for the upper bounds that apply to all languages and complexity measures, and we separately show tighter upper bounds, when required. As an example, the data complexity of all problems for FO-rewritable languages is in P. In many cases, this is a tighter upper bound than the one provided by the general procedure, and it is derived by leveraging rewritability into unions of conjunctive queries. Hardness results are stated for the most specific languages that they apply to (and then hold for all generalizations, of course).

$\mathcal{L}$	**Data**	*fp*-**comb.**	ba-**comb.**	**Comb.**
$\mathsf{L}_\bot$, $\mathsf{LF}_\bot$, $\mathsf{AF}_\bot$	$\leq$ P	D^{P}	D^{P}	PSPACE
$\mathsf{S}_\bot$, $\mathsf{SF}_\bot$	$\leq$ P	D^{P}	D^{P}	EXP
$\mathsf{A}_\bot$	$\leq$ P	D^{P}	D^{EXP}	D^{EXP}
$\mathsf{G}_\bot$	P	D^{P}	EXP	2EXP
$\mathsf{F}_\bot$, $\mathsf{GF}_\bot$	P	D^{P}	D^{P}	EXP
$\mathsf{WS}_\bot$, $\mathsf{WA}_\bot$	P	D^{P}	2EXP	2EXP

Table 2: Complexity of IS-MINEX$^{\models}(\mathcal{L})$.

6 IS-MINEX$^{\models}$and MINEX-EXISTS$^{\models}$

We now analyze the complexity of recognizing a MinEX and deciding the existence of a MinEX for a given database and an OMQ. We start with IS-MINEX$^{\models}(\mathcal{L})$, i.e., deciding whether a given set of facts is a minimal explanation for a negative query answer, and the membership results. The following theorem proves all upper bounds in Table 2.

Theorem 8. *For any language $\mathcal{L}$ considered in this paper, if OMQA($\mathcal{L}$) is in the complexity class $\mathcal{C}$ in the combined (resp., ba-combined, fp-combined, data) complexity, then* IS-MINEX$^{\models}$*($\mathcal{L}$) can be decided with a $\mathcal{C}$ check and a linear number of co-$\mathcal{C}$ checks in the combined (resp., ba-combined, fp-combined, data) complexity.*

Proof sketch. Deciding whether a given set E of facts is a MinEX requires to carry out essentially three tasks: (1) deciding whether $(D \cup E, \Sigma)$ is consistent; (2) deciding whether $D \cup E \models (Q, \Sigma)$; and (3) deciding whether E is inclusion-minimal. Task (1) can be solved by checking that $D \cup E$ does not entail the body of any negative constraint, which hence consists of a linear number of *non*-entailment tests. Task (2) can be carried out by performing a single entailment test. Task (3) can be decided by checking that all facts in E are required to entail (Q, Σ), and this can be done by verifying that removing every single fact ψ from E in turn leads to a set of facts not entailing (Q, Σ). Also this task consists of a linear number of *non*-entailment tests. □

We now focus on the hardness results for IS-MINEX$^{\models}(\mathcal{L})$. Among others, Ceylan et al. (2019) studied the problem of deciding whether a set E is a minimal explanation for $D \models (Q, \Sigma)$, called IS-MINEX($\mathcal{L}$). All the lower bounds in Table 2 are obtained from the following result that establishes a connection between "positive" and "negative" explanations.

Lemma 9. *For any language $\mathcal{L}$ considered in this paper,* IS-MINEX*($\mathcal{L}$) is reducible in polynomial time to* IS-MINEX$^{\models}$*($\mathcal{L}$) in the data, fp-combined, ba-combined, and combined complexity.*

We now focus on the problem MINEX-EXISTS$^{\models}(\mathcal{L})$ of deciding the existence of (minimal) explanations for negative query answers. We start by providing the membership results. The following theorem proves all the upper bounds in Table 3 but the NP and P ones, for which we need tighter results stated next. Intuitively, to decide whether there exists a minimal explanation for a negative query answer, it suffices

$\mathcal{L}$	**Data**	*fp*-**comb.**	*ba*-**comb.**	**Comb.**
$\mathsf{L}_\bot$, $\mathsf{LF}_\bot$, $\mathsf{AF}_\bot$	$\leq$ P	NP	Σ_2^{P}	PSPACE
$\mathsf{S}_\bot$, $\mathsf{SF}_\bot$	$\leq$ P	NP	Σ_2^{P}	EXP
$\mathsf{A}_\bot$	$\leq$ P	NP	P^{NEXP}	P^{NEXP}
$\mathsf{G}_\bot$	NP	NP	EXP	2EXP
$\mathsf{F}_\bot$, $\mathsf{GF}_\bot$	NP	NP	Σ_2^{P}	EXP
$\mathsf{WS}_\bot$, $\mathsf{WA}_\bot$	NP	NP	2EXP	2EXP

Table 3: Complexity of MINEX-EXISTS$^{\models}(\mathcal{L})$.

to check whether there is any explanations for the negative query answers, i.e., there is no need to double check the minimality. Hence, we guess a set $E \subseteq H$ of facts (feasible in NP), and then check via oracle calls that E is consistent and allows us to entail the OMQ (Tasks (1) and (2) in the proof sketch of Theorem 8).

Theorem 10. *For any language $\mathcal{L}$ considered here, if OMQA($\mathcal{L}$) is in the complexity class $\mathcal{C}$ in the combined (resp., ba-combined, fp-combined, data) complexity, then* MINEX-EXISTS$^{\models}(\mathcal{L})$ *is in* $\text{NP}^{\mathcal{C}}$ *in the combined (resp., ba-combined, fp-combined, data) complexity.*

The following theorem provides the NP upper bounds in the *fp*-combined complexity in Table 3. The result is obtained from the proof of Theorem 8 with the additional observation that in the *fp*-combined setting, for the Datalog$^{\pm}$ languages considered, checking whether a set of facts is consistent is feasible in P, because the negative constraints are fixed.

Theorem 11. MINEX-EXISTS$^{\models}(\mathcal{L})$ *is in* NP *in the fp-combined complexity for all languages $\mathcal{L}$ considered here.*

We conclude the discussion of the membership results for MINEX-EXISTS$^{\models}$ by showing that the problem is in P in the data complexity for all the FO-rewritable languages considered, namely, L, S, and A (as well as their full restrictions). The proof relies on the fact that, for these languages, any OMQ can be rewritten into an equivalent FO query Q, which is a union of conjunctive queries. Starting from the values of the tuples in the database and H, it is possible to build in polynomial time the set $\mathcal{D}$ containing all the polynomially-many sets of facts that can possibly satisfy Q. Once $\mathcal{D}$ is computed, all its elements can be tested in turn, and we verify (in polynomial time) whether any of them is a (non-necessarily minimal) explanation for the negative answer of the OMQ.

Theorem 12. *If $\mathcal{L}$ is FO-rewritable language, then* MINEX-EXISTS$^{\models}(\mathcal{L})$ *is in* P *in the data complexity.*

As for the hardness results for MINEX-EXISTS$^{\models}(\mathcal{L})$, the following theorem proves the lower bounds in the data complexity in Table 3 for the non-FO-rewritable languages, namely, $\mathsf{GF}_\bot$ and its generalizations. Its proof is a reduction from the satisfiability of 3CNF Boolean formulas $\phi(X)$. Intuitively, in a fixed program, there are rules deciding the satisfiability of $\phi(X)$. The set H contains facts associated with truth assignments for the variables X. A candidate explanation E encodes a truth assignment σ_E for the variables X, and E is actually an explanation iff σ_E satisfies $\phi(X)$.

Theorem 13. MINEX-EXISTS$^{\models}(\mathsf{GF}_\bot)$ *is* NP-hard *in the data complexity.*

The hardness results in Table 3 in the *fp*-combined, *ba*-combined, and combined complexity can mainly be obtained from the following lemma showing a link between the complexity of consistent-OMQA($\mathcal{L}$) and of MINEX-EXISTS$^{\models}(\mathcal{L})$. The Σ_2^{P}-hardness and the P^{NEXP}-hardness results in Table 3 require tighter theorems, which will be stated next.

Lemma 14. *For any language $\mathcal{L}$ considered here, consistent-OMQA($\mathcal{L}$) is reducible in polynomial time to* MINEX-EXISTS$^{\models}(\mathcal{L})$ *in the data, fp-combined, ba-combined, and combined complexity.*

The following theorem proves the Σ_2^{P}-hardness results (in the *ba*-combined complexity) in Table 3. The proof uses a reduction from the Σ_2^{P}-complete problem $\text{QBF}_{2,\forall}^{DNF}$: decide the validity of a quantified Boolean formula $\Phi = \exists X \forall Y \phi(X, Y)$, where $\phi(X, Y)$ is in 3DNF. We need to encode the validity of the quantified formula. In H, there are facts associated with the various truth assignments for the variables X. The test for the unsatisfiability of $\phi(X, Y)$, when a truth assignment for X is encoded in a candidate explanation, is captured with a negative constraint. A candidate explanation E encoding a truth assignment σ_E for X is actually an explanation iff $\forall Y \phi(X/\sigma_E, Y)$ is valid. So, if $\forall X \exists Y \neg\phi(X, Y)$ holds, then there are no explanations.

Theorem 15. MINEX-EXISTS$^{\models}(\mathcal{L})$ *is* Σ_2^{P}-hard *in the ba-combined complexity for all languages $\mathcal{L}$ considered here.*

The following theorem proves the two P^{NEXP}-hardness results in Table 3 for A in the *ba*-combined and combined complexity. The result can be shown via a reduction from the problem ETP defined by Eiter, Lukasiewicz, and Predoiu (2016): given a triple (m, TP_1, TP_2), where m is a number in unary, and TP_1 and TP_2 are two tiling problems for the exponential square $2^n \times 2^n$, decide whether there exists an initial tiling conditions w of length m such that TP_1 has a solution with w *and* TP_2 has no solution with w. The idea of the reduction is to have the candidate explanations taken from H to encode the possible initial tiling conditions. Then, there are rules that allow us to derive the query if TP_1 has a solution with the initial condition w encoded in the explanation, and TP_2 has no solution with w.

Theorem 16. MINEX-EXISTS$^{\models}(\mathsf{A}_\bot)$ *is* P^{NEXP}-hard *in the ba-combined and combined complexity.*

7 MINEX-REL$^{\models}$ and MINEX-ALLREL$^{\models}$

In this section, we deal with the problems regarding relevant facts, i.e., facts appearing in at least one minimal explanation. In particular, we analyze the problem of deciding whether a fact is relevant and the problem of deciding whether a set of facts is the set of all and only the relevant facts. We start by looking at the problem MINEX-REL$^{\models}(\mathcal{L})$ of deciding whether a fact is relevant or not, and first provide the membership results. The following theorem proves all the upper bounds in Table 4, except for those in the data complexity for FO-languages, for which we need a tighter statement. Intuitively, to decide whether ψ is relevant, it suffices to guess a set E of facts containing ψ (feasible in NP), and then, via an oracle call, check that E is a minimal explanation.

$\mathcal{L}$	**Data**	***fp*-comb.**	***ba*-comb.**	**Comb.**
$\mathsf{L}_\bot, \mathsf{LF}_\bot, \mathsf{AF}_\bot$	$\leq$ P	Σ_2^{P}	Σ_2^{P}	PSPACE
$\mathsf{S}_\bot, \mathsf{SF}_\bot$	$\leq$ P	Σ_2^{P}	Σ_2^{P}	EXP
$\mathsf{A}_\bot$	$\leq$ P	Σ_2^{P}	P^{NEXP}	P^{NEXP}
$\mathsf{G}_\bot$	NP	Σ_2^{P}	EXP	2EXP
$\mathsf{F}_\bot, \mathsf{GF}_\bot$	NP	Σ_2^{P}	Σ_2^{P}	EXP
$\mathsf{WS}_\bot, \mathsf{WA}_\bot$	NP	Σ_2^{P}	2EXP	2EXP

Table 4: Complexity of MINEX-REL$^{\models}(\mathcal{L})$.

$\mathcal{L}$	**Data**	***fp*-comb.**	***ba*-comb.**	**Comb.**
$\mathsf{L}_\bot, \mathsf{LF}_\bot, \mathsf{AF}_\bot$	$\leq$ P	D_2^{P}	D_2^{P}	PSPACE
$\mathsf{S}_\bot, \mathsf{SF}_\bot$	$\leq$ P	D_2^{P}	D_2^{P}	EXP
$\mathsf{A}_\bot$	$\leq$ P	D_2^{P}	P^{NEXP}	P^{NEXP}
$\mathsf{G}_\bot$	D^{P}	D_2^{P}	EXP	2EXP
$\mathsf{F}_\bot, \mathsf{GF}_\bot$	D^{P}	D_2^{P}	D_2^{P}	EXP
$\mathsf{WS}_\bot, \mathsf{WA}_\bot$	D^{P}	D_2^{P}	2EXP	2EXP

Table 5: Complexity of MINEX-ALLREL$^{\models}(\mathcal{L})$.

Theorem 17. *For any language $\mathcal{L}$ considered here, if* IS-MINEX$^{\models}(\mathcal{L})$ *is in the complexity class $\mathcal{C}$ in the combined (resp., ba-combined, fp-combined, data) complexity, then* MINEX-REL$^{\models}(\mathcal{L})$ *is in* $\text{NP}^{\mathcal{C}}$ *in the combined (resp., ba-combined, fp-combined, data) complexity.*

For the FO-rewritable languages, the following theorem proves the P-membership results in the data complexity shown in Table 4 that are not obtained via Theorem 17. To prove this result, we exploit, also in this case, the rewritability of the OMQ into an FO query Q and the possibility to compute in polynomial time the set $\mathcal{D}$ containing all the polynomially many sets of facts that can possibly satisfy Q. With $\mathcal{D}$ at hand, its elements can be considered in turn to verify whether there exists a minimal explanation containing the fact to be tested for relevance.

Theorem 18. *For any FO-rewritable language $\mathcal{L}$ considered here,* MINEX-REL$^{\models}(\mathcal{L})$ *is in* P *in the data complexity.*

We now consider the hardness results. To study the hardness of MINEX-REL$^{\models}(\mathcal{L})$, we can resort to its "positive" variant. Ceylan et al. (2019) studied the problem of deciding whether a fact ψ belongs to some minimal explanation for $D \models (Q, \Sigma)$, called MINEX-REL$(\mathcal{L})$. It is possible to show that MINEX-REL$(\mathcal{L})$ reduces to MINEX-REL$^{\models}(\mathcal{L})$, thereby obtaining all the hardness results in Table 4.

Theorem 19. *For any language $\mathcal{L}$ considered here,* MINEX-REL$(\mathcal{L})$ *is reducible in polynomial time to* MINEX-REL$^{\models}(\mathcal{L})$ *in the data, fp-combined, ba-combined, and combined complexity.*

We now analyze the problem MINEX-ALLREL$^{\models}(\mathcal{L})$ of deciding whether a set contains all and only the relevant facts. We focus on the membership results first. The following theorem proves all the upper bounds in Table 5. Intuitively, to check that H' is the set of all and only the relevant facts, it suffices to check that all facts in H' are relevant, and all facts outside H' are not relevant.

Theorem 20. *For any language $\mathcal{L}$ considered here, if* MINEX-REL$^{\models}(\mathcal{L})$ *is in the complexity class $\mathcal{C}$ in the combined (resp., ba-combined, fp-combined, data) complexity, then* MINEX-ALLREL$^{\models}(\mathcal{L})$ *can be decided with a $\mathcal{C}$ check and a co-$\mathcal{C}$ in the combined (resp., ba-combined, fp-combined, data) complexity.*

As for hardness, the following theorem proves all the hardness results in Table 5 in the data complexity for the non-FO-rewritable languages via a reduction from the classical D^{P}-complete problem SAT-UNSAT: given two Boolean formulas in 3CNF $\phi(X)$ and $\psi(X)$, decide whether ϕ is satisfiable, and ψ is not satisfiable. In a fixed program, there are the rules to check the satisfiability of the formulas. In H, there are facts to encode possible truth assignments for the Boolean variables X, plus some additional facts to recognize the satisfaction of ϕ and ψ, for the specific truth assignments in the candidate explanations. The idea is to have all facts in H to be relevant, apart from the one associated with the satisfaction of ψ.

Theorem 21. MINEX-ALLREL$^{\models}(\mathsf{GF}_\bot)$ *is* D^{P}-hard *in the data complexity.*

The following theorem proves all the D_2^{P} lower bounds in Table 5 in the *fp*-combined complexity. It also gives hardness results matching the upper bounds for $\mathsf{L}_\bot$, $\mathsf{F}_\bot$, $\mathsf{S}_\bot$, and their specializations in the *ba*-combined complexity. We show a reduction from the prototypical D_2^{P}-complete problem $\text{QBF}_{2,\forall,\neg}^{CNF} \wedge \text{QBF}_{2,\exists}^{CNF}$: decide the validity of two quantified Boolean formulas $\Phi = \exists X \forall Y \neg \phi(X, Y)$ and $\Psi = \forall X \exists Y \psi(X, Y)$ (notice that by an argument similar to the one in the proof of Theorem 3.9 in (Lukasiewicz and Malizia 2017), X and Y can be assumed w.l.o.g. to be the same in $\phi(X, Y)$ and $\psi(X, Y)$). The tests of satisfaction of $\phi(X, Y)$ and $\psi(X, Y)$ are in the query. In H, there are facts representing truth assignments for variables X, plus additional facts to recognize the validity of Φ and Ψ: we check for the relevance or not of these additional facts.

Theorem 22. *For any language $\mathcal{L}$ considered here,* MINEX-ALLREL$^{\models}(\mathcal{L})$ *is* D_2^{P}-hard *in the fp-combined complexity.*

The remaining hardness results in the *ba*-combined complexity as well as all the hardness results in the combined complexity are established via the following theorem, which shows that MINEX-EXISTS$^{\models}$can be reduced to the complement of MINEX-ALLREL$^{\models}$.

Theorem 23. *For any language $\mathcal{L}$ considered here,* MINEX-EXISTS$^{\models}(\mathcal{L})$ *is reducible in polynomial time to the complement of* MINEX-ALLREL$^{\models}(\mathcal{L})$ *in the data, fp-combined, ba-combined, and combined complexity.*

8 MINEX-NEC$^{\models}$and MINEX-ALLNEC$^{\models}$

In this section, we study the problems regarding necessary facts, i.e., facts appearing in all the minimal explanations. In particular, we analyze the problem of deciding whether a fact is necessary and the problem of deciding whether a set of facts is the set of all and only the necessary facts.

We now focus on the problem MINEX-NEC$^{\models}(\mathcal{L})$ of deciding whether a fact is necessary. As for the membership

results, the following theorem proves all upper bounds in Table 6 but the coNP and P ones, for which we need tighter results, as stated next. Intuitively, we can *dis*prove that a fact ψ is necessary by checking that there is a (non-necessarily minimal) explanation excluding ψ. Hence, we can answer by guessing a set $E \subseteq H \setminus \{\psi\}$ of facts (feasible in NP), and then check via oracle calls that E is consistent and entails the OMQ (Tasks (1) and (2) in the proof sketch of Theorem 8).

Theorem 24. *For any language $\mathcal{L}$ considered here, if OMQA($\mathcal{L}$) is in the complexity class $\mathcal{C}$ in the combined (resp., ba-combined, fp-combined, data) complexity, then* MINEX-NEC$^{\models}$*($\mathcal{L}$) is in co-($\mathrm{NP}^{\mathcal{C}}$) in the combined (resp., ba-combined, fp-combined, data) complexity.*

The following theorem provides the coNP upper bounds in the *fp*-combined complexity in Table 6. The result is obtained from the proof of Theorem 24 with the additional observation that in the *fp*-combined setting, for the Datalog$^{\pm}$ languages considered, checking whether a set of facts is consistent is feasible in P, because the negative constraints are fixed.

Theorem 25. *For any language $\mathcal{L}$ considered here,* MINEX-NEC$^{\models}$*($\mathcal{L}$) is in* coNP *in the fp-combined complexity.*

We conclude the discussion of the membership results for MINEX-NEC$^{\models}(\mathcal{L})$ by showing that the problem is in P in the data complexity for all the FO-rewritable languages considered, namely, L, S, and A (as well as their full restrictions). For the FO-rewritable languages, the following theorem proves the P membership results in the data complexity shown in Table 6 that are not obtained via Theorem 24. To prove this result, we exploit the rewritability of an OMQ into an FO query Q and the possibility to compute in polynomial time the set $\mathcal{D}$ containing all the polynomially many sets of facts that can possibly satisfy Q. Once $\mathcal{D}$ has been computed, its elements can be considered in turn to verify whether there exists a non-necessarily minimal explanation excluding the fact to be tested for necessity. If such an explanation is found, we answer no, otherwise we answer yes.

Theorem 26. *For any FO-rewritable language $\mathcal{L}$ considered here,* MINEX-NEC$^{\models}$*($\mathcal{L}$) is in* P *in the data complexity.*

Regarding the hardness results of MINEX-NEC$^{\models}(\mathcal{L})$, they can be proven by showing a reduction from the complement of MINEX-EXISTS$^{\models}(\mathcal{L})$ to MINEX-NEC$^{\models}(\mathcal{L})$.

Theorem 27. *For any language $\mathcal{L}$ considered here, the complement of* MINEX-EXISTS$^{\models}$*($\mathcal{L}$) is reducible in polynomial time to* MINEX-NEC$^{\models}$*($\mathcal{L}$) in the data, fp-combined, ba-combined, and combined complexity.*

We now study the problem MINEX-ALLNEC$^{\models}(\mathcal{L})$ of deciding whether a set contains all and only the necessary facts. We start by looking at the membership results. The following theorem proves all upper bounds in Table 7. Intuitively, to check that H' is the set of all and only the necessary facts, it suffices to check that all facts in H' are necessary and all facts outside H' are not necessary.

Theorem 28. *For any language $\mathcal{L}$ considered here, if* MINEX-NEC$^{\models}$*($\mathcal{L}$) is in the complexity class $\mathcal{C}$ in the combined (resp., ba-combined, fp-combined, data) complexity, then* MINEX-ALLNEC$^{\models}$*($\mathcal{L}$) can be decided by a check in $\mathcal{C}$ and a check in co-$\mathcal{C}$ in the combined (resp., ba-combined, fp-combined, data) complexity.*

$\mathcal{L}$	**Data**	***fp*-comb.**	***ba*-comb.**	**Comb.**
$\mathsf{L}_\bot$, $\mathsf{LF}_\bot$, $\mathsf{AF}_\bot$	$\leq$ P	coNP	Π_2^{P}	PSPACE
$\mathsf{S}_\bot$, $\mathsf{SF}_\bot$	$\leq$ P	coNP	Π_2^{P}	EXP
$\mathsf{A}_\bot$	$\leq$ P	coNP	$\mathrm{P}^{\mathrm{NEXP}}$	$\mathrm{P}^{\mathrm{NEXP}}$
$\mathsf{G}_\bot$	coNP	coNP	EXP	2EXP
$\mathsf{F}_\bot$, $\mathsf{GF}_\bot$	coNP	coNP	Π_2^{P}	EXP
$\mathsf{WS}_\bot$, $\mathsf{WA}_\bot$	coNP	coNP	2EXP	2EXP

Table 6: Complexity of MINEX-NEC$^{\models}(\mathcal{L})$.

$\mathcal{L}$	**Data**	***fp*-comb.**	***ba*-comb.**	**Comb.**
$\mathsf{L}_\bot$, $\mathsf{LF}_\bot$, $\mathsf{AF}_\bot$	$\leq$ P	D^{P}	$\mathrm{D}_2^{\mathrm{P}}$	PSPACE
$\mathsf{S}_\bot$, $\mathsf{SF}_\bot$	$\leq$ P	D^{P}	$\mathrm{D}_2^{\mathrm{P}}$	EXP
$\mathsf{A}_\bot$	$\leq$ P	D^{P}	$\mathrm{P}^{\mathrm{NEXP}}$	$\mathrm{P}^{\mathrm{NEXP}}$
$\mathsf{G}_\bot$	D^{P}	D^{P}	EXP	2EXP
$\mathsf{F}_\bot$, $\mathsf{GF}_\bot$	D^{P}	D^{P}	$\mathrm{D}_2^{\mathrm{P}}$	EXP
$\mathsf{WS}_\bot$, $\mathsf{WA}_\bot$	D^{P}	D^{P}	2EXP	2EXP

Table 7: Complexity of MINEX-ALLNEC$^{\models}(\mathcal{L})$.

As for the hardness results, the following theorem proves all the hardness results in Table 7 in the data complexity for the languages considered here that are not FO-rewritable. The proof is via a reduction from SAT-UNSAT. In H, there are facts associated with the possible truth assignments for the variables X of the formulas, plus additional facts allowing to recognize that the formulas ϕ and ψ are unsatisfiable. In this case, the idea is to have all facts in H to be non-necessary, apart from the one associated with the unsatisfaction of ψ.

Theorem 29. MINEX-ALLNEC$^{\models}(\mathsf{GF}_\bot)$ *is* D^{P}-hard *in the data complexity.*

The next theorem proves all the hardness results in Table 7 in *fp*-combined complexity. Also here, we provide a reduction from SAT-UNSAT. The set H includes a couple of facts allowing to recognize the unsatisfiability of ϕ and ψ. We check that the fact associated with the unsatisfiability of ψ is necessary.

Theorem 30. *For any language $\mathcal{L}$ considered here,* MINEX-ALLNEC$^{\models}$*($\mathcal{L}$) is* D^{P}-hard *in the fp-combined complexity.*

The following theorem proves the $\mathrm{D}_2^{\mathrm{P}}$-hardness in the *ba*-combined complexity for all the languages we consider. The theorem then provides matching lower bounds for the $\mathrm{D}_2^{\mathrm{P}}$-completeness results in Table 7, while for the remaining languages we need tighter lower bounds shown next.

The proof uses a reduction from the $\mathrm{D}_2^{\mathrm{P}}$-complete problem $\mathrm{QBF}_{2,\forall,\neg}^{CNF} \wedge \mathrm{QBF}_{2,\exists}^{CNF}$: given two quantified Boolean formulas $\Phi = \exists X \forall Y \neg\phi(X,Y)$ and $\Psi = \forall Z \exists K \psi(Z,K)$, decide whether Φ and Ψ are both valid. In H, there are facts associated with the various valid truth assignments for the variables in X and Z. The test for the satisfiability of $\phi(X,Y)$ and $\psi(Z,K)$, when a truth assignment for X and Z is encoded in a candidate explanation, is captured with two negative constraints. Additionally, H contains facts encoding a special truth assignment (using a special truth value) for the variables

in X and Z that allow us to bypass the satisfiability tests for $\phi(X, Y)$ and $\psi(Z, K)$. Then, H' contains the facts encoding the special truth assignment for Z. All facts encoding valid truth assignments are not necessary. Thus, H' contain exactly all necessary facts iff all facts encoding the special truth assignment for Z are necessary and all facts encoding the special truth assignment for X are not necessary. Then, Φ is valid iff we can build an explanation not using any of the facts encoding the special truth assignment for X; Ψ is valid iff every explanation that we can build must contain all facts encoding the special truth assignment for Z.

Theorem 31. *For any language $\mathcal{L}$ here considered,* MINEX-ALLNEC$^{\not\models}$*($\mathcal{L}$) is* $\mathrm{D}_2^{\mathrm{P}}$-hard *in the ba-combined complexity.*

The remaining hardness results in the *ba*-combined complexity as well as all the hardness results in the combined complexity are established via the following theorem, which shows that MINEX-EXISTS$^{\not\models}$ can be reduced to the complement of MINEX-ALLNEC$^{\not\models}$.

Theorem 32. *For any language $\mathcal{L}$ considered here,* MINEX-EXISTS$^{\not\models}$*($\mathcal{L}$) is reducible in polynomial time to the complement of* MINEX-ALLNEC$^{\not\models}$*($\mathcal{L}$) in the data, fp-combined, ba-combined, and combined complexity.*

9 Related Work

The literature on explanations is very rich. Abstracting away from subtle differences, the study of explanations can be classified in accordance to the (i) underlying *logical formalism*, (ii) *reasoning task* to be explained (e.g., query answering and consistency checking), and (iii) *type of consequences* to be explained (e.g., entailments vs. non-entailments). We focus on existential rules as the underlying logical formalism, on OMQA as the reasoning task, and on non-entailments.

Our study may be seen as an instance of abductive reasoning, explaining observations in terms of sets of hypotheses that may have led to those observations. Abductive reasoning has been studied for several formalisms, such as propositional logic (Eiter and Gottlob 1995), logic programs (Eiter, Gottlob, and Leone 1997b), default theories (Eiter, Gottlob, and Leone 1997a), probabilistic temporal logic (Molinaro, Sliva, and Subrahmanian 2014), and DLs (Du et al. 2011; Du, Wang, and Shen 2014; Del-Pinto and Schmidt 2019; Wang et al. 2015; Klarman, Endriss, and Schlobach 2011; Calvanese et al. 2013). The study of ABox abduction by Klarman, Endriss, and Schlobach (2011) is related to our approach, as they also view a set of assertions (i.e., facts) as explanations. The closest work to ours is the one by Calvanese et al. (2013): for a given query that is *not* entailed from the knowledge base, the idea is to find a set of assertions such that, when they are added to the ABox, the entailment holds (avoiding inconsistencies). They consider arbitrary, inclusion-minimal, and cardinality-minimal explanations. Differently from their work, we focus on existential rules, and also consider novel explanation problems. One important difference is in the way abducibles are defined, which in turn affects how explanations are defined: in our case, explanations are subsets of a given set H of abducible *facts*; in contrast, they start with a set of abducible *predicates*, called *signature*, and explanations are sets of facts whose predicate belongs to the signature. This formulation is suitable when abducible predicates are known, but relevant constants may not be known. The same problem formalization has been adopted in (Du et al. 2011; Du, Wang, and Shen 2014). The latter work also identified a class of TBoxes guaranteeing the existence of a finite number of minimal explanations (as there can be infinitely many minimal explanations). Furthermore, they introduced a special kind of minimal explanations, called *representative* explanations, from which all minimal explanations can be retrieved. Different works have focused on the efficient computation of explanations, e.g., see (Du et al. 2011; Du, Wang, and Shen 2014; Wang et al. 2015).

Explanations for positive entailments have been studied in DLs extensively; see, e.g., (McGuinness and Borgida 1995; Borgida, Franconi, and Horrocks 2000), where explanations are in the form of proofs. Most of the subsequent works focused on explanations, which are minimal subsets of the given theory (Schlobach and Cornet 2003). This is known as *axiom pinpointing* (Kalyanpur et al. 2007; Baader and Suntisrivaraporn 2008; Peñaloza and Sertkaya 2017), and such explanations are called *justifications* (Horridge, Parsia, and Sattler 2008). These works focus on DLs, and on explaining classical reasoning tasks. Explanations for (positive) OMQA has been first investigated for the *DL-Lite* family of languages (Borgida, Calvanese, and Rodriguez-Muro 2008). Recently, a thorough investigation is given by Ceylan et al. (2019) for existential rules and by Ceylan et al. (2020) for DLs. These works address the problem of explaining why a query is entailed, as opposed to our approach.

Explaining query answers has also been studied in the context of *inconsistency-tolerant query answering*. The problem has been addressed in (Du, Wang, and Shen 2015; Bienvenu, Bourgaux, and Goasdoué 2019) for DL languages and in (Hecham et al. 2017; Lukasiewicz, Malizia, and Molinaro 2020) for existential rule languages. The main difference between this paper and the aforementioned ones is that we provide explanations under standard semantics and for *consistent* knowledge bases.

10 Summary and Outlook

We have addressed the problem of explaining why a query is not entailed in OMQA under existential rules. (Minimal) Explanations for a non-entailment can be seen as a (minimal) set of facts such that, if the database were extended with this set, the query would be entailed, while the knowledge base would remain consistent with the added information. We have conducted a detailed complexity analysis for various explanation problems, for a wide range of existential rule languages under different complexity measures.

As for future work, it would be interesting to study our framework under different minimality criteria (e.g., by considering cardinality-minimal explanations) or to express other forms of preferences among explanations. Another interesting direction for future work is to study the problems of computing all explanations or relevant/necessary facts.

Acknowledgments

This work was supported by the Alan Turing Institute under the UK EPSRC grant EP/N510129/1, the AXA Research Fund, and by the EPSRC grants EP/R013667/1, EP/L012138/1, and EP/M025268/1.

References

Baader, F., and Suntisrivaraporn, B. 2008. Debugging SNOMED CT using axiom pinpointing in the description logic $\mathcal{EL}^+$. In *Proc. KR-MED*, 1–7.

Baader, F.; Calvanese, D.; McGuinness, D. L.; Nardi, D.; and Patel-Schneider, P. F., eds. 2007. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2nd edition.

Bienvenu, M.; Cate, B. T.; Lutz, C.; and Wolter, F. 2014. Ontology-based data access: A study through disjunctive Datalog, CSP, and MMSNP. *ACM Trans. Database Syst.* 39(4):33:1–33:44.

Bienvenu, M.; Bourgaux, C.; and Goasdoué, F. 2019. Computing and explaining query answers over inconsistent *DL-Lite* knowledge bases. *J. Artif. Intell. Res.* 64:563–644.

Borgida, A.; Calvanese, D.; and Rodriguez-Muro, M. 2008. Explanation in the *DL-Lite* family of description logics. In *Proc. OTM*, 1440–1457.

Borgida, A.; Franconi, E.; and Horrocks, I. 2000. Explaining $\mathcal{ALC}$ subsumption. In *Proc. ECAI*, 209–213.

Calì, A.; Gottlob, G.; and Kifer, M. 2013. Taming the infinite chase: Query answering under expressive relational constraints. *J. Artif. Intell. Res.* 48:115–174.

Calì, A.; Gottlob, G.; and Lukasiewicz, T. 2012. A general Datalog-based framework for tractable query answering over ontologies. *J. Web Sem.* 14:57–83.

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

Calvanese, D.; Ortiz, M.; Simkus, M.; and Stefanoni, G. 2013. Reasoning about explanations for negative query answers in *DL-Lite*. *J. Artif. Intell. Res.* 48:635–669.

Ceylan, İ. İ.; Lukasiewicz, T.; Malizia, E.; and Vaicenavičius, A. 2019. Explanations for query answers under existential rules. In *Proc. IJCAI*, 1639–1646.

Ceylan, İ. İ.; Lukasiewicz, T.; Malizia, E.; and Vaicenavičius, A. 2020. Explanations for ontology-mediated query answering in description logics. In *Proc. ECAI*.

Del-Pinto, W., and Schmidt, R. A. 2019. ABox abduction via forgetting in $\mathcal{ALH}$. In *Proc. AAAI*, 2768–2775.

Du, J.; Qi, G.; Shen, Y.; and Pan, J. Z. 2011. Towards practical ABox abduction in large OWL DL ontologies. In *Proc. AAAI*, 1160–1165.

Du, J.; Wang, K.; and Shen, Y. 2014. A tractable approach to abox abduction over description logic ontologies. In *Proc. AAAI*, 1034–1040.

Du, J.; Wang, K.; and Shen, Y. 2015. Towards tractable and practical ABox abduction over inconsistent description logic ontologies. In *Proc. AAAI*, 1489–1495.

Eiter, T., and Gottlob, G. 1995. The complexity of logic-based abduction. *J. ACM* 42(1):3–42.

Eiter, T.; Gottlob, G.; and Leone, N. 1997a. Abduction from logic programs: Semantics and complexity. *Theor. Comput. Sci.* 189(1-2):129–177.

Eiter, T.; Gottlob, G.; and Leone, N. 1997b. Semantics and complexity of abduction from default theories. *Artif. Intell.* 90(1-2):177–223.

Eiter, T.; Lukasiewicz, T.; and Predoiu, L. 2016. Generalized consistent query answering under existential rules. In *Proc. KR*, 359–368.

Fagin, R.; Kolaitis, P. G.; Miller, R. J.; and Popa, L. 2005. Data exchange: Semantics and query answering. *Theor. Comput. Sci.* 336(1):89–124.

Hecham, A.; Arioua, A.; Stapleton, G.; and Croitoru, M. 2017. An empirical evaluation of argumentation in explaining inconsistency-tolerant query answering. In *Proc. DL*.

Horridge, M.; Parsia, B.; and Sattler, U. 2008. Laconic and precise justifications in OWL. In *Proc. ISWC*, 323–338.

Kalyanpur, A.; Parsia, B.; Horridge, M.; and Sirin, E. 2007. Finding all justifications of OWL DL entailments. In *Proc. ISWC/ASWC*, 267–280.

Klarman, S.; Endriss, U.; and Schlobach, S. 2011. ABox abduction in the description logic $\mathcal{ALC}$. *J. Autom. Reasoning* 46(1):43–80.

Lukasiewicz, T., and Malizia, E. 2017. A novel characterization of the complexity class Θ_k^P based on counting and comparison. *Theor. Comput. Sci.* 694:21–33.

Lukasiewicz, T.; Malizia, E.; and Molinaro, C. 2020. Explanations for inconsistency-tolerant query answering under existential rules. In *Proc. AAAI*, 2909–2916.

McGuinness, D. L., and Borgida, A. 1995. Explaining subsumption in description logics. In *Proc. IJCAI*, 816–821.

Molinaro, C.; Sliva, A.; and Subrahmanian, V. S. 2014. Super-solutions: Succinctly representing solutions in abductive annotated probabilistic temporal logic. *ACM Trans. Comput. Log.* 15(3):18:1–18:35.

Peñaloza, R., and Sertkaya, B. 2017. Understanding the complexity of axiom pinpointing in lightweight description logics. *Artif. Intell.* 250:80–104.

Schlobach, S., and Cornet, R. 2003. Non-standard reasoning services for the debugging of description logic terminologies. In *Proc. IJCAI*, 355–360.

Vardi, M. Y. 1982. The complexity of relational query languages. In *Proc. STOC*, 137–146.

Wang, Z.; Chitsaz, M.; Wang, K.; and Du, J. 2015. Towards scalable and complete query explanation with OWL 2 EL ontologies. In *Proc. CIKM*, 743–752.