




Explanations for query answers under existential rules [☆]

İsmail İlkan Ceylan ^{a, }, Thomas Lukasiewicz ^{b, }, Enrico Malizia ^{c, },
Andrius Vaicenavičius ^a

^a Department of Computer Science, University of Oxford, UK

^b Institute of Logic and Computation, Vienna University of Technology, Austria

^c DISI, University of Bologna, Italy

ARTICLE INFO

MSC:

68Q17

68T27

68T30

Keywords:

Artificial intelligence

Computational complexity

Ontologies

Existential rules

Datalog+/-

Query answering

Ontology mediated query answering

Explanations

ABSTRACT

Ontology-based data access is an extensively studied paradigm aiming at improving query answers with the use of an “ontology”. An ontology is a specification of a domain of interest, which, in this context, is described via a logical theory. As a form of logical entailment, ontology-mediated query answering is fully interpretable, which makes it possible to derive explanations for ontological query answers. This is a quite important aspect, as the fact that many recent AI systems mostly operating as black boxes has led to some serious concerns. In the literature, various works on explanations in the context of description logics (DLs) have appeared, mostly focusing on explaining concept subsumption and concept unsatisfiability in the ontologies. Some works on explaining query entailment in DLs have appeared as well, however, mainly dealing with inconsistency-tolerant semantics and, actually, *non*-entailment of the queries. Surprisingly, explaining ontological query entailment has received little attention for ontology languages based on existential rules. In fact, although DLs are popular formalisms to model ontologies, it is generally agreed that rule-based ontologies are well-suited for data-intensive applications, as they allow us to conveniently deal with higher-arity relations, which naturally occur in standard relational databases. The goal of this work is to close this gap, and study the problem of explaining query entailment in the context of existential rules ontologies in terms of minimal subsets of database facts. We provide a thorough complexity analysis for several decision problems associated with minimal explanations for various classes of existential rules, and for different complexity measures.

1. Introduction

Ontology-based data access (OBDA) [95] has recently emerged as a popular paradigm in knowledge representation and reasoning technologies. The goal of OBDA is to facilitate the access to data by separating the user from the raw data sources through the use of an ontology. An ontology is an explicit specification via an unambiguous language, typically based on logic, of an abstract model of a domain of interest. In this context, the ontology therefore provides a unified conceptual view of the data sources, which can hence be

[☆] This article is an extended and revised version of a preliminary paper appeared in: Proc. of IJCAI 2019 [35]. Compared to the conference version, we here provide full details of the proofs, some proofs have been streamlined, and we additionally consider the Datalog[±] fragment WG.

* Corresponding author.

E-mail addresses: ismail.ceylan@cs.ox.ac.uk (İ.İ. Ceylan), thomas.lukasiewicz@tuwien.ac.at (T. Lukasiewicz), enrico.malizia@unibo.it (E. Malizia), andrius.vaicenavicius@st-annes.ox.ac.uk (A. Vaicenavičius).

<https://doi.org/10.1016/j.artint.2025.104294>

Received 4 January 2024; Received in revised form 6 November 2024; Accepted 28 January 2025

heterogeneous, and makes it accessible via queries solely formulated in the vocabulary of the ontology without any knowledge of the actual structure of the data. Furthermore, by formalizing the domain knowledge, the ontology also enriches the possibly incomplete data sources, enabling hence more complete answers to queries.

Description logics (DLs) [8] and *existential rules* [28–30], together, encompass the most widely used knowledge representation languages in the context of OBDA. Both families of languages are first-order logic fragments, which result from a trade-off between the expressivity of the language and the computational complexity (or even decidability) of reasoning in the language. Since they are logic-based languages, ontology-mediated query answering is fully interpretable for them, which makes it possible to derive explanations for ontological query answers. This is quite important for modern AI systems, as the fact that many recent AI systems mostly operating as black boxes has led to some serious concerns [44]. Ontologies have found applications in data exchange [54], medical diagnosis [13], and life sciences [11], just to name a few, all of which can potentially benefit from the idea of explanations.

In fact, there has been a significant interest in tracking down and understanding the causes of various types of “consequences” in ontologies. Early works dealing with explanations in ontologies primarily focused on DLs as the underlying ontology language and, interestingly, they had a “debugging” perspective. This was because, since devising an ontology is an error-prone task, people were interested in having tools supporting the analysis of ontologies and their consequences. In particular, the consequences of an ontology that they focused on were *concept subsumption* or *concept unsatisfiability*. In the former task, one wants to know why a concept subsumes another; this was needed to double-check in the ontology the explicitly and implicitly defined taxonomies. In the latter task, it is asked why a given concept is unsatisfiable; this was regarded as an important sign of a badly-designed ontology.

A prominent approach has been to identify explanations for the above two types of consequences in terms of subsets of the ontology axioms [7,71]. The benefit of this approach, in contrast to those based on proof systems, which had previously been proposed in the literature, allows us to abstract away from the particular proof technique used to derive an entailment, and hence to pinpoint the sets of axioms that are responsible for a consequence. Such explanation sets are, furthermore, required to be minimal with respect to some order, like subset-inclusion, cardinality, or a preference order. These explanations are called *justifications* [64,71,105], and the overall approach, in the DL literature [7,65], is also known as *axiom pinpointing*. Despite the extensive work carried out on explaining consequences in ontologies, there has been very little work in the direction of explaining query entailment. To date, the only works dealing with the explanation of query answers are for the *DL-Lite* family of languages [19,21] (see the related work section).

Surprisingly, explanations have not been studied in the context of existential rules. This formalism to express ontologies is recognized to be well-suited for data-intensive applications such as OBDA, as it allows us to deal with higher-arity relations, which naturally occur in standard relational databases. We provide an example to give an intuition on what query answer explanations are in the context of existential rules.

Example 1.1. In this paper, a knowledge base $KB = (D, \Sigma)$ is a pair, where D is a relational database, and Σ is a set of “rules” describing the domain of interest. Let us assume to have the following knowledge base, whose database D is:

$$dep(cs, u), dep(math, u), affil(matthew, cs), affil(john, math), prof(john), head(john, math).$$

The facts in this database state that cs and $math$ are two departments in the university u , that $matthew$ and $john$ are affiliated to the departments cs and $math$, respectively, and that $john$ is a professor and the head of the department $math$. Let us assume that the rules Σ of the knowledge base are:

$$head(P, D) \rightarrow affil(P, D),$$

$$head(P, D) \rightarrow prof(P),$$

$$dep(D, U) \wedge affil(X, D) \rightarrow (\exists P) prof(P) \wedge affil(P, D) \wedge head(P, D).$$

The first rule states that if P is the head of a department D , then P must be affiliated to D . The second rule states that if P is the head of a department D , then P must be a professor. The third rule states that if D is a department in the university U and some person X is affiliated to D , then there must be a professor P affiliated to D who is the head of D .

Consider now the query $q_a = prof(P)$, asking whether there is a professor in the university. The knowledge base entails the query, i.e., the query holds true over the knowledge base. In the present paper, an explanation will be a set of facts *from the database* that are sufficient to entail the query via the mediation of the rules. The set $E_1 = \{prof(john)\}$ is clearly an explanation for the query, but it is not the only one. Also the set $E_2 = \{dep(cs, u), affil(matthew, cs)\}$ is an explanation for the query. Indeed, since E_2 declares that cs is a department to which $matthew$ is affiliated, by the third rule, there *must* be a professor (of not specified identity) that is the head of cs . This rule does not say who this professor is, but the rule states that there must be one. So, the set E_2 provides a reason, i.e., explains, why q_a holds true over the knowledge base, and the reason is the content of E_2 : the department cs has $matthew$ affiliated (and by the third rule there must be a professor leading cs). Another explanation for q_a is $E_3 = \{head(john, math)\}$, because via the second rule we know that $john$ must be a professor as well.

Consider now the query $q_b = head(P, cs)$, asking whether there is a head of the department cs . Also q_b is entailed by the knowledge base, and an explanation is the set E_2 mentioned above, for the same reason already pointed out.

Consider now the query $q_c = head(P, math)$, asking whether there is a head of the department $math$. In this case, the set $E_4 = \{head(john, math)\}$ is clearly an explanation for q_c , but also $E_5 = \{dep(math, u), affil(john, math)\}$ is an explanation for q_c , for a reason similar to that for which E_2 is an explanation for q_a . \triangleleft

Interestingly, more recently, query answer explanations within the rule framework have found application in “compositional” neuro-symbolic systems [107]. To give an example of these systems, we report the scenario described by Tsamoura et al. [107] (see the referenced work for additional details). Assume that a compositional neuro-symbolic system needs to learn to recognize (images of) chess configurations on a chessboard corresponding to the concepts that the black king has a valid move (“safe”), is stalemated (“draw”), or is mated (“mate”). The system is composed by two parts. The first sub-symbolic (neuronal) part of the system receives in input an image of the chessboard with the pieces on it, and outputs the locations of the pieces on the chessboard. The neuronal part’s output is re-encoded into a set database facts stating the position of the pieces on the chessboard. The second symbolic (logical) part of the system receives the database facts obtained from the output of the neuronal part, and via an ontology describing the rules of the game of chess outputs whether the configuration is “safe”/“draw”/“mate”. When this system needs to be trained as a whole, the learning process has to go through two parts working together, but in different ways. The training set for the system is constituted by pairs of a chessboard configuration (an image) and a label “safe”/“draw”/“mate”. During the training process, given an image, the neuronal part makes some prediction regarding the locations of the pieces on the chess board. This prediction is re-encoded into database facts. These database facts are then used to infer, via the ontology, whether the position is “safe”/“draw”/“mate”. From this predicted label, it can be seen whether there is a mistake or not with respect to the label in the training instance. If the label predicted and the label in the training instance differ, the training step proceeds as follows. From the label in the training instance, the possible sets of database facts describing a suitable chessboard configuration explaining the desired “safe”/“draw”/“mate” label appearing in the training instance are built—in [107], these explanations are called “abductive proofs”. These explanations are essentially the possible expected outputs of the neuronal part, which can then be trained via standard backpropagation without the need for the logical theory in the ontology to be differentiable (see [107] for more details).

Studying queries’ entailment explanations and analyzing the computational complexity of related problems are therefore quite relevant research questions. This is precisely the main focus of the present work. More specifically, given a knowledge base, i.e., a pair composed by a relational database and an ontology expressed via existential rules, and a (conjunctive) query, we are interested in explaining the query entailment by the knowledge base in terms of the minimal subsets of database facts satisfying the query. Such a minimal subset of the database is called *minimal explanation*, or simply *MinEx*. By incorporating ideas from axiom pinpointing [94], we introduce a class of problems based on the notion of minimal explanation. In particular, for a knowledge base and a query, we analyze the problem IS-MINEX (resp., ALL-MINEX) of deciding whether a given set of database facts (resp., a given sets of database facts) is a MinEx (resp., are all and only the MinExes) for the query; the problems MINEX-IRREL and MINEX-REL asking for the existence of a MinEx for the query excluding sets of forbidden facts and including a distinguished fact, respectively; and the problems SMALL-MINEX and LARGE-MINEX concerning whether the query admits a MinEx whose size does not exceed and is at least a given threshold, respectively.

We conduct a detailed complexity analysis for each of the problems introduced, and provide a host of complexity results that cover a representative set of existential rules. In fact, it is known that query answering under arbitrary existential rules is undecidable (see, e.g., [30]). This has led to identifying restrictions on existential rules guaranteeing decidability. The main decidable paradigms that we consider in this paper are *guardedness* [30] (which includes *linearity* [28]), *acyclicity* [54], and *stickiness* [29]. Furthermore, we carry out the complexity analysis of all these problems for four interesting complexity measures. The most general one is the *combined* complexity, in which both the knowledge base and the query are considered part of the problem’s input (i.e., they can vary). Then, we consider the *bounded-arity combined* complexity, or *ba-combined* complexity, which is similar to the previous, with the exception that the relational schema over which the knowledge base is defined has predicates whose arity is bounded by a fixed integer. Moreover, the *fixed-program combined* complexity, or *fp-combined* complexity, is also analyzed, in which the ontology of the input knowledge base is fixed, and only the database and the query can change in the input. Furthermore, we study the *data* complexity, where only the database may vary in the input (all the rest is fixed, also the query).

In what follows, we delineate a quick overview of our complexity results, which range from membership in AC^0 to 2EXP-completeness. In the discussion below, we denote by OMQA the problem of deciding ontological query entailment, that is, given a knowledge base and a query, decide whether the query is entailed by the knowledge base.

- Regarding IS-MINEX, its data complexity does not increase compared to OMQA. Indeed, in the data complexity, IS-MINEX remains tractable (in most cases), because, to check the minimality of an explanation, we need to perform a polynomial number of extra non-entailment checks, which, although are not required to answer OMQA, can be carried out in polynomial time in the data complexity for most of the existential rules languages that we consider (and, in general, in the same complexity class of OMQA). Moreover, IS-MINEX is in AC^0 for FO-rewritable languages in the data complexity, because we can capture the property of being a MinEx with a FO-formula. In the *fp-combined* complexity setting and above, IS-MINEX shows in various cases an increased complexity compared to OMQA. This is because, in these cases, the necessary check of the explanation’s minimality does not come for free, as the non-entailment checks are in a complement non-deterministic complexity class.
- Interestingly, for acyclic rules, we prove IS-MINEX D^{EXP} -complete in the *ba-combined* and *combined* complexity, instead of the OMQA’s easier NEXP-completeness. The class D^{EXP} is defined, similarly to D^P , as the class of languages that are the intersection of a NEXP language and a co-NEXP language—this class is at the second level of the Boolean Hierarchy over NEXP [42]. This complexity result in this case stems from the fact that checking whether a set is a MinEx requires to test that the set entails the query, which is a NEXP task, and that the set is minimal, which is a co-NEXP task. These two tasks can be shown to be completely independent in this case, and hence, intuitively, we can have them solved in parallel by two distinct machines, from which the complexity in $D^{EXP} = NEXP \wedge co-NEXP$ follows. To the best of our knowledge, this D^{EXP} -completeness result characterizes IS-MINEX, together with ALL-MINEX, among the very few known natural problems complete for D^{EXP} .

- For ALL-MINEX, we observe an intractability result already in the data complexity for guarded existential rules, because checking that there is no MinEx outside the input candidate set of all MinExes requires a co-NP check. Surprisingly, we see that, in the *fp*-, *ba*-, and combined complexity settings, IS-MINEX and ALL-MINEX are complete for the same complexity classes. This is because, when solving ALL-MINEX, for any set \mathcal{E} of subsets of the database D , checking the minimality of all the sets in \mathcal{E} and ensuring that there is no MinEx outside \mathcal{E} can be combined. This results in matching complexities of the two problems for these complexity settings.
- The problems MINEX-IRREL and SMALL-MINEX have the same complexity, which, compared to that of OMQA, shows an increase only in the data complexity setting. For both problems, we observe intractability results in the data complexity for guarded rules, as we need to guess an explanation satisfying the conditions (i.e., it avoids forbidden sets or is smaller than a threshold, respectively), which requires an NP machine. These problems do not have a higher complexity than OMQA in the *fp*-combined complexity setting and above, because, if there is an explanation E that either avoids the forbidden sets or is smaller than some threshold number, then all subsets of E enjoy this property as well. Therefore, we do *not* need to check the minimality of the guessed explanation.
- The complexity results for the problems MINEX-REL and LARGE-MINEX match as well. Similarly to what happens for MINEX-IRREL and SMALL-MINEX, there is an increase in the complexity compared to OMQA in the data complexity due to the need to guess in NP an explanation satisfying the required properties (i.e., either including the distinguished fact or having a size bigger than the threshold). We have Σ_2^P -completeness results in the *fp*-combined complexity, which is one step higher in the polynomial hierarchy compared to OMQA's complexity in the same setting. This is because we need to guess an explanation satisfying the required property and check that the set is a MinEx. However, already checking whether a set is a MinEx is intractable in the *fp*-combined complexity for all the existential rules languages here considered; from this the Σ_2^P results follow.
- For these two problems, on acyclic rules, we obtain P^{NEXP} -completeness in the *ba*-combined and combined complexity, instead of the OMQA's easier NEXP-completeness. This is because, for MINEX-REL and LARGE-MINEX, after guessing in NP the candidate explanation, we need to check that it actually entails the query, which is a task in NEXP, and we also need to check its minimality, which is a task in co-NEXP. However, on the contrary of what happens for IS-MINEX, these two tasks are not independent in this case, as they are linked by the initial NP guess of the candidate MinEx. Hence, in this case, we obtain an NP^{NEXP} complexity, which, by the collapse of the Strong Exponential Hierarchy [62], equals P^{NEXP} , instead of a D^{EXP} complexity, as for IS-MINEX and ALL-MINEX.

The paper is organized as follows. Basics on relational databases, conjunctive queries, existential rules, as well as basic complexity classes are recalled in Section 2. Formal definitions of minimal explanations and the problems here analyzed, together with some examples, are provided in Section 3. Then, we start our complexity analysis, and, in each section, we study each of the considered problems in turn. More specifically, from Section 4 to Section 9, we analyze, in this order, the problems IS-MINEX, ALL-MINEX, MINEX-IRREL, SMALL-MINEX, MINEX-REL, and LARGE-MINEX, respectively. In Section 10, we discuss related work, and we settle our paper in the research context of explanations in ontologies. Section 11 concludes our work and we also discuss possible directions for further research.

2. Preliminaries

In this section, we recall the basics on relational databases, (unions of) conjunctive queries, ontological query answering, and the complexity classes encountered in this paper.

Relational databases and (unions of) conjunctive queries. Throughout the paper, we assume the mutually disjoint countably infinite sets *Const*, *Null*, and *Var* of *constants*, (*labelled*) *nulls*, and *variables*, respectively. We refer to constants, nulls, and variables also as *terms*.

A (*relational*) *schema* S is a finite set of *relation symbols* (or *predicates*) with an associated arity. We write r/n to denote that the arity of the predicate r is $n \geq 0$. A (*relational*) *atom* over S is an expression of the form $r(t_1, \dots, t_n)$, where r is an n -ary predicate from S , and (t_1, \dots, t_n) is an n -tuple of terms. An atom is *ground* if it contains only constants and nulls. A *fact* is a ground atom containing only constants. For a nullary predicate $p/0$, the only ground atom over p , which is also a fact, is denoted by $p()$. A *schema instance* over S is a (possibly infinite) set of ground atoms over S , while a *database* over S is a finite schema instance over S containing only facts, i.e., a finite instance without nulls. For a set A of atoms, $dom(A)$ denotes the set of all terms (variables, constants, and nulls) occurring in A .

Let $T = \{t_1, \dots, t_u\}$ and $S = \{s_1, \dots, s_v\}$ be two sets of terms. A *mapping* from T to S is a function $h : T \rightarrow S$; a mapping h from T to S can be seen as a set $h = \{t_{i_1} \mapsto s_{i_1}, \dots, t_{i_n} \mapsto s_{i_n}\}$, where $t_{i_j} \in T$ and $s_{i_j} \in S$ for all j , denoting that $h(t_{i_j}) = s_{i_j}$. The restriction of h to $T' \subseteq T$, denoted by $h|_{T'}$, is the mapping from T' to S such that, for every $t \in T'$, $h|_{T'}(t) = h(t)$. Let A and B be two sets of atoms. A *homomorphism* from A to B is a mapping h from the set of terms in A to the set of terms in B , i.e., from $dom(A)$ to $dom(B)$, such that (i) if $t \in Const$, then $h(t) = t$, and (ii) if $r(t_1, \dots, t_n) \in A$, then $h(r(t_1, \dots, t_n)) = r(h(t_1), \dots, h(t_n)) \in B$. When dealing with homomorphisms, in what follows we will regard conjunctions of atoms and tuples of variables as set of atoms and variables, respectively.

A *conjunctive query* (CQ) over a schema S is a first-order formula of the form

$$q(\mathbf{X}) = \exists \mathbf{Y} \phi(\mathbf{X}, \mathbf{Y}),$$

where ϕ is a conjunction of atoms over S without nulls and over the variables \mathbf{X}, \mathbf{Y} . The evaluation of CQs is defined via homomorphisms. For an instance I , the evaluation of $q(\mathbf{X})$ over I , denoted $q(I)$, is the set of all tuples $\mathbf{t} \in \text{Const}^{|\mathbf{X}|}$ such that there exists a homomorphism h from $\phi(\mathbf{X}, \mathbf{Y})$ to I with $h(\mathbf{X}) = \mathbf{t}$.

If \mathbf{X} is empty, i.e., if $q = \exists \mathbf{Y} \phi(\mathbf{Y})$, then q is a *Boolean conjunctive query* (BCQ). Notice that, for a BCQ, the only possible answer is the empty tuple. A BCQ $q = \exists \mathbf{Y} \phi(\mathbf{Y})$ is *true* or *holds* over I , if $q(I) = \{\emptyset\}$, which means that there is a homomorphism h such that $h(\phi(\mathbf{Y})) \subseteq I$. When q holds over I , we denote this by $I \models q$.

A *union of conjunctive queries* over S is a first-order formula of the form

$$q(\mathbf{X}) = q_1(\mathbf{X}) \vee \dots \vee q_n(\mathbf{X}) = \exists \mathbf{Y}_1 \phi_1(\mathbf{X}, \mathbf{Y}_1) \vee \dots \vee \exists \mathbf{Y}_n \phi_n(\mathbf{X}, \mathbf{Y}_n),$$

where each $q_i(\mathbf{X})$ is a CQ over S . The evaluation of such a union of CQs q over an instance I , denoted again by $q(I)$, is defined as the set of tuples $\bigcup_{i=1}^n q_i(I)$. By a slight abuse of notation, we may treat a union of conjunctive queries $q(\mathbf{X})$ like the one above as the set of CQs $\{q_1(\mathbf{X}_1), \dots, q_n(\mathbf{X}_n)\}$.

Also in this case, if \mathbf{X} is empty, i.e., if $q = q_1 \vee \dots \vee q_n = \exists \mathbf{Y}_1 \phi_1(\mathbf{Y}_1) \vee \dots \vee \exists \mathbf{Y}_n \phi_n(\mathbf{Y}_n)$, then q is a *union of Boolean conjunctive queries* (UCQ).¹ Also for a UCQ the only possible answer is the empty tuple, as the only possible answer for any of the constituent BCQs $q_i = \exists \mathbf{Y}_i \phi_i(\mathbf{Y}_i)$ is the empty tuple. We say that a UCQ $q = q_1 \vee \dots \vee q_n$ is *true*, or *holds*, over I , denoted $I \models q$, if $q(I) = \{\emptyset\}$, that is, there is a query $q_i \in q$ that is true over I .

Dependencies. A *tuple-generating dependency* (TGD) σ over a relational schema S is a first-order sentence

$$\forall \mathbf{X} \forall \mathbf{Y} (\phi(\mathbf{X}, \mathbf{Y}) \rightarrow \exists \mathbf{Z} \psi(\mathbf{X}, \mathbf{Z})),$$

where \mathbf{X}, \mathbf{Y} , and \mathbf{Z} , are pairwise disjoint tuples of variables from Var , and $\phi(\mathbf{X}, \mathbf{Y})$ and $\psi(\mathbf{X}, \mathbf{Z})$ are (non-empty) conjunctions of atoms over S , all without nulls [12]. For brevity, we write σ as ' $\phi(\mathbf{X}, \mathbf{Y}) \rightarrow \exists \mathbf{Z} \psi(\mathbf{X}, \mathbf{Z})$ '. We refer to $\phi(\mathbf{X}, \mathbf{Y})$ and $\psi(\mathbf{X}, \mathbf{Z})$ as the *body* and *head* of σ , denoted $\text{body}(\sigma)$ and $\text{head}(\sigma)$, respectively. An instance I satisfies a TGD σ like the one above, 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 an extension h' of $h|_{\mathbf{X}}$, i.e., $h' \supseteq h|_{\mathbf{X}}$, such that $h'(\psi(\mathbf{X}, \mathbf{Z})) \subseteq I$. The instance I satisfies a set Σ of TGDs, written $I \models \Sigma$, if $I \models \sigma$ for each $\sigma \in \Sigma$. Finite sets of TGDs are also called *programs*, and TGDs are also called *existential rules*. In the following, we consider only finite sets of TGDs.

Knowledge bases and ontological query answering. A *knowledge base* over a relational schema S is a pair (D, Σ) , where D is a database over S , and Σ is a program over S . Programs are the ontologies of the knowledge bases, and in this paper we use the two terms interchangeably. The set of *models* of a knowledge base $KB = (D, \Sigma)$ (over S), denoted $\text{mods}(KB)$, is the set of instances (over S) $\{I \mid I \supseteq D \wedge I \models \Sigma\}$. The *certain answers* to a (union of) CQs $q(\mathbf{X})$ over a knowledge base $KB = (D, \Sigma)$ are defined as the set of tuples

$$\text{ans}(q(\mathbf{X}), KB) = \bigcap_{I \in \text{mods}(KB)} q(I).$$

A different, however equivalent, way to define ontological query answering in the context of existential rules is via the concept of the *chase* (see, e.g., [12,30,86,106], and references therein).

As it is customary when studying the complexity of computing the answers to a (U)CQ, one focuses on its decision variant: given a knowledge base KB , a (U)CQ $q(\mathbf{X})$, and a tuple of constants \mathbf{t} , decide whether $\mathbf{t} \in \text{ans}(q(\mathbf{X}), KB)$.

For a knowledge base $KB = (D, \Sigma)$ and a UCQ q , the answer to q over KB is *true*, denoted by $KB \models q$, if $\text{ans}(q, KB) = \{\emptyset\}$ (see above). In this case, we also say that D entails q via Σ . The UCQ/BCQ answering problem is the following decision problem: given a knowledge base KB , and a UCQ/BCQ q , decide whether $KB \models q$. If \mathcal{L} is a class of programs (we mention some of these classes below), we denote by $\text{OMQA}(\mathcal{L})$ the problem of UCQ/BCQ answering when the knowledge bases are restricted to those whose programs belong to \mathcal{L} . Since CQ answering can be reduced in logarithmic-space to BCQ answering, we focus on BCQs and UCQs.

This general formulation of the UCQ/BCQ answering problem refers to the *combined* complexity setting of the problem, that is, the database, the program, and the UCQ/BCQ, are considered part of the input [108]. It is common, however, to study also refined complexity measures that are more realistic in practice [108]. Here, we consider the *bounded-arity combined* (or *ba-combined*) complexity, where the arity of the underlying schema is bounded by a fixed integer, the *fixed-program combined* (or *fp-combined*) complexity, where the program is fixed, and the *data complexity*, where the UCQ/BCQ and the program are fixed.

Decidability paradigms. It is known that ontological query answering under arbitrary existential rules is undecidable (see, e.g., [30]). This has led to identifying restrictions on existential rules guaranteeing decidability. The classes of programs/TGDs (or Datalog[±] languages/fragments) that we consider to guarantee the decidability of UCQ/BCQ answering are among the most frequently analyzed in the literature. More specifically, the (syntactic) restrictions here considered are guardedness [30], linearity [28], stickiness [29], and acyclicity, each having a “weak” counterpart: weak guardedness [30], weak stickiness [29], and weak acyclicity [54]—definitions of the just mentioned existential rules languages can also be found collected all together in [81, Sections 3 and 8] and [27, Appendix A].

A TGD σ is *guarded*, if there exists an atom in the body that contains (or “guards”) all the body variables of σ . Although the chase under a set of guarded TGDs does not necessarily terminate, query answering is decidable. This follows from the fact that the result of the chase procedure is “treelike”, or, in other words, has finite treewidth [30]. The class of guarded TGDs, denoted \mathcal{G} , is the family

¹ Notice here that, in what follows, the acronym UCQ refer to unions of BCQs, and *not* to unions of CQs.

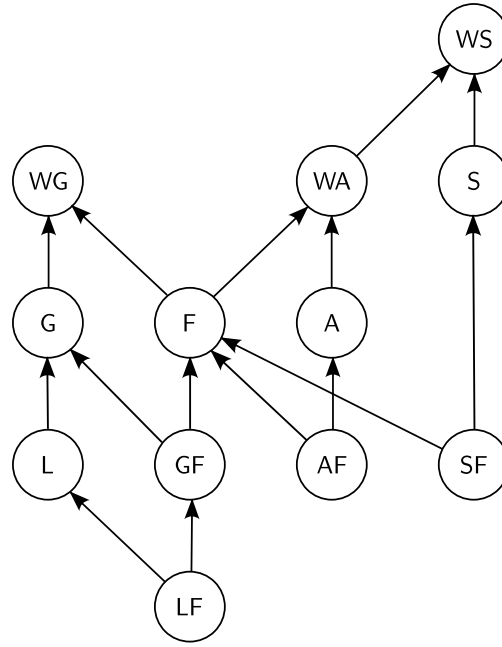


Fig. 1. A schematic representation of the inclusion relationships between the Datalog⁺ fragments considered in this paper. Two fragments X and Y, represented in the figure as two nodes, are such that $X \subset Y$ if and only if there is an arc from X to Y.

of all possible sets of guarded TGDs. A key subclass of guarded TGDs are the *linear* TGDs with just one body atom (which is a guard), and the corresponding class is denoted L.

The *acyclicity* of a set Σ of TGDs is related to the topology of its predicate graph, which is defined as follows: its nodes are the predicates occurring in Σ , and there is an arc from p to r if and only if there is a TGD $\sigma \in \Sigma$ such that p occurs in $body(\sigma)$ and r occurs in $head(\sigma)$. The program Σ is *acyclic* (a.k.a. *non-recursive*) if its predicate graph contains no directed cycles. It is easy to show that acyclicity ensures the termination of the chase [54].

Stickiness is inherently different from guardedness and acyclicity. It ensures neither finite treewidth nor termination of the chase, like for guardedness and acyclicity, respectively. Instead, the decidability of query answering via stickiness is obtained via backward-chaining techniques. The goal underlying stickiness is to express non-guarded statements without forbidding recursion as in acyclicity. The central property of stickiness is as follows: variables that appear more than once in a rule's body (i.e., join variables) are always propagated (or “stick”) to the inferred atoms. A set of TGDs with the above property is *sticky*, and the corresponding class is denoted S.

Another key fragment of TGDs are the *full* TGDs, i.e., TGDs without existentially quantified variables in the rules' heads, and the corresponding class is denoted F. This class of TGDs are essentially plain Datalog rules. Moreover, it is known that the logical core of the RL profile of OWL 2, which is aimed at applications that require efficient reasoning without sacrificing too much expressive power, corresponds to full TGDs. If full TGDs enjoy linearity, guardedness, stickiness, or acyclicity, then we obtain the classes LF, GF, SF, and AF, respectively.

Interestingly, the main classes of TGDs that we have mentioned above, based on the notions of guardedness, acyclicity and stickiness, come with their “weakly” version: *weakly-guarded* (WG) [30], *weakly-acyclic* (WA) [54], and *weakly-sticky* (WS) [29], respectively. The definition of all these “weakly” versions follows the same principle: the underlying syntactic condition is relaxed in such a way that only certain “harmful” variables are taken into account; for details, we refer the reader to the references given above. Notice that the precise definition of these “harmful” variables is not required in this paper, as our results will require us to only directly deal with acyclic and guarded-full programs. In some cases, our results will even hold for knowledge bases with empty programs, i.e., plain relational databases. The results for the other Datalog⁺ fragments mentioned above, will be inherited by results on simpler fragments.

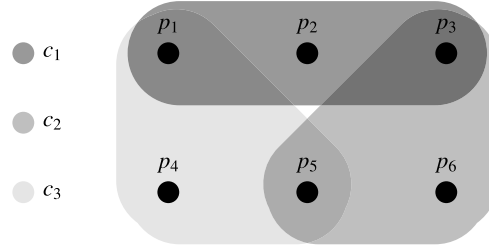
The relationships between the Datalog⁺ fragments here considered are depicted in Fig. 1. Table 1 reports the complexity of UCQ/BCQ answering for the Datalog⁺ fragments and the complexity measures here analyzed [81].

Complexity classes. We assume that the reader has some background in computational complexity theory, including the notions of (oracle) Turing machine, and hardness and completeness of a problem for a complexity class, as can be found in standard textbooks (see, e.g., [68,87]). In what follows, we briefly recall the complexity classes that we encounter in our work. The complexity class PSPACE (resp., P, EXP, and 2EXP) contains all the decision problems that can be solved in polynomial space (resp., polynomial, exponential, and double exponential time) via a deterministic Turing machine. The complexity classes NP and NEXP contain all the decision problems that can be solved in polynomial and exponential time via a non-deterministic Turing machine, respectively, while co-NP and co-NEXP are their complementary classes, where ‘yes’- and ‘no’-instances are interchanged. The complexity class $\Sigma_2^P = NP^{NP}$ (resp., NP^{NEXP}) is the class of decision problems that can be solved in non-deterministic polynomial time using an NP

Table 1

Complexity of ontological UCQ answering under existential rules [81, and references therein]. All non-“in” entries are completeness results. Hardness results hold even on BCQs.

\mathcal{L}	Data	<i>fp</i> -combined	<i>ba</i> -combined	Combined
L, LF, AF	in AC^0	NP	NP	PSPACE
S, SF	in AC^0	NP	NP	EXP
A	in AC^0	NP	NEXP	NEXP
G	P	NP	EXP	2EXP
F, GF	P	NP	NP	EXP
WS, WA	P	NP	2EXP	2EXP
WG	EXP	EXP	EXP	2EXP

**Fig. 2.** Protein network, where each complex c_i contains some of the proteins p_1, \dots, p_6 .

oracle (resp., a NEXP oracle). Interestingly, due to the collapse of the Strong Exponential Hierarchy [62], it was shown that NP^{NEXP} equals P^{NEXP} , which is the class of decision problems that can be solved in polynomial time by a deterministic Turing machine using a NEXP oracle. The complexity class $D^P = NP \wedge co-NP$ (resp., $D^{EXP} = NEXP \wedge co-NEXP$) is the class of decision problems that are an intersection of a problem in NP and a problem in co-NP (resp., a problem in NEXP and a problem in co-NP)—notice here that we are speaking of intersections of languages (remember that, in computational complexity theory, a language is just a set of strings) and *not* of intersections of complexity classes; more precisely, $D^P = \{L \mid (L = L_1 \cap L_2) \wedge (L_1 \in NP) \wedge (L_2 \in co-NP)\}$, and $D^{EXP} = \{L \mid (L = L_1 \cap L_2) \wedge (L_1 \in NEXP) \wedge (L_2 \in co-NEXP)\}$. The complexity class AC^0 contains all languages that are decidable via uniform families of Boolean circuits of polynomial size and constant depth. The inclusion relationships (which are all currently believed to be strict) between the above mentioned complexity classes are shown below²:

$$AC^0 \subseteq P \subseteq NP, co-NP \subseteq D^P \subseteq \Sigma_2^P \subseteq PSPACE \subseteq EXP \subseteq NEXP, co-NEXP \subseteq D^{EXP} \subseteq P^{NEXP} \subseteq 2EXP.$$

3. Explanations for query answers

In this section, we introduce the notion of minimal explanation, and we define the problems that we will analyze. Definitions are followed by examples.

Definition 3.1 (MinEx). For a knowledge base $KB = (D, \Sigma)$ and a query q such that $KB \models q$, an *explanation* for $KB \models q$, or for q w.r.t. KB , is a subset $E \subseteq D$ of facts such that $(E, \Sigma) \models q$.

A *minimal explanation* E , or *MinEx*, for $KB \models q$, or for q w.r.t. KB , is an explanation for $KB \models q$ that is inclusion-minimal, i.e., there is no proper subset $E' \subsetneq E$ that is an explanation for $KB \models q$.

In the following, when the query and/or the knowledge base we refer to are clear from the context, we say that a set E is a MinEx without explicitly mentioning the query and/or the knowledge base.

We provide below a running example that will be used throughout this section to illustrate the various problems studied in this paper. This example is taken from the study of protein networks [73,97].

Example 3.2. We consider the protein containment scenario illustrated in Fig. 2. In this example, there are proteins p_1, \dots, p_6 and complexes c_1, c_2, c_3 , which are sets of proteins. We want to find a minimal subset of the proteins covering all the complexes, i.e., a minimal subset of the proteins having at least one representative from each complex.³

This problem can be expressed as the search for MinExes for a suitable knowledge base and query. Consider a database encoding all the proteins:

² For these inclusion relationships, the notation “ $A \subseteq B, C \subseteq D$ ” is a shorthand for: $A \subseteq B$; $A \subseteq C$; $B \subseteq D$; and $C \subseteq D$. Moreover, no inclusion relationships are currently known for B and C , and it is currently believed that $B \not\subseteq C$ and $C \not\subseteq B$.

³ This example could be adapted to any other task equivalent to finding minimal hypergraphs transversals [57,59].

$$D_{prot} = \{protein(p_i) \mid 1 \leq i \leq 6\}.$$

The following program states which proteins are contained in which complexes according to Fig. 2:

$$\Sigma_{prot} = \{protein(p_i) \rightarrow \bigwedge_{c_j \ni p_i} covered(c_j) \mid 1 \leq i \leq 6\}.$$

We denote by KB_{prot} the knowledge base $(D_{prot}, \Sigma_{prot})$.

The following query asks whether every complex can be covered:

$$q_{prot} = covered(c_1) \wedge covered(c_2) \wedge covered(c_3).$$

Consider now a subset $E \subseteq D_{prot}$. Clearly, $(E, \Sigma_{prot}) \models q_{prot}$ if and only if $(E, \Sigma_{prot}) \models covered(c_i)$ for every complex c_i . Thus, the MinExes for $KB_{prot} \models q_{prot}$ are in bijection with the minimal protein covers of the complexes. \triangleleft

The aim of this work is to analyze the complexity of various tasks associated with MinExes under existential rules. We give here an overview of the six problems considered in this paper, which have been inspired by those studied in axiom pinpointing [94] and logic-based abduction [48]. These problems concern recognising MinExes, determining the roles of facts in MinExes, and the cardinality of MinExes. In what follows, after the definitions of the problems, we provide examples for them based on Example 3.2.

The most fundamental problem that we study is recognising a MinEx for a given query and knowledge base. This is a natural decision version of the problem of computing a minimal explanation.

Problem: IS-MINEX (\mathcal{L})

Input: A knowledge base $KB = (D, \Sigma)$ with $\Sigma \in \mathcal{L}$, a UCQ q with $KB \models q$, and a set of facts $E \subseteq D$.

Output: Is E a MinEx for $KB \models q$?

Example 3.3. The following are *some* MinExes for $KB_{prot} \models q_{prot}$:

$$E_1 = \{protein(p_1), protein(p_3)\},$$

$$E_2 = \{protein(p_2), protein(p_5)\},$$

$$E_3 = \{protein(p_2), protein(p_4), protein(p_6)\}.$$

The above sets are also minimal protein covers of the complexes.

The set $E' = \{protein(p_4), protein(p_5), protein(p_6)\}$ is not a MinEx, as the proteins in E' do not cover all the complexes, and thus $(E', \Sigma_{prot}) \not\models q_{prot}$. A set of facts, together with the program, might entail the query but not be a MinEx. For example, $E'' = \{protein(p_1), protein(p_2), protein(p_3)\}$ is such that $(E'', \Sigma_{prot}) \models q_{prot}$, however E'' is not a MinEx, as it is *not* minimal; indeed, the fact $protein(p_2)$ can be removed without breaking the query entailment. \triangleleft

Another fundamental problem, ALL-MINEX, is the problem of recognising the set of all MinExes for a given query. This problem is the decision version of the problem of computing the set of all MinExes, which is particularly important when we want to gain a complete understanding of how a particular query can be entailed.

Problem: ALL-MINEX(\mathcal{L})

Input: A knowledge base $KB = (D, \Sigma)$ with $\Sigma \in \mathcal{L}$, a UCQ q with $KB \models q$, and a set $\mathcal{E} = \{E_1, \dots, E_n\}$ of subsets of D .

Output: Is \mathcal{E} the set of all and only the MinExes for $KB \models q$?

Example 3.4. The following set of subsets of D_{prot} is exactly the set of all the MinExes for $KB_{prot} \models q_{prot}$:

$$\mathcal{E} = \{ \{protein(p_1), protein(p_3)\}, \{protein(p_1), protein(p_5)\}, \{protein(p_1), protein(p_6)\},$$

$$\{protein(p_2), protein(p_5)\}, \{protein(p_3), protein(p_4)\}, \{protein(p_3), protein(p_5)\},$$

$$\{protein(p_2), protein(p_4), protein(p_6)\} \}.$$

This set corresponds to the set of all and only the minimal protein covers of the complexes. \triangleleft

In some applications, the user might specify some database subsets as “forbidden”. For example, the user might know that some facts are incompatible or might be interested in explanations excluding some specific sets of facts. We study this problem, MINEX-IRREL, of deciding whether there is a MinEx for a query that avoids forbidden sets of facts.

Problem: MINEX-IRREL(\mathcal{L})

Input: A knowledge base $KB = (D, \Sigma)$ with $\Sigma \in \mathcal{L}$, a UCQ q with $KB \models q$, and a set $\mathcal{F} = \{F_1, \dots, F_n\}$ of subsets of D .

Output: Is there a MinEx E for $KB \models q$ such that, for every $F_i \in \mathcal{F}$, $F_i \not\subseteq E$?

Table 2

Complexity results for IS-MINEX(\mathcal{L}). All non-“in” entries are completeness results. Membership results hold for UCQs, while hardness results hold even on BCQs. In square brackets, we report the number of the theorems providing the membership and the hardness results, respectively.

\mathcal{L}	Data	fp -combined	ba -combined	Combined
L, LF, AF	in AC ⁰ [4.2 / –]	D ^P [4.1 / 4.4]	D ^P [4.1 / 4.4]	PSpace [4.1 / 4.6]
S, SF	in AC ⁰ [4.2 / –]	D ^P [4.1 / 4.4]	D ^P [4.1 / 4.4]	EXP [4.1 / 4.6]
A	in AC ⁰ [4.2 / –]	D ^P [4.1 / 4.4]	D ^{Exp} [4.1 / 4.5]	D ^{Exp} [4.1 / 4.5]
G	P [4.1 / 4.3]	D ^P [4.1 / 4.4]	EXP [4.1 / 4.6]	2EXP [4.1 / 4.6]
F, GF	P [4.1 / 4.3]	D ^P [4.1 / 4.4]	D ^P [4.1 / 4.4]	EXP [4.1 / 4.6]
WS, WA	P [4.1 / 4.3]	D ^P [4.1 / 4.4]	2EXP [4.1 / 4.6]	2EXP [4.1 / 4.6]
WG	EXP [4.1 / 4.3]	EXP [4.1 / 4.6]	EXP [4.1 / 4.6]	2EXP [4.1 / 4.6]

Example 3.5. Suppose that we are aware of configurations of proteins that are not allowed to be together in a cover:

$$\mathcal{F} = \{\{protein(p_1)\}, \{protein(p_3), protein(p_5)\}, \\ \{protein(p_2), protein(p_4), protein(p_6)\}\}.$$

In this case, $\{protein(p_3), protein(p_4)\}$ is a desirable MinEx that does not contain any set from \mathcal{F} and covers all the complexes. On the other hand, observe that neither $\{protein(p_1), protein(p_3)\}$ nor $\{protein(p_3), protein(p_5)\}$ are desirable MinExes, as they both contain some forbidden set. \triangleleft

Another problem, MINEX-REL, is the problem of deciding the relevance of a fact in explaining the entailment of a given query. We say that a fact ψ is *relevant* for $KB \models q$ if there is a MinEx E for $KB \models q$ with $\psi \in E$. This problem is of interest when we want to decide whether a particular fact *can* contribute to the query entailment.

Problem: MINEX-REL(\mathcal{L})

Input: A knowledge base $KB = (D, \Sigma)$ with $\Sigma \in \mathcal{L}$, a UCQ q with $KB \models q$, and a fact $\psi \in D$.

Output: Is ψ relevant for $KB \models q$, i.e., is there a MinEx E for $KB \models q$ such that $\psi \in E$?

Example 3.6. Suppose that we are interested in knowing whether the fact $\psi = protein(p_6)$ is contained in some MinEx. The sets $\{protein(p_1), protein(p_6)\}$ and $\{protein(p_2), protein(p_4), protein(p_6)\}$ are MinExes for $KB_{prot} \models q_{prot}$ containing this fact. Therefore, $\psi = protein(p_6)$ is relevant for $KB_{prot} \models q_{prot}$, and, in particular, there is a minimal protein cover containing the protein p_6 . \triangleleft

We also study two cardinality-based problems for MinExes. These problems can be seen as the decision versions of the problems of computing the size of the smallest and largest MinExes. The first problem, SMALL-MINEX, asks whether there is a MinEx whose size is smaller than some given threshold, while the second, LARGE-MINEX, asks whether there is a MinEx whose size is larger than some given threshold.

Problem: SMALL-MINEX(\mathcal{L})

Input: A knowledge base $KB = (D, \Sigma)$ with $\Sigma \in \mathcal{L}$, a UCQ q with $KB \models q$, and an integer $n \geq 1$ represented in binary.

Output: Is there a MinEx E for $KB \models q$ such that $|E| \leq n$?

Problem: LARGE-MINEX(\mathcal{L})

Input: A knowledge base $KB = (D, \Sigma)$ with $\Sigma \in \mathcal{L}$, a UCQ q with $KB \models q$, and an integer $n \geq 1$ represented in binary.

Output: Is there a MinEx E for $KB \models q$ such that $|E| \geq n$?

Example 3.7. Observe that every MinEx for $KB_{prot} \models q_{prot}$ is either of size 2 or 3 (see Example 3.4). For example, $\{protein(p_1), protein(p_3)\}$ is a MinEx of size 2 and $\{protein(p_2), protein(p_4), protein(p_6)\}$ is a MinEx of size 3, which are a smallest and a largest MinEx for $KB_{prot} \models q_{prot}$, respectively. \triangleleft

4. Recognizing minimal explanations

In this section, we start our complexity analysis by considering the fundamental problem IS-MINEX: for an instance (KB, q, E) , where $KB = (D, \Sigma)$ is a knowledge base, q is a UCQ (with $KB \models q$), and $E \subseteq D$ is a set of facts, decide whether E is a MinEx for $KB \models q$. A summary of the complexity results for this problem is reported in Table 2.

Before delving into the details of the complexity analysis of IS-MINEX, we highlight some of interesting results that we obtain in this section. We are able to show that IS-MINEX(A) is D^{Exp}-complete in the ba -combined and combined complexity (Theorem 4.5). Recall that D^{Exp} = NEXP \cap co-NEXP = $\{L \mid (L = L_1 \cap L_2) \wedge L_1 \in \text{NEXP} \wedge L_2 \in \text{co-NEXP}\}$. To our knowledge, IS-MINEX(A) is among the very few, if not the first, natural problem to be shown complete for D^{Exp}. We also prove that deciding whether a set of database

Algorithm 1: A general algorithm for the IS-MINEX problem.

Input: A knowledge base $KB = (D, \Sigma)$, a UCQ q , and a set $E \subseteq D$ of facts.
Output: accept, if E is a MinEx for $(D, \Sigma) \models q$; reject, otherwise.

/ The statement “verify ψ ” is equivalent to “if not ψ then return reject; whereas the statement “verify not ψ ” is equivalent to “if ψ then return reject” */*

Procedure IsMinex(KB, q, E):

```

1  verify ( $E, \Sigma$ )  $\models q$ 
2  foreach  $\alpha \in E$  do
3    [ verify not ( $E \setminus \{\alpha\}, \Sigma$ )  $\models q$ 
4  return accept
```

facts is a minimal set entailing a BCQ is already D^P -hard for plain relations databases when the query is assumed not to be fixed (Theorem 4.4), i.e., there is no need to have a program to make the problem difficult to solve. Furthermore, on the side of the results of feasibility, we show that, in the data complexity, deciding IS-MINEX for the FO-rewritable Datalog[±] fragments is FO-rewritable (Theorem 4.2), and hence also feasible in polynomial time.

We begin by proving the membership results for this problem. These results are obtained via a general algorithm, Algorithm 1, tackling IS-MINEX (details and comments about the algorithm will be provided in the proof of Theorem 4.1).

The algorithms presented in this paper are described via pseudocode in which we may use the keywords “**verify**”/“**verify not**”. The syntax and the semantics adopted in this paper for the statement “**verify** ψ ” (resp., “**verify not** ψ ”) are as follows: ψ is a Boolean expression that can be built by Boolean values, Boolean tests, and calls to Boolean functions, linked by Boolean logic connectives; the statement “**verify** ψ ” (resp., “**verify not** ψ ”) is expanded to “**if not** ψ **then return reject**” (resp., “**if** ψ **then return reject**”). Keywords “**verify**” and “**verify not**” can straight after also be followed by a **begin/end**-enclosed code-block (see, e.g., Algorithm 2), which defines a(n anonymous) Boolean function with no arguments and having access to all the variables in the outer function’s scope; this inline-defined Boolean function is invoked to evaluate the Boolean expression associated with the preceding “**verify**” or “**verify not**”. We underline here that, when there is a sequence of *non*-nested “**verify**”/“**verify not**” statements in an algorithm (see, e.g., lines 2 and 3 in Algorithm 3), these are simply evaluated one after the other in the order specified, and the execution is interrupted only if one of them fails by returning reject at the corresponding line.

Going back to the general procedure for the IS-MINEX problem, thanks to the use of an OMQA oracle, Algorithm 1 provides us the upper bounds for IS-MINEX for all the classes of TGDs and the complexity measures considered here. The resulting upper bounds are those reported in Table 2 and are tight, except for those in the data complexity for the cases in which OMQA is FO-rewritable (namely, the languages $\mathcal{L}, \mathcal{S}, \mathcal{A}$, and their specializations), for which the tighter upper bounds reported in the table will require a stronger result stated later. A key insight of Algorithm 1 is that negation is not allowed in the programs, and hence entailment is monotonic in the framework here studied. For this reason, *non*-entailment is preserved by removing elements from the database. Indeed, for a knowledge base (D, Σ) and a query q , if a set $S \subseteq D$ does not entail q via Σ , then every subset S' of S does not entail q via Σ either. This implies that, to verify the inclusion-minimality of an explanation E , it suffices to check that removing any *single* fact from E yields a set that does *not* entail the query via the program; hence, we do not need to carry out this check for all the possible (exponentially-many) subsets of E .

Theorem 4.1. *For every class \mathcal{L} of TGDs considered in this paper, if OMQA(\mathcal{L}) is in the complexity class \mathbf{C} in the combined (resp., *ba*-combined, *fp*-combined, and *data*) complexity, then IS-MINEX(\mathcal{L}) can be decided by a check in \mathbf{C} and a linear number of checks in *co*- \mathbf{C} in the combined (resp., *ba*-combined, *fp*-combined, and *data*) complexity.*

Proof. Let $KB = (D, \Sigma)$ be a knowledge base with Σ belonging to \mathcal{L} , let q be UCQ, and let E be a subset of D . A general approach to test whether E is a MinEx for $KB \models q$ is outlined in Algorithm 1. In particular, we need to check whether (i) the knowledge base (E, Σ) entails q (line 1), which essentially tests whether E is an explanation; and whether (ii) the just-ascertained explanation E is actually subset-minimal (lines 2 and 3).

The entailment test at line 1 is a single check in the complexity class \mathbf{C} . To test the subset-minimality of E , by the entailment monotonicity of the considered framework, it suffices to check whether removing any single element α from E results into a set such that $(E \setminus \{\alpha\}, \Sigma)$ does *not* entail q (rather than checking this for all subsets of E). Therefore, we need to perform $|E|$ -many *non*-entailment tests, i.e., a linear number of checks, each of them in *co*- \mathbf{C} . \square

The membership results in Table 2 obtained from the previous theorem descend also from the following computational complexity facts: NP and NEXP are complexity classes closed under conjunction, hence a linear number of *co*-NP and *co*-NEXP checks can be combined into a single check in *co*-NP and *co*-NEXP, respectively; additionally, $\text{NP} \wedge \text{co-NP} = D^P$ and $\text{NEXP} \wedge \text{co-NEXP} = D^{\text{EXP}}$.

Next, we show that IS-MINEX is in AC^0 in the data complexity for FO-rewritable languages. This is obtained by showing that the IS-MINEX problem, in these cases, can be encoded into a FO formula, which can be evaluated in AC^0 in the data complexity. The intuition behind this proof is showing that a tailored FO formula can both encode that a set of facts F entails the query and that none of the facts can be removed from F without losing the entailment property.

Theorem 4.2. *For every class \mathcal{L} of TGDs considered in this paper such that \mathcal{L} is FO-rewritable, IS-MINEX(\mathcal{L}) is in AC^0 in the data complexity.*

Proof. We will show that $\text{IS-MINEX}(\mathcal{L})$, when \mathcal{L} is FO-rewritable, can be expressed via an FO query. By this, because answering FO queries is in AC^0 in the data complexity [109], $\text{IS-MINEX}(\mathcal{L})$ will result to be in AC^0 in the data complexity, as well, when \mathcal{L} is FO-rewritable. To prove this we will use the following fact that we remind to the reader: If \mathcal{L} is a FO-rewritable class of TGDs, then, for each program $\Sigma \in \mathcal{L}$ and each UCQ q over a relational schema S , there is a FO query/formula $\phi_{q,\Sigma}$ such that, for every set F of facts over S , $(F, \Sigma) \models q$ if and only if $F \models \phi_{q,\Sigma}$. Note here that, since q is a UCQ, all variables of q are existentially quantified, and hence there are no free variables in $\phi_{q,\Sigma}$ either.

As a first step, we prove that testing the subset-minimality of explanations for q w.r.t. a knowledge base over S can be encoded into a FO formula ϕ_{\min} . We will show how ϕ_{\min} is defined by considering its constituent pieces.

Let us first focus on a generic n -ary predicate p from S . For a set $\mathbf{X}_p = \{X_1, \dots, X_n\}$ of fresh variables, consider the FO formula $\phi_{q,\Sigma}^{\neq p}(\mathbf{X}_p)$ obtained from $\phi_{q,\Sigma}$ as follows: all occurrences in $\phi_{q,\Sigma}$ of p -atoms $p(\mathbf{t})$, where \mathbf{t} is an n -tuple of constants and variables, are substituted with the expression $(p(\mathbf{t}) \wedge (p(\mathbf{t}) \neq p(X_1, \dots, X_n)))$. Note that, on the contrary of what happens for $\phi_{q,\Sigma}$, the FO formula $\phi_{q,\Sigma}^{\neq p}(\mathbf{X}_p)$ has free variables, namely, $\mathbf{X}_p = \{X_1, \dots, X_n\}$.

Observe that, for an n -tuple \mathbf{c} of constants, $\phi_{q,\Sigma}^{\neq p}(\mathbf{c})$ is such that, for every set F of facts over S , $F \models \phi_{q,\Sigma}^{\neq p}(\mathbf{c})$ if and only if $(F \setminus \{p(\mathbf{c})\}, \Sigma) \models q$, because the portions $(p(\mathbf{t}) \wedge (p(\mathbf{t}) \neq p(X_1, \dots, X_n)))$ of the formula $\phi_{q,\Sigma}^{\neq p}(\mathbf{X}_p)$ can be satisfied only by p -atoms different from $p(\mathbf{c})$. This implies that $F \models \phi_{q,\Sigma}^{\neq p}(\mathbf{c})$ if and only if $F \setminus \{p(\mathbf{c})\}$ is an explanation for q w.r.t. (F, Σ) , and, symmetrically, we have that $F \models \neg \phi_{q,\Sigma}^{\neq p}(\mathbf{c})$ if and only if $F \setminus \{p(\mathbf{c})\}$ is not an explanation for q .

We are almost ready to define the FO formula ϕ_{\min} , which tests the minimality of explanations for q . To this aim, consider now the FO formula $\phi^p = \forall \mathbf{X}_p (p(\mathbf{X}_p) \rightarrow \neg \phi_{q,\Sigma}^{\neq p}(\mathbf{X}_p))$, built upon the formula $\phi_{q,\Sigma}^{\neq p}(\mathbf{X}_p)$ seen above. From what we have just observed about $\neg \phi_{q,\Sigma}^{\neq p}(\mathbf{c})$, the FO formula ϕ^p is satisfied by those sets F of facts over S such that, if F contains any generic p -atom $p(\mathbf{c})$, then $F \setminus \{p(\mathbf{c})\}$ is not an explanation for q . We can now define $\phi_{\min} = \bigwedge_{p \in S} \phi^p$. From our previous discussion, a set F of facts over S satisfies ϕ_{\min} if and only if F is such that the removal of *each* single fact from F yields a fact set that is not an explanation for q (observe here that ϕ_{\min} does *not* test for F to be an explanation for q ; ϕ_{\min} just tests that, for any atom $\alpha \in F$, $F \setminus \{\alpha\}$ is *not* an explanation for q).

To conclude, let us consider the FO formula $\phi = \phi_{q,\Sigma} \wedge \phi_{\min}$. Observe that, to satisfy ϕ , a set F of facts over S must be an explanation for q , as $\phi_{q,\Sigma}$ is part of ϕ . Hence, $F \models \phi$ if and only if F is an explanation for q that is not an explanation for q any longer when any of its fact is removed, i.e., F must be a MinEx for q . \square

In the rest of this section, we prove lower bounds for IS-MINEX. We start by showing that, when $\mathcal{L} \supseteq \text{GF}$, $\text{IS-MINEX}(\mathcal{L})$ is at least as hard as $\text{OMQA}(\mathcal{L})$, i.e., there is a reduction from $\text{OMQA}(\mathcal{L})$ to $\text{IS-MINEX}(\mathcal{L})$. This theorem proves all the P-hardness results reported in Table 2 and all the hardness results (in the data, *fp*-combined, *ba*-combined, and combined, complexity) for the WG fragment—the following statement actually covers also some hardness results shown in Table 2 for the *ba*-combined and combined complexity (namely, for G, WS, and WA); however, another statement, needed to cover the remaining cases in the *ba*-combined and combined complexity, will cover these results as well. The intuition behind the reduction is translating an OMQA instance α into an IS-MINEX instance β , such that the set of *all* facts in the database of α is a MinEx for a suitable modified query in β . In this way, if the database of α does not entail the original query in α , then that set of facts is not an explanation (at all) of the query in β .

Theorem 4.3. *For every class \mathcal{L} of TGDs considered in this paper such that $\mathcal{L} \supseteq \text{GF}$, $\text{IS-MINEX}(\mathcal{L})$ is at least as hard as $\text{OMQA}(\mathcal{L})$ in the data (resp., *fp*-combined, *ba*-combined, and combined) complexity. Hardness holds even on instances whose queries are BCQs.*

Proof. We prove the statement by showing a *logspace* reduction (so for the reduction to hold also for the P-hardness results) from $\text{OMQA}(\mathcal{L})$, where $\mathcal{L} \supseteq \text{GF}$. Given an $\text{OMQA}(\mathcal{L})$ instance (KB, q) , where $KB = (D, \Sigma)$ is a knowledge base, and q is a BCQ, we build in logspace, as explained below, an instance (KB', q', E) of $\text{IS-MINEX}(\mathcal{L})$, where $KB' = (D', \Sigma')$ is a knowledge base, q' is a BCQ, and E is a candidate MinEx.

(The database). We obtain D' from D and q as follows. For each fact $p(\mathbf{a}) \in D$, D' contains two facts $p'(\mathbf{a}, i)$ and $r(i-1, i)$, where p' and r are fresh predicate symbols, with the former associated with p , and i is an incremental integer constant (over all facts) starting from 1 ($'i-1'$ is just an expression indicating the integer constant preceding i). These integer values are used to “enumerate” all the facts in D' coming from D (i.e., different facts in D' coming from D have different integer values i in the last position). The predicate r intuitively states that the integer constant $'i'$ is the subsequent number represented by the integer constant $'i-1'$. Further, D' contains two additional facts $u(0)$ and $\text{num-facts}(|D|)$, both of them over fresh predicate symbols. To conclude, D' contains the facts of a full grounding of q , where each variable in q is grounded with a fresh constant, and we call this set of facts D_q ; this ensures that D' entails q .

(The program). We build the program Σ' starting from Σ as follows. The program Σ' contains all the rules of Σ . Moreover, for each n -ary predicate p of the relational schema, Σ' contains the rules

$$p'(\mathbf{X}, J) \rightarrow p(\mathbf{X}), \quad p'(\mathbf{X}, J) \rightarrow t(J),$$

where $\mathbf{X} = \{X_1, \dots, X_n\}$ is a set of variables, and t is a fresh predicate symbol. Furthermore, Σ' contains two other rules

$$u(I) \wedge r(I, J) \wedge t(J) \rightarrow u(J), \quad u(J) \wedge \text{num-facts}(J) \rightarrow \text{all}(),$$

ensuring that the fresh fact $all()$ can be derived from a set of facts F only if all p' -facts are in F .

(The query). The query is $q' = q \wedge all()$.

(The candidate MinEx). We take the set $E = D' \setminus D_q$.

Observe that this instance of IS-MINEX can be computed in logarithmic space. Notice that, like Σ , we have that $\Sigma' \in \mathcal{L}$, because $\Sigma \in \mathcal{L}$ and the additional rules are guarded and full (remember that we are assuming $\mathcal{L} \supseteq \text{GF}$). Moreover, Σ' is built independently from D , hence, in the data and the fp -combined complexity settings, where the relational schema is fixed, Σ' is fixed as well. For the ba -combined complexity setting, observe that the fresh predicates have bounded arity, hence, when Σ is over a relational schema with bounded arity, also Σ' is over a relational schema with bounded arity. Observe that q' depends on q and does not depend on D ; hence, when q is fixed, also q' is fixed. Moreover, since q is a BCQ, q' is a BCQ, as well. We further notice that $(D', \Sigma') \models q'$, because D_q is part of D' .

We now show that $(D, \Sigma) \models q$ if and only if E is a MinEx for $(D', \Sigma') \models q'$.

(\Rightarrow) Suppose that $(D, \Sigma) \models q$. Observe that, via the rules ' $p'(\mathbf{X}, J) \rightarrow p(\mathbf{X})$ ' in Σ' , all the facts present in D can be derived from E . This implies that $(E, \Sigma') \models q$. By construction of E , we have that $(E, \Sigma') \models all()$, and thus $(E, \Sigma') \models q'$, which means that E is an explanation for q' w.r.t. (D', Σ') . Moreover, E is a subset-minimal explanation for q' , because, by construction, removing any fact from E would break the derivation of the fact $all()$.

(\Leftarrow) Suppose that E is a MinEx for $(D', \Sigma') \models q'$. Since $q' = q \wedge all()$, it must be the case that $(E, \Sigma') \models q$, as well. Observe that q involves only *non*-primed predicate symbols (i.e., the p -predicates, and not the p' -predicates, for every predicate symbol p), and none of the fresh predicate symbols introduced to build D' and Σ' appears in q . Hence, the facts needed to satisfy the ' q ' part within the query q' are only the non-primed predicate facts that can be derived from E via Σ' . Notice that the non-primed predicate facts derived from E via Σ' are contained in D . Moreover, since all the rules in Σ' not involving primed or fresh predicate symbols are also in Σ , we have that $(D, \Sigma) \models q$. \square

Next, we show that IS-MINEX(\mathcal{L}) is D^P -hard in the fp -combined complexity, even when the program is empty. This proves all the D^P -hardness results in Table 2. In the proof of this theorem, the classical SAT-UNSAT problem is used as the starting problem in our reduction. This is different from what was done in the proof of the D^P -hardness of IS-MINA in [94, Theorem 4], in which a tailored D^P -complete problem is used for the reduction. By using the canonical problem SAT-UNSAT, recognizing from our proof the two sources of complexity for the problem IS-MINEX is immediate. Indeed, from our proof it is evident that the NP-hardness part of the D^P -hardness is obtained by linking the satisfiability check of a Boolean formula ϕ to checking whether a suitable set E of facts is an explanation of a tailored query, whereas the co-NP-hardness part of the D^P -hardness is obtained by associating the unsatisfiability check of a Boolean formula ψ with checking whether the mentioned explanation E is also minimal.

Theorem 4.4. *For every class \mathcal{L} of TGDs considered in this paper, IS-MINEX(\mathcal{L}) is D^P -hard in the fp -combined (resp., ba -combined) complexity. Hardness holds even on instances whose programs are empty and whose queries are BCQs.⁴*

Proof. We show a reduction to IS-MINEX(\mathcal{L}) from the D^P -complete problem SAT-UNSAT [88]: for an instance $(\phi(\bar{x}), \psi(\bar{y}))$, where $\phi(\bar{x})$ and $\psi(\bar{y})$ are two 3CNF Boolean formulas defined over the Boolean variables sets \bar{x} and \bar{y} , respectively, decide whether $\phi(\bar{x})$ is satisfiable and $\psi(\bar{y})$ is unsatisfiable.

Let $(\phi(\bar{x}), \psi(\bar{y}))$ be an instance of SAT-UNSAT, where $\bar{x} = \{x_1, \dots, x_n\}$ and $\bar{y} = \{y_1, \dots, y_m\}$. From $(\phi(\bar{x}), \psi(\bar{y}))$, we build as follows an instance (KB, q, E) of IS-MINEX(\mathcal{L}), where $KB = (D, \Sigma)$ is a knowledge base, q is a BCQ, and E is a candidate MinEx. We will encode the (un)satisfiability of the formulas ϕ and ψ via the database D and the query q .

(The database). The database D contains facts that will be used to impose the consistency of truth assignments, mimicked by mappings from q to D , to the *literals* of the formulas $\phi(\bar{x})$ and $\psi(\bar{y})$:

$$\begin{array}{lll} simlit(f, f), & opplit(f, t), & simlit(\star, \star), \\ simlit(t, t), & opplit(t, f), & opplit(\star, \star), \end{array}$$

where t and f are constants associated with the Boolean values *true* and *false*, respectively, and \star is an additional constant used as a “jolly”. Intuitively, the “jolly” constant allows to bypass the Boolean formulas’ satisfiability checks.

Further, D contains facts capturing the satisfying assignments to the *literals* for a clause:

$$\begin{array}{ll} clsat_\phi(\tau_1, \tau_2, \tau_3), & clsat_\phi(\star, \star, \star), \\ clsat_\psi(\tau_1, \tau_2, \tau_3), & clsat_\psi(\star, \star, \star), \end{array}$$

where each $\tau_i \in \{f, t\}$ and at least one of τ_i for $i \in \{1, 2, 3\}$ is t . Note that in D there are 7 $clsat_\phi$ -facts and 7 $clsat_\psi$ -facts. Intuitively, the facts with the “jolly” constant allow to bypass the satisfiability checks of ϕ and ψ .

⁴ Notice that this hardness result holds already over plain relational databases: given a set E of facts and a BCQ q (both E and q are part of the input and may vary), deciding whether E is a minimal set of facts entailing q is D^P -hard.

(The program). We take Σ to be the empty set.

(The query). For presentation purposes, we gradually introduce the query q by giving intuitions for each piece defined. In what follows, $\ell_{j,k}^\alpha$ is the k^{th} literal in the j^{th} clause of the formula $\alpha \in \{\phi, \psi\}$, and $\ell_{j,k}^\alpha$ is the Boolean variable of $\ell_{j,k}^\alpha$.

A first query piece imposes that the query variables $T_{j,k}^\phi$ and $T_{j,k}^\psi$, associated with the formula literals $\ell_{j,k}^\phi$ and $\ell_{j,k}^\psi$, respectively, are mapped to consistent constants; in this way, we simulate that the assignments to the formula literals, derived by the mappings of the query atoms to the database facts, are consistent. This part of the query is:

$$\text{consist} \equiv \bigwedge_{\alpha \in \{\phi, \psi\}} \left(\bigwedge_{\substack{\text{for all pairs of literals} \\ (\ell_{j,k}^\alpha, \ell_{j',k'}^\alpha) \text{ in } \alpha \text{ s.t.} \\ \ell_{j,k}^\alpha = \ell_{j',k'}^\alpha}} \text{simlit}(T_{j,k}^\alpha, T_{j',k'}^\alpha) \wedge \bigwedge_{\substack{\text{for all pairs of literals} \\ (\ell_{j,k}^\alpha, \ell_{j',k'}^\alpha) \text{ in } \alpha \text{ s.t.} \\ \ell_{j,k}^\alpha = \neg \ell_{j',k'}^\alpha}} \text{opplit}(T_{j,k}^\alpha, T_{j',k'}^\alpha) \right).$$

A second piece of the query captures the satisfiability of ϕ and ψ :

$$\text{satisfied} \equiv \bigwedge_{\substack{\text{for all clauses} \\ c_j \text{ of } \phi}} \text{clsat}_\phi(T_{j,1}^\phi, T_{j,2}^\phi, T_{j,3}^\phi) \wedge \bigwedge_{\substack{\text{for all clauses} \\ c_j \text{ of } \psi}} \text{clsat}_\psi(T_{j,1}^\psi, T_{j,2}^\psi, T_{j,3}^\psi).$$

We define the query as: $q = D_{st} \wedge \text{consist} \wedge \text{satisfied}$, where with the notation D_{st} in the query we mean the conjunction of all the structural facts from $D_{st} = D \setminus \{\text{clsat}_\phi(\star, \star, \star), \text{clsat}_\psi(\star, \star, \star)\}$.

(The candidate MinEx). We take $E = D_{st} \cup \{\text{clsat}_\psi(\star, \star, \star)\} = D \setminus \{\text{clsat}_\phi(\star, \star, \star)\}$.

The reduction can be computed in polynomial time, and is such that $(D, \Sigma) \models q$, because $\text{clsat}_\phi(\star, \star, \star)$ and $\text{clsat}_\psi(\star, \star, \star)$ are in D ; indeed, assigning the value \star to all the query variables $T_{j,k}^\alpha$ satisfies the query. Furthermore, the empty program Σ is vacuously linear (and guarded), acyclic, sticky, and full; clearly, such a program is also fixed.

We now show that $(\phi(\bar{x}), \psi(\bar{y}))$ is a ‘yes’-instance of SAT-UNSAT if and only if E is a MinEx for $(D, \Sigma) \models q$.

(\Rightarrow) Suppose that $(\phi(\bar{x}), \psi(\bar{y}))$ is a ‘yes’-instance of SAT-UNSAT, i.e., $\phi(\bar{x})$ is satisfiable and $\psi(\bar{y})$ is unsatisfiable. Assume by contradiction that E is not a MinEx for $(D, \Sigma) \models q$. Then, either $(E, \Sigma) \not\models q$ (i.e., E is not an explanation at all) or E is a non-minimal explanation. Let us consider the case when E is not an explanation. Because the inclusion of $\text{clsat}_\psi(\star, \star, \star)$ in E guarantees that all clsat_ψ atoms in the query are satisfied, it must be the case that the query is not satisfied by E via Σ because $\phi(\bar{x})$ is unsatisfiable: a contradiction. Let us now consider the case when E is a non-minimal explanation. Since the D_{st} part of the query requires for its satisfaction that all facts but $\text{clsat}_\psi(\star, \star, \star)$ are in every explanation, if E is an explanation that is not minimal, then it must be the case that $((E \setminus \{\text{clsat}_\psi(\star, \star, \star)\}), \Sigma) \models q$. This implies that $\psi(\bar{y})$ is satisfiable: a contradiction. Therefore, E is a MinEx for $(D, \Sigma) \models q$.

(\Leftarrow) First, observe that the query is built so to be entailed by a set F of facts only if $F \supseteq D_{st}$, and, when $\phi(\bar{x})$ is unsatisfiable, the set of facts F has to contain the fact $\text{clsat}_\phi(\star, \star, \star)$ as well.

Suppose now that $(\phi(\bar{x}), \psi(\bar{y}))$ is a ‘no’-instance of SAT-UNSAT, i.e., $\phi(\bar{x})$ is unsatisfiable or $\psi(\bar{y})$ is satisfiable. Let us consider the case in which $\phi(\bar{x})$ is unsatisfiable. From our prior observation, E does not entail the query, as E does not include $\text{clsat}_\phi(\star, \star, \star)$. Hence, E is not an explanation. Let us now consider the case in which $\phi(\bar{x})$ is satisfiable. Since $(\phi(\bar{x}), \psi(\bar{y}))$ is a ‘no’-instance of SAT-UNSAT, it must be the case that $\psi(\bar{y})$ is satisfiable too. By this, $((E \setminus \{\text{clsat}_\psi(\star, \star, \star)\}), \Sigma) \models q$, as the fact $\{\text{clsat}_\psi(\star, \star, \star)\}$ is not required for the satisfaction of the query part encoding the satisfiability of $\phi(\bar{x})$ (by definition of the query) or for the query part encoding the satisfiability of $\psi(\bar{y})$ (as $\psi(\bar{y})$ is satisfiable in this case). Therefore, E is an explanation, but in this case it is not minimal. \square

We are left to focus on the remaining hardness results in the *ba*-combined and combined complexity. In the following, we show the D^{Exp} -hardness of IS-MINEX(A) in the *ba*-combined complexity; this statement provides the two D^{Exp} -hardness results reported in Table 2. The result is obtained via a reduction from a D^{Exp} -complete problem that is a variation of the exponential tiling problem. Intuitively, in the result below, the NEXP-hardness component of the D^{Exp} -hardness is obtained by linking the existence of a suitable exponential-square tiling for a tiling system to checking whether a set of facts E is an explanation for a specific query, and the co-NEXP-hardness component of the D^{Exp} -hardness is obtained by linking the non-existence of a suitable exponential-square tiling for a different tiling system to checking whether the proposed set E is moreover a minimal explanation.

To more formally define the exponential tiling problem, we start by defining its instances. A tiling system $\mathcal{T} = (T, V, H)$ is formed by a set of tile-types $T = \{t_1, \dots, t_k\}$, and by the vertical and horizontal adjacency rules $V \subseteq T \times T$ and $H \subseteq T \times T$, respectively [55]—the rules are simply represented by pairs of tile-types that can contiguously be placed vertically or horizontally, respectively, on the surface. A tiling for \mathcal{T} is a function $f : \{1, \dots, i, \dots\} \times \{1, \dots, j, \dots\} \mapsto T$ placing a tile of the specified type at position (i, j) on the surface, such that $(f(i, j), f(i+1, j)) \in H$ and $(f(i, j), f(i, j+1)) \in V$; i.e., f complies with the horizontal and vertical adjacency rules. A tiling problem asks, for a tiling system, whether it admits a tiling covering the entirety of a given surface.

The following problem is NEXP-complete [55]: for an instance (\mathcal{T}, t, n) , where $\mathcal{T} = (T, V, H)$ is a tiling system, $t \in T$ is a tile-type, and n is an integer in unary notation, decide whether \mathcal{T} admits a tiling of the exponential square $2^n \times 2^n$ (i.e., a finite surface) that places a tile of type t at the first position of the first row.

The Exponential Tiling Problem (EXP-TILING) [51, Theorem 15] and [81, Lemma 3.2] here considered is a variation of the one above, from which EXP-TILING inherits its NEXP-hardness: for an instance (\mathcal{T}, s, n) , where \mathcal{T} is a tiling system, s is an initial tiling

condition, which is a sequence of tiles with $s[i]$ denoting the i -th element of s , and n is an integer in unary notation, decide whether \mathcal{T} admits a tiling of the exponential square $2^n \times 2^n$ that places a tile of type $s[i]$ at the i -th position of the first row.

To obtain our D^{Exp} -hardness results, we need however a reduction from a different problem, the *Exponential Tiling-Non-Tiling Problem* EXP-TNT, that we here introduce: for an instance $((\mathcal{T}_1, s_1, n_1), (\mathcal{T}_2, s_2, n_2))$, where $\text{ExpT}_1 = (\mathcal{T}_1, s_1, n_1)$ and $\text{ExpT}_2 = (\mathcal{T}_2, s_2, n_2)$ are two (completely independent) instances of EXP-TILING, decide whether ExpT_1 and ExpT_2 are a ‘yes’-instance and a ‘no’-instance of EXP-TILING, respectively. The D^{Exp} -hardness of EXP-TNT follows from the NEXP-completeness of EXP-TILING and from the fact that the instances ExpT_1 and ExpT_2 are completely independent, and hence they can encode any language in $D^{\text{Exp}} = \text{NEXP} \wedge \text{co-NEXP}$.⁵

Theorem 4.5. IS-MINEX(A) is D^{Exp} -hard in the *ba-combined* (resp., *combined*) complexity. Hardness holds even on instances whose queries are BCQs.

Proof. To prove the statement, we exhibit a reduction from the prototypical D^{Exp} -complete *Exponential Tiling-Non-Tiling Problem* (EXP-TNT). From an instance $((\mathcal{T}_1, s_1, n_1), (\mathcal{T}_2, s_2, n_2))$ of EXP-TNT, we build as follows an instance (KB, q, E) of IS-MINEX(A), where $KB = (D, \Sigma)$ is a knowledge base, q is a BCQ, and E is a candidate MinEx.

For the two instances $\text{ExpT}_1 = (\mathcal{T}_1, s_1, n_1)$ and $\text{ExpT}_2 = (\mathcal{T}_2, s_2, n_2)$ of EXP-TILING, we use the TGD and database encoding presented in [51, Theorem 15], [52, Lemma 23], and [81, Lemma 3.2]. More specifically, we have two sets of bounded-arity TGDs $\Sigma_{\mathcal{T}_i, n_i, |s_i|}$, for $i = 1, 2$, to encode the structure of the tiling task, that is, describe how a correct tiling looks like when covering the exponential square $2^{n_i} \times 2^{n_i}$ and has an initial condition of length $|s_i|$; and two sets $D_{\mathcal{T}_i}$ of facts, for $i = 1, 2$, to encode the adjacency rules of \mathcal{T}_1 and \mathcal{T}_2 , respectively. There are also the database facts D_{s_i} , for $i = 1, 2$, encoding the initial tiling conditions s_1 and s_2 , respectively. The two sets of TGDs and the database facts for ExpT_1 and ExpT_2 are made disjoint by using disjoint predicates (we index them with superscript i). That is, we have $\Sigma^1 = \Sigma_{\mathcal{T}_1, n_1, |s_1|}^1$, $D^1 = D_{\mathcal{T}_1}^1 \cup D_{s_1}^1$, and $\Sigma^2 = \Sigma_{\mathcal{T}_2, n_2, |s_2|}^2$, $D^2 = D_{\mathcal{T}_2}^2 \cup D_{s_2}^2$. The rules Σ^i and the database D^i , for $i = 1, 2$, are such that the tiling system \mathcal{T}_i admits a tiling of the exponential square $2^{n_i} \times 2^{n_i}$ with initial tiling condition s_i if and only if D^i entails the fact $\text{yes}^i()$ via Σ^i [51, Theorem 15], [52, Lemma 23], and [81, Lemma 3.2]. We can now provide the details of the reduction.

(The database). The database is taken to be $D = D^1 \cup D^2 \cup \{\text{yes}^1(), \text{yes}^2()\}$.

(The program). We take the program to be $\Sigma = \Sigma^1 \cup \Sigma^2$.

(The query). The query is $q = D^1 \wedge D^2 \wedge \text{yes}^1() \wedge \text{yes}^2()$, where with symbols D^1 and D^2 in the query we mean the conjunction of all database facts from D^1 and D^2 , respectively.

(The candidate MinEx). We take $E = D^1 \cup D^2 \cup \{\text{yes}^2()\} = D \setminus \{\text{yes}^1()\}$.

Observe that the reduction can be computed in polynomial time and that the program has bounded arity. Moreover, the reduction is such that $(D, \Sigma) \models q$, because $D^1 \cup D^2 \cup \{\text{yes}^1(), \text{yes}^2()\} \subseteq D$.

We now prove that \mathcal{T}_1 admits a tiling of the exponential square $2^{n_1} \times 2^{n_1}$ with initial condition s_1 and \mathcal{T}_2 does not admit a tiling of the exponential square $2^{n_2} \times 2^{n_2}$ with initial condition s_2 if and only if E is a MinEx for $(D, \Sigma) \models q$.

(\Rightarrow) Suppose that $((\mathcal{T}_1, s_1, n_1), (\mathcal{T}_2, s_2, n_2))$ is a ‘yes’-instance of EXP-TNT and assume by contradiction that E is not a MinEx for $(D, \Sigma) \models q$. There are two cases: (i) E is an explanation for $(D, \Sigma) \models q$ that is not minimal, or (ii) $(E, \Sigma) \not\models q$ (i.e., E is not an explanation for q at all).

For (i), since the presence of the facts of D^1 and D^2 is required in E to satisfy q , if E is a non-minimal explanation for q , it must be the case that the fact $\text{yes}^2()$ can be removed from E and still have $((E \setminus \{\text{yes}^2()\}), \Sigma) \models q$. However, this implies that $\text{yes}^2()$ can be derived via Σ^2 , which means that \mathcal{T}_2 admits a tiling of the exponential square $2^{n_2} \times 2^{n_2}$ with initial condition s_2 : a contradiction. For (ii), since the facts of D^1 , of D^2 , and $\text{yes}^2()$, are in E , if $(E, \Sigma) \not\models q$, it must be the case that the fact $\text{yes}^1()$ cannot be derived via Σ^1 , which is a contradiction. Thus, E is a MinEx for $(D, \Sigma) \models q$.

(\Leftarrow) Suppose that E is a MinEx for $(D, \Sigma) \models q$. Since E entails q via Σ , the fact $\text{yes}^1()$, which is not present in E , can be derived via Σ^1 , implying that \mathcal{T}_1 admits a tiling of the exponential square $2^{n_1} \times 2^{n_1}$ with initial condition s_1 . Moreover, since E is a minimal explanation and no fact of D^1 or D^2 can be removed from E without losing the property of entailing q , the fact $\text{yes}^2()$ is required to be in E to entail q . This implies that $\text{yes}^2()$ cannot be derived via Σ^2 , and hence that \mathcal{T}_2 does not admit a tiling of the exponential square $2^{n_2} \times 2^{n_2}$ with initial condition s_2 . Thus, $((\mathcal{T}_1, s_1, n_1), (\mathcal{T}_2, s_2, n_2))$ is a ‘yes’-instance of EXP-TNT. \square

The remaining hardness results in Table 2 follow from proving that IS-MINEX is not easier than OMQA in the *fp-combined*, *ba-combined*, and *combined* complexity. Intuitively, to obtain this reduction, we exploit the fact that in the *fp-combined* setting, and above, we can encode the database in the query.

Theorem 4.6. For every class \mathcal{L} of TGDs considered in this paper, IS-MINEX(\mathcal{L}) is at least as hard as OMQA(\mathcal{L}) in the *fp-combined*, (resp., *ba-combined*, *combined*) complexity. Hardness holds even on instances whose queries are BCQs.

⁵ The argument proving this is an extension of the one proposed in the proof of [Theorem 17.1 in 87] showing that SAT-UNSAT is D^P -complete.

Table 3

Complexity results for ALL-MINEX(\mathcal{L}). All non-“in” entries are completeness results. Membership results hold for UCQs, while hardness results hold even on BCQs. In square brackets, we report the number of the theorems providing the membership and the hardness results, respectively. *The theorem indicated holds also for ALL-MINEX by noticing that, in the reduction proposed, the MinEx E is the only MinEx for the query.

\mathcal{L}	Data	fp -combined	ba -combined	Combined
L, LF, AF	in P [5.5 / –]	D ^P [5.3 / 4.4*]	D ^P [5.3 / 4.4*]	PSPACE [5.1 / 4.6*]
S, SF	in P [5.5 / –]	D ^P [5.3 / 4.4*]	D ^P [5.3 / 4.4*]	EXP [5.1 / 4.6*]
A	in P [5.5 / –]	D ^P [5.3 / 4.4*]	D ^{Exp} [5.2 / 4.5*]	D ^{Exp} [5.2 / 4.5*]
G	co-NP [5.1 / 5.6]	D ^P [5.3 / 4.4*]	EXP [5.1 / 4.6*]	2EXP [5.1 / 4.6*]
F, GF	co-NP [5.1 / 5.6]	D ^P [5.3 / 4.4*]	D ^P [5.3 / 4.4*]	EXP [5.1 / 4.6*]
WS, WA	co-NP [5.1 / 5.6]	D ^P [5.3 / 4.4*]	2EXP [5.1 / 4.6*]	2EXP [5.1 / 4.6*]
WG	EXP [5.1 / 4.3*]	EXP [5.1 / 4.6*]	EXP [5.1 / 4.6*]	2EXP [5.1 / 4.6*]

Proof. We prove the statement by showing a logspace reduction from OMQA(\mathcal{L}) to IS-MINEX(\mathcal{L}). Given an OMQA(\mathcal{L}) instance (KB, q) , where $KB = (D, \Sigma)$ is a knowledge base, and q is a BCQ, we build an instance (KB', q', E) of IS-MINEX(\mathcal{L}), where $KB' = (D', \Sigma')$ is a knowledge base, q' is a BCQ, and E is a candidate MinEx.

(The database). We define $D' = D \cup D_q$, where D_q is a set containing the facts of a full grounding of q over fresh constants, i.e., constants not belonging to $dom(D)$.

(The program). We let $\Sigma' = \Sigma$.

(The query). The query is $q' = q \wedge D$, where with the notation D in the query we mean the conjunction of all facts in D .

(The candidate MinEx). We take the set $E = D$.

Observe that this instance of IS-MINEX can be computed in logspace. From $\Sigma \in \mathcal{L}$ follows that $\Sigma' \in \mathcal{L}$, because $\Sigma' = \Sigma$. Since Σ' does not change w.r.t. Σ , when we are in the fp -combined (resp., ba -combined) complexity setting, the classes of instances obtained via this reduction comply with the restrictions on the fixed (resp., bounded-arity) program. Notice moreover that $(D', \Sigma') \models q'$, because D' contains a grounding of q , and that q' is a BCQ.

We now show that $(D, \Sigma) \models q$ if and only if E is a MinEx for $(D', \Sigma') \models q'$.

(\Rightarrow) Suppose that $(D, \Sigma) \models q$. By the definition of Σ' , q' , and E , we have that $(E, \Sigma') \models q'$, hence E is an explanation for q . Moreover, since D' is in q' , no fact from E can be removed without breaking the property of entailing the query q' . Therefore, E is also minimal.

(\Leftarrow) Suppose that $(D, \Sigma) \not\models q$. Since $(D, \Sigma) \not\models q$ and the facts of D_q do not belong to E , we have that $(E, \Sigma') \not\models q'$, by which E is not an explanation (at all) of q' . \square

5. Recognizing the set of all minimal explanations

In this section, we analyze the problem ALL-MINEX of deciding whether given subsets of a database are all and only the MinExes of a given query: for an instance (KB, q, \mathcal{E}) , where $KB = (D, \Sigma)$ is a knowledge base, q is a UCQ (with $KB \models q$), and $\mathcal{E} = \{E_1, \dots, E_n\}$ is a set of subsets of D , decide whether \mathcal{E} is precisely the set of all and only the MinExes for $KB \models q$. A summary of the complexity results for this problem is reported in Table 3.

We highlight for this section an interesting feasibility result. More specifically, for FO-rewritable Datalog[±] fragments, we provide a polynomial time algorithm computing all the MinExes for a given fixed query and a given fixed ontology (see the proof of Theorem 5.4). A second interesting aspect of this section is the general algorithm, i.e., Algorithm 2, to answer ALL-MINEX. This algorithm is more refined than the naive one in which one guesses a missing MinEx, checks that the guessed MinEx is actually missing from the set given in input, and then answers ‘no’. The latter procedure, although completely fine from a correctness perspective, would provide non-tight complexity upper bounds, as it would be too eager of computational resources. The approach employed by Algorithm 2 provides tight complexity upper bounds more easily, as we need to give additional comments only for the D^{Exp} and D^P membership results.

We start hence with the mentioned general procedure, Algorithm 2, which provides a prototypical approach to solve the problem ALL-MINEX (details and comments about the algorithm are provided in the proof of Theorem 5.1). This algorithm allows to obtain (not necessarily tight) upper bounds for ALL-MINEX for all Datalog[±] fragments and complexity settings considered here. The tight membership results obtained via this algorithm are those reported in Table 3 except for the P, D^P, and D^{Exp} ones, for which we need tighter statements that we will provide later.

Observe that there is a significant difference in the way ALL-MINEX needs to be dealt with compared to IS-MINEX. Apart from the minor difference of the necessity to check that multiple sets are MinExes and not just one, like in IS-MINEX, what really makes ALL-MINEX different from IS-MINEX is that to solve the former we need to check that there are *no* MinExes besides those in the input. The intuition at the base of the working of Algorithm 2 is that we first need to check that all sets of facts in \mathcal{E} entail the query, i.e., all of them are actually explanations. Then, we can check, at the *same time*, that all sets in \mathcal{E} are minimal (and hence MinExes) and that there is no MinEx outside \mathcal{E} : we can do this by simply checking that there is *no* explanation E' for the query that either is a proper subset of any of the sets in \mathcal{E} or is not a superset of any set in \mathcal{E} . Indeed, if such E' existed, depending on whether the found

Algorithm 2: A general algorithm for the ALL-MINEX problem.

Input: A knowledge base $KB = (D, \Sigma)$, a UCQ q , and a set $\mathcal{E} = \{E_1, \dots, E_n\}$ of sets $E_i \subseteq D$ of facts.
Output: accept, if \mathcal{E} is the set of all and only the MinExes for $(D, \Sigma) \models q$; reject, otherwise.

/ "verify not" is here followed by a begin/end-enclosed block defining a Boolean function that is straight after invoked to evaluate the preceding "verify not" */*

Procedure AllMinex(KB, q, \mathcal{E}):

```

1  foreach  $E_i \in \mathcal{E}$  do
2    | verify  $(E_i, \Sigma) \models q$ 
3  verify not
4    begin
5    |  $E' \leftarrow$  guess a subset of  $D$ 
6    | verify  $(E'$  is a strict subset of some  $E_i \in \mathcal{E}$ ) or
        |  $(E'$  is not a strict superset of any  $E_j \in \mathcal{E})$ 
7    | verify  $(E', \Sigma) \models q$ 
8    | return accept
9    end
10 return accept

```

E' is contained in a set of \mathcal{E} , or does not contain any of the sets of \mathcal{E} , then E' would witness that a set of facts in \mathcal{E} is a non-minimal explanation, or that E' contains a MinEx that is not in \mathcal{E} , respectively.

Theorem 5.1. *For every class \mathcal{L} of TGDs considered in this paper, if $\text{OMQA}(\mathcal{L})$ is in the complexity class \mathbf{C} in the combined (resp., ba-combined, fp-combined, and data) complexity, then ALL-MINEX(\mathcal{L}) can be decided by a linear number of checks in \mathbf{C} and a check in $\text{co-(NP)}^{\mathbf{C}}$ in the combined (resp., ba-combined, fp-combined, and data) complexity.*

Proof. Let (D, Σ) be a knowledge base with Σ belonging to \mathcal{L} , let q be a UCQ, and let $\mathcal{E} = \{E_1, \dots, E_n\}$ be a set of sets of facts, where each $E_i \subseteq D$. A general approach to check that \mathcal{E} is the set of all and only the MinExes for q is outlined in Algorithm 2 (more comments below). In particular, the set \mathcal{E} is the set of all and only the MinExes for $(D, \Sigma) \models q$ if and only if the following conditions hold: (1) all sets $E_i \in \mathcal{E}$ are MinExes for $(D, \Sigma) \models q$, and (2) there is no MinEx for q missing in \mathcal{E} . Observe that the condition (1) holds if and only if: (1a) all sets $E_i \in \mathcal{E}$ entail q via Σ , and (1b) all sets $E_i \in \mathcal{E}$ are subset-minimal w.r.t. the entailment of q .

Condition (1a) can be checked via $|\mathcal{E}|$ -many (i.e., a linear number of) entailment checks, each of them feasible in the complexity class \mathbf{C} (lines 1 and 2).

Now consider conditions (1b) and (2). Condition (1b) holds if and only if there is *no* set of facts $E' \subseteq D$ entailing the query such that $E' \subsetneq E_i$, for some $E_i \in \mathcal{E}$. Indeed, if E_i is a non-minimal explanation, then it is possible to guess an explanation strictly contained in E_i .⁶ Condition (2) holds if and only if there is *no* set of facts $E' \subseteq D$ entailing the query such that $E' \not\supseteq E_j$, for all $E_j \in \mathcal{E}$. Indeed, if there is a MinEx \tilde{E} missing in \mathcal{E} , then it is possible to guess $E' = \tilde{E}$, and E' is such that it does not contain any of the $E_j \in \mathcal{E}$ (as \tilde{E} is missing in \mathcal{E}). Given the quite similar structure of the tests of these two conditions, their checks can be carried out together (lines 3 to 9).

We can verify the opposite, i.e., the existence of such a set E' (lines 4 to 9), by guessing a subset $E' \subseteq D$ (line 5, feasible in NP), then checking that E' satisfies at least one of the required conditions (line 6, feasible in polynomial time), and then checking, in the complexity class \mathbf{C} , that E' entails q via Σ (line 7). Therefore, overall, this opposite task can be carried out in $\text{NP}^{\mathbf{C}}$. To conclude, checking whether the conditions (1b) and (2) hold is feasible in $\text{co-(NP)}^{\mathbf{C}}$. \square

The membership results in Table 3 obtained from the previous theorem descend also from the following computational complexity facts: $\text{co-(NP)}^{\mathbf{P}} = \text{co-NP}$; for $\mathbf{C} \in \{\text{PSPACE}, \text{EXP}, 2\text{EXP}\}$, $\text{co-(NP)}^{\mathbf{C}} = \mathbf{C}$; the complexity classes P, PSPACE, EXP, and 2EXP, are closed under conjunctions, and hence a linear number of checks in these classes are still in these classes; and a check in P followed by a check in co-NP can be both carried out by a single co-NP machine.

Next we prove that, for the language A, ALL-MINEX can be solved in D^{EXP} ; this result covers the D^{EXP} upper bounds in the combined and ba-combined complexity listed in Table 3. Intuitively, this is due to the fact that OMQA(A) is in NEXP, from which checking whether each set of facts in \mathcal{E} entails the query can be shown in NEXP, and checking whether every set of facts in \mathcal{E} is minimal and that there is no MinEx outside \mathcal{E} is in co-NEXP.

Theorem 5.2. *ALL-MINEX(A) is in D^{EXP} in the combined (resp., ba-combined) complexity.*

Proof. We build on the proof of Theorem 5.1. First, notice that OMQA(A) is in NEXP. Then, the condition (1a) can be checked with a linear number of NEXP tests, that can be combined into a single NEXP test, as NEXP is closed under conjunction. Let us now focus on checking conditions (1b) and (2). We claim that checking that one of them does *not* hold is feasible in NEXP. Indeed, the initial guess of the set E' (line 5) and the additional containment tests for E' (line 6) can be carried out by the same NEXP machine checking also

⁶ Observe that this case covers also the one in which \mathcal{E} contains two sets E_i and E_j such that $E_i \supsetneq E_j$.

the entailment from E' (line 7). Thus, the conditions (1b) and (2) can be checked in co-NEXP. Therefore, in this case, ALL-MINEX is in D^{Exp} . \square

We now show that whenever $\text{OMQA}(\mathcal{L})$ is in NP, the problem ALL-MINEX is in D^P . This theorem covers all the membership results in D^P in Table 3. This is obtained in a similar way to what was done for the membership in D^{Exp} . In this case, it can be shown that checking whether each set of facts in \mathcal{E} entails the query is in NP, and checking whether every set of facts in \mathcal{E} is minimal and that there is no MinEx outside \mathcal{E} is in co-NP.

Theorem 5.3. *For every class \mathcal{L} of TGDs considered in this paper such that $\text{OMQA}(\mathcal{L})$ is in NP in the ba-combined (resp., fp-combined) complexity, ALL-MINEX(\mathcal{L}) is in D^P in the ba-combined (resp., fp-combined) complexity.*

Proof. Consider the proof of Theorem 5.1. The condition (1a) can be checked with a linear number of NP tests, that can be combined into a single NP test, as NP is closed under conjunction. Let us focus now on conditions (1b) and (2). Checking that condition (1b) or (2) do not hold requires to guess a set $E' \subseteq D$ complying with the containment requirements of line 6 and that entails q (line 7). Observe that an NP machine can combine the guess of E' and the guess of the NP certificate witnessing $(E', \Sigma) \models q$ (as $\text{OMQA}(\mathcal{L})$ is in NP). Hence, in this case, the execution of lines 4 to 9 can actually be carried out in NP. Thus, checking the conditions (1b) and (2) is in co-NP. Hence, the overall procedure is in D^P . \square

To conclude with the membership results for ALL-MINEX, we prove that, when \mathcal{L} is a FO-rewritable language, ALL-MINEX is in P in the data complexity. This proves all the P membership results in Table 3. This result is achieved via an intermediate step: we first show that, when we are in the data complexity setting, and \mathcal{L} is FO-rewritable, computing all the MinExes for a query can be done in polynomial time.

Theorem 5.4. *Let (D, Σ) be a knowledge base with $\Sigma \in \mathcal{L}$, where \mathcal{L} is one of the FO-rewritable classes of TGDs considered in this paper, and let q be a UCQ. If Σ and q are fixed, then computing the set of all the MinExes for $(D, \Sigma) \models q$ is feasible in polynomial time.*

Proof. Since query answering over Datalog[±] ontologies is preserved under homomorphisms and \mathcal{L} is an FO-rewritable class of TGDs, besides an FO-rewriting of q and Σ , there also exists a UCQ-rewriting of q and Σ [10,17,99], i.e., there is a UCQ q_Σ such that, for every $D' \subseteq D$, $(D', \Sigma) \models q$ if and only if $D' \models q_\Sigma$. Assume w.l.o.g. that $q_\Sigma = q_\Sigma^1 \vee \dots \vee q_\Sigma^m$, where $q_\Sigma^i = \exists \mathbf{Y}^i \phi_\Sigma^i(\mathbf{Y}^i)$, \mathbf{Y}^i is a tuple of variables, and ϕ_Σ^i is a conjunction of atoms. Recall that the active domain $\text{dom}(D)$ of the database D is the set of all constants appearing in D (as D does not contain nulls). In what follows, for a tuple of constants $\mathbf{a} \in \text{dom}(D)^{|\mathbf{Y}^i|}$, we denote by $\{\phi_\Sigma^i(\mathbf{a})\}$ the set of the facts that are the conjuncts in the formula $\phi_\Sigma^i(\mathbf{a})$.

Let D_Σ^i be the set containing all the sets of facts that can be obtained by grounding $\phi_\Sigma^i(\mathbf{Y}^i)$ with constants from the active domain, i.e., $D_\Sigma^i = \{\{\phi_\Sigma^i(\mathbf{a})\} \mid \mathbf{a} \in \text{dom}(D)^{|\mathbf{Y}^i|}\}$. Notice that D_Σ^i is a set of sets of facts. Observe that, since the query q and the program Σ are fixed, the query q_Σ is fixed, and hence $|\mathbf{Y}^i|$ is a constant. Moreover, the active domain $\text{dom}(D)$ is of polynomial size in the size of the database D . Therefore, it takes polynomial time to compute D_Σ^i . Furthermore, again because the query q_Σ is fixed, there is a constant number m of disjuncts in q_Σ , implying that it takes polynomial time to compute $D_\Sigma = \cup_{i=1}^m D_\Sigma^i$, from which it follows that D_Σ has size polynomial in the size of D .

Let \tilde{D}_Σ be the set of subset-minimal elements of D_Σ , that is, $\tilde{D}_\Sigma = \{F \in D_\Sigma \mid \nexists G \in D_\Sigma \text{ s.t. } G \subsetneq F\}$. This set can be computed in polynomial time in the size of D_Σ , whose size is polynomial in the size of D , and hence \tilde{D}_Σ can be computed in polynomial time in the size of D . Let $\tilde{D}_\Sigma[D] = \{F \in \tilde{D}_\Sigma \mid F \subseteq D\}$ be the selection of the sets in \tilde{D}_Σ that are moreover subsets of D . Clearly, also $\tilde{D}_\Sigma[D]$ can be computed in polynomial time in the size of D .

We claim that $\tilde{D}_\Sigma[D]$ is precisely the set of all and only the MinExes for $(D, \Sigma) \models q$. From the way in which $\tilde{D}_\Sigma[D]$ has been defined, it is not hard to see that each element in $\tilde{D}_\Sigma[D]$ is actually a MinEx for q . It remains to prove that if E is a MinEx for q , then $E \in \tilde{D}_\Sigma[D]$. Let E be a MinEx for $(D, \Sigma) \models q$. We have that $E \models q_\Sigma$, because q_Σ is a UCQ-rewriting of q and Σ . Since E is a MinEx for q , none of its strict subsets satisfies $q_\Sigma = q_\Sigma^1 \vee \dots \vee q_\Sigma^m$. Hence there must be an index i such that E is a subset-minimal set of facts satisfying q_Σ^i , and each $E' \subsetneq E$ does not satisfy any q_Σ^j . Because each set D_Σ^k is defined as containing all the subset-minimal sets of facts satisfying q_Σ^k that can be built from $\text{dom}(D)$, it must be the case that $E \in D_\Sigma^i$ and none of the strict subsets of E belong to any D_Σ^j . Thus, since $D_\Sigma = \cup_{j=1}^m D_\Sigma^j$, $E \in D_\Sigma$ and none of the strict subsets of E belong to D_Σ . Therefore, $E \in \tilde{D}_\Sigma$ as well. From the fact that $E \subseteq D$ we obtain that $E \in \tilde{D}_\Sigma[D]$. \square

From the previous theorem, we obtain that, when \mathcal{L} is a FO-rewritable language, ALL-MINEX is in P in the data complexity. Because, in order to check whether a set of sets of facts \mathcal{E} is actually the set of all and only the MinExes for a query, it suffices to compute the set of all MinExes (in polynomial time) and then compare these two sets.

Theorem 5.5. *For every class \mathcal{L} of TGDs considered in this paper such that \mathcal{L} is FO-rewritable, ALL-MINEX(\mathcal{L}) is in P in the data complexity.*

We now show the lower bounds for the ALL-MINEX problem. First, we show that ALL-MINEX(GF) is co-NP-hard in data complexity. This result covers all the co-NP-hardness results reported in Table 3, as this hardness extends to all languages $\mathcal{L} \supseteq \text{GF}$. The result

is shown via a reduction from the canonical problem UNSAT, rather than from a tailored problem closer to ALL-MINEX as in [94, Theorem 28]. Intuitively, the reduction builds a knowledge base, a query, and a set \mathcal{E} of MinExes, such that the only MinExes that might be outside \mathcal{E} are sets of facts encoding consistent assignments for the Boolean variables of a Boolean formula ϕ ; the construction is done so that if ϕ is satisfiable, then a set of facts encoding a satisfying assignment for ϕ is actually a MinEx outside \mathcal{E} . Again, thanks to the use of a canonical problem in the reduction, we can easily individuate that the source of the complexity of the problem is the task of individuating a new explanation if it exists.

Theorem 5.6. *For every class \mathcal{L} of TGDs considered in this paper such that $\mathcal{L} \supseteq \text{GF}$, $\text{ALL-MINEX}(\mathcal{L})$ is co-NP-hard in the data complexity. Hardness holds even on instances whose queries are BCQs.*

Proof. We provide a reduction to $\text{ALL-MINEX}(\mathcal{L})$ from the co-NP-complete problem UNSAT, which is the problem of deciding whether a 3CNF Boolean formula is unsatisfiable. Let $\phi(\bar{x})$ be a 3CNF formula over Boolean variables $\bar{x} = \{x_1, \dots, x_n\}$ and clauses $C = \{c_1, \dots, c_m\}$. We assume w.l.o.g. that $\phi(\bar{x})$ has at least two clauses not sharing any variable. From $\phi(\bar{x})$, we build the following instance (KB, q, \mathcal{E}) of $\text{ALL-MINEX}(\mathcal{L})$, where $KB = (D, \Sigma)$ is a knowledge base, q is a BCQ, and \mathcal{E} is a set of sets of facts.

(The database). For each Boolean variable $x_i \in \bar{x}$, D contains the facts $\text{false}(x_i)$ and $\text{true}(x_i)$, where x_i is a constant representing the respective variable in $\phi(\bar{x})$, and the facts $\text{false}(x_i)$ and $\text{true}(x_i)$ intuitively state that to the variable x_i is assigned the Boolean value *false* and *true*, respectively.

For each clause c_j of $\phi(\bar{x})$, the database D contains the following facts:

- a fact $\text{succ-cl}(j-1, j)$: this fact states that the j^{th} clause is the successor of the $(j-1)^{\text{th}}$ clause (here, j is a numeric constant, and $j-1$ is an expression denoting the numeric constant associated with the value $j-1$); and
- a fact encoding c_j : e.g., for $c_j = (x_p \vee x_q \vee \neg x_r)$, D contains the fact $c^{lp,p,n}(j, x_p, x_q, x_r)$. The presence of such a fact in D states that the j^{th} clause (see j as the first argument of the fact) is one in which the first, second, and third literal are positive, positive, and negative, respectively (see the superscript of the predicate), and that the first, second, and third variable in the clause are x_p , x_q , and x_r , respectively (again, j is a numeric constant identifying the clause, and x_p , x_q , and x_r are constant associated with the respective Boolean variables of $\phi(\bar{x})$).

More formally, let ℓ be a literal in a clause of $\phi(\bar{x})$, we denote by $\text{var}(\ell)$ the variable of the literal ℓ , and we let $\text{sign}(\ell) = p$ and $\text{sign}(\ell) = n$ if $\ell = x$ and $\ell = \neg x$, for some Boolean variable x , respectively. E.g., for a literal $\ell = \neg x_q$, we have $\text{var}(\ell) = x_q$ and $\text{sign}(\ell) = n$. We denote by $\ell_{j,k}$ the k^{th} literal of the clause c_j . For the clause c_j of $\phi(\bar{x})$, the database contains the fact: $c^{\text{sign}(\ell_{j,1}), \text{sign}(\ell_{j,2}), \text{sign}(\ell_{j,3})}(j, \text{var}(\ell_{j,1}), \text{var}(\ell_{j,2}), \text{var}(\ell_{j,3}))$.

In addition, the database contains the facts $\text{satchain}(0)$ that acts as an initial condition in the satisfiability check, and the fact $\text{maxcl}(m)$ that captures the number of clauses in the formula. To summarize, the database contains facts:

$$D = \{\text{satchain}(0), \text{maxcl}(m)\} \cup \{\text{false}(x_i), \text{true}(x_i) \mid \text{for each variable } x_i \in \bar{x}\} \cup \\ \{\text{succ-cl}(j-1, j), c^{\text{sign}(\ell_{j,1}), \text{sign}(\ell_{j,2}), \text{sign}(\ell_{j,3})}(j, \text{var}(\ell_{j,1}), \text{var}(\ell_{j,2}), \text{var}(\ell_{j,3})) \mid \text{for each clause } c_j \text{ of } \phi\}.$$

(The program). A rule in the program states that if a variable is assigned both *false* and *true* values, then $\text{sat}()$ holds:

$$\text{false}(X) \wedge \text{true}(X) \rightarrow \text{sat}().$$

The program also contains rules capturing different ways to satisfy a clause; $\text{satchl}(J)$ is derived when the J -th clause is *true*. Below, the symbol “•” is a placeholder for either p or n , and “_” is a “don’t care” variable occurring only once:

$$\begin{aligned} c^{lp, \bullet, \bullet}(J, X, _, _) \wedge \text{true}(X) &\rightarrow \text{satchl}(J), \\ c^{ln, \bullet, \bullet}(J, X, _, _) \wedge \text{false}(X) &\rightarrow \text{satchl}(J), \\ c^{lp, \bullet, \bullet}(J, _, Y, _) \wedge \text{true}(Y) &\rightarrow \text{satchl}(J), \\ c^{ln, \bullet, \bullet}(J, _, Y, _) \wedge \text{false}(Y) &\rightarrow \text{satchl}(J), \\ c^{lp, \bullet, p}(J, _, _, Z) \wedge \text{true}(Z) &\rightarrow \text{satchl}(J), \\ c^{lp, \bullet, n}(J, _, _, Z) \wedge \text{false}(Z) &\rightarrow \text{satchl}(J). \end{aligned}$$

The last two rules in the program capture that $\text{sat}()$ is entailed if the *false*-/*true*-facts in a set of facts encode a satisfying assignment for the formula $\phi(\bar{x})$ (more precisely, the rules check that all clauses of the formula are satisfied):

$$\begin{aligned} \text{satchain}(I) \wedge \text{succ-cl}(I, J) \wedge \text{satchl}(J) &\rightarrow \text{satchain}(J), \\ \text{maxcl}(M) \wedge \text{satchain}(M) &\rightarrow \text{sat}(). \end{aligned}$$

(The query). We take the query to be $q = \text{sat}()$.

Table 4

Complexity results for MINEX-IRREL(\mathcal{L}). All non-“in” entries are completeness results. Membership results hold for UCQs, while hardness results hold even on BCQs. In square brackets, we report the number of the theorems providing the membership and the hardness results, respectively.

\mathcal{L}	Data	fp -combined	ba -combined	Combined
L, LF, AF	in P [6.4 / –]	NP [6.3 / 6.5]	NP [6.3 / 6.5]	PSpace [6.1 / 6.5]
S, SF	in P [6.4 / –]	NP [6.3 / 6.5]	NP [6.3 / 6.5]	EXP [6.1 / 6.5]
A	in P [6.4 / –]	NP [6.3 / 6.5]	NEXP [6.2 / 6.5]	NEXP [6.2 / 6.5]
G	NP [6.1 / 6.6]	NP [6.3 / 6.5]	EXP [6.1 / 6.5]	2EXP [6.1 / 6.5]
F, GF	NP [6.1 / 6.6]	NP [6.3 / 6.5]	NP [6.3 / 6.5]	EXP [6.1 / 6.5]
WS, WA	NP [6.1 / 6.6]	NP [6.3 / 6.5]	2EXP [6.1 / 6.5]	2EXP [6.1 / 6.5]
WG	EXP [6.1 / 6.5]	EXP [6.1 / 6.5]	EXP [6.1 / 6.5]	2EXP [6.1 / 6.5]

(The candidate set of all MinExes). As a candidate set of all MinExes, we have: $\mathcal{E} = \{\{false(x_i), true(x_i)\} \mid x_i \in \bar{x}\}$.

This instance of ALL-MINEX can be computed in polynomial time, the program and the query do not depend on $\phi(\bar{x})$, and hence they are fixed, the program is guarded and full, and $(D, \Sigma) \models q$ (via the rule ‘ $false(X) \wedge true(X) \rightarrow sat()$ ’).

We now show that $\phi(\bar{x})$ is unsatisfiable if and only if \mathcal{E} is the set of all and only the MinExes for q .

There are two ways for $sat()$ to be entailed from a set F of facts. A first way is that both facts $false(x_i)$ and $true(x_i)$, for some variable $x_i \in \bar{x}$, are present in F . In this case, $sat()$ is obtained via the rule ‘ $false(X) \wedge true(X) \rightarrow sat()$ ’.

A second way is when F encodes, via the $false$ -/ $true$ -facts that it contains, a satisfying assignment for $\phi(\bar{x})$. In this case, via the rules encoding the possible ways of satisfying the clauses, it is possible to derive from F all facts $satcl(1), \dots, satcl(m)$. Then, via successive applications of the rule ‘ $satchain(I) \wedge succ-cl(I, J) \wedge satcl(J) \rightarrow satchain(J)$ ’, the facts $satchain(1), \dots, satchain(m)$ can be derived. As a last step, via the rule ‘ $maxcl(M) \wedge satchain(M) \rightarrow sat()$ ’ also $sat()$ can be derived, because we have completed the “chain” until the last link. Observe that we can traverse the chain until the last link only if, for all clauses c_j of $\phi(\bar{x})$, it is possible to derive $satcl(j)$.

Because each set in \mathcal{E} contains both facts $false(x_i)$ and $true(x_i)$, for some variable $x_i \in \bar{x}$, and nothing else, each set in \mathcal{E} is a MinEx—observe that, since we have assumed that $\phi(\bar{x})$ has at least two clauses that do not share any variable, neither the singleton $\{false(x_i)\}$, nor the singleton $\{true(x_i)\}$, for any $x_i \in \bar{x}$, can entail the query. Therefore, in order to show that $\phi(\bar{x})$ is unsatisfiable if and only if \mathcal{E} is the set of all and only the MinEx for q it suffices to prove that $\phi(\bar{x})$ is unsatisfiable if and only if there is no MinEx outside \mathcal{E} .

(\Rightarrow) Suppose that $\phi(\bar{x})$ is unsatisfiable, and let us assume by contradiction that there is a MinEx E outside of \mathcal{E} . First, notice that, for all $x_i \in \bar{x}$, $E \not\supseteq \{false(x_i), true(x_i)\}$, otherwise E would not be a MinEx outside \mathcal{E} . Therefore, E encodes a consistent (not necessarily complete) truth assignment for \bar{x} . This means that the rule ‘ $false(X) \wedge true(X) \rightarrow sat()$ ’ cannot be triggered by E . From what we have said, the remaining program rules encode the satisfiability of $\phi(\bar{x})$. Since E entails the query via Σ , it must be the case that E encodes a (not necessarily complete) satisfying assignment for $\phi(\bar{x})$: a contradiction, as we have assumed $\phi(\bar{x})$ to be unsatisfiable.

(\Leftarrow) Suppose that $\phi(\bar{x})$ is satisfiable, and let σ be an assignment to the variables in \bar{x} that satisfies $\phi(\bar{x})$. Then, take $E = \{false(x_i) \mid \sigma(x_i) = false\} \cup \{true(x_i) \mid \sigma(x_i) = true\}$. By construction, $(E, \Sigma) \models q$ and does not contain any set in \mathcal{E} . Therefore, E must contain a MinEx for $(D, \Sigma) \models q$ that does not belong to \mathcal{E} . \square

The remaining hardness results for ALL-MINEX in Table 3 are obtained via minimal variations of the hardness proofs for IS-MINEX. More specifically, in the reductions shown in the proofs of Theorems 4.3 to 4.6, we have constructed the subset E of the database that we have shown to be a MinEx. By inspection of the proofs, it can be seen that this constructed subset E , when it is a MinEx, it is the *only* MinEx in all these reductions. Therefore, we have that all these arguments, when we take $\mathcal{E} = \{E\}$ as the set of *all* the MinExes, apply to ALL-MINEX as well. So, these theorems restated with ALL-MINEX replacing IS-MINEX hold as well.

6. Irrelevant facts for explanations

In this section, we study the complexity of the problem MINEX-IRREL, asking whether there exists a MinEx avoiding a collection of forbidden sets of facts: for an instance (KB, q, F) , where $KB = (D, \Sigma)$ is a knowledge base, q is a UCQ (with $KB \models q$), and $F = \{F_1, \dots, F_n\}$ is a set of subsets of D , decide whether there exists a MinEx E for $(D, \Sigma) \models q$ such that, for all $F_i \in F$, $E \not\supseteq F_i$. A summary of the complexity results for this problem is reported in Table 4.

We first exhibit an algorithm, Algorithm 3, that provides a general procedure for MINEX-IRREL (details and comments about the algorithm are provided in the proof of Theorem 6.1). This general approach provides (not necessarily tight) upper bounds for MINEX-IRREL. The tight ones are all those in Table 4 but the P, the NP in the fp - and ba - combined complexity, and the NEXP ones. For these, we need stronger statements that will be proven later. The main takeaway of this section is the idea behind the algorithm, which is also the reason why we obtain complexity results for MINEX-IRREL that are in lower complexity classes compared to those of IS-MINEX. A procedure simply guessing an explanation E excluding the forbidden sets, and then checking that E entails the query and that none of its subset entails the query is overly complex. Actually, we can avoid to test the minimality of E , as this guessed explanation surely contain within itself a subset-minimal explanation.

Algorithm 3: A general algorithm for the MINEX-IRREL problem.**Input:** A knowledge base $KB = (D, \Sigma)$, a UCQ q , and a set $F = \{F_1, \dots, F_n\}$ of sets $F_i \subseteq D$ of facts.**Output:** accept, if there is a MinEx for $(D, \Sigma) \models q$ not including any $F_i \in F$; reject, otherwise.**Procedure** MinexIrrel(KB, q, F):

```

1   $E \leftarrow$  guess a subset of  $D$ 
2  verify  $(\forall F_i \in F)(E \not\supseteq F_i)$ 
3  verify  $(E, \Sigma) \models q$ 
4  return accept

```

Theorem 6.1. For every class \mathcal{L} of TGDs considered in this paper, if $\text{OMQA}(\mathcal{L})$ is in the complexity class \mathbf{C} in the combined (resp., *ba*-combined, *fp*-combined, and data) complexity, then $\text{MINEX-IRREL}(\mathcal{L})$ can be decided by a check in $\text{NP}^{\mathbf{C}}$ in the combined (resp., *ba*-combined, *fp*-combined, and data) complexity.

Proof. Let (D, Σ) be a knowledge base, let q be a UCQ, and let $F = \{F_1, \dots, F_n\}$, where each $F_i \subseteq D$, be a set of forbidden sets of facts. A general procedure to check the existence of a MinEx for $(D, \Sigma) \models q$ not containing any of the forbidden sets is outlined in Algorithm 3 (more comments below).

Observe that checking whether there exists a MinEx for $(D, \Sigma) \models q$ not containing any set in F just requires to check the existence of *any* (not-necessarily minimal) explanation E for $(D, \Sigma) \models q$ not containing any of the forbidden sets. Indeed, the existence of E guarantees also the existence within E of a MinEx for $(D, \Sigma) \models q$ not containing any of the forbidden sets. Therefore, to solve this problem, we can guess in NP a subset $E \subseteq D$ (line 1), check (in polynomial time) that E does not contain any of the forbidden sets in F (line 2), and then check via an OMQA oracle call, in the complexity class \mathbf{C} , that E entails the query q via the program Σ (line 3). \square

The membership results in Table 4 obtained from the previous theorem descend also from the following computational complexity facts: $\text{NP}^{\text{P}} = \text{NP}$; and for $\mathbf{C} \in \{\text{PSPACE}, \text{EXP}, 2\text{EXP}\}$, $\text{NP}^{\mathbf{C}} = \mathbf{C}$.

We now show that MINEX-IRREL is in NEXP for the Datalog[±] fragment \mathbf{A} in the combined and *ba*-combined complexity. This provides all the NEXP membership results in Table 4. This result holds because a single NEXP machine can guess the explanation and then carry out the checks, which will not be harder than NEXP, as $\text{OMQA}(\mathbf{A})$ is in NEXP in the combined and *ba*-combined complexity.

Theorem 6.2. $\text{MINEX-IRREL}(\mathbf{A})$ is in NEXP in the combined (resp., *ba*-combined) complexity.

Proof. By inspection of Algorithm 3 and of the proof of Theorem 6.1, we can notice that a single NEXP machine can guess the set E at line 1 of the algorithm and subsequently carry out the checks at line 2, which is feasible in polynomial time, and at line 3, which is in NEXP for the specific language and complexity settings considered. Hence, in this case, we do not need an oracle call to perform the entailment test, and the overall procedure can be carried out in NEXP. \square

MINEX-IRREL can be shown in NP for those cases in which $\text{OMQA}(\mathcal{L})$ is in NP. This provides all the NP membership results in Table 4. The intuition behind this result is that, when $\text{OMQA}(\mathcal{L})$ is in NP, the same NP machine can guess the needed explanation and then perform also the entailment test.

Theorem 6.3. For every class \mathcal{L} of TGDs considered in this paper such that $\text{OMQA}(\mathcal{L})$ is in NP in the *ba*-combined (resp., *fp*-combined) complexity, $\text{MINEX-IRREL}(\mathcal{L})$ is in NP in the *ba*-combined (resp., *fp*-combined) complexity.

Proof. If OMQA is in NP, by inspection of Algorithm 3 and of the proof of Theorem 6.1, we can see that the same NP machine guessing the set E at line 1 of the algorithm can also guess the concise certificate to subsequently check in polynomial time the entailment property at line 3. Therefore, in this case, we do not need an oracle call to perform the entailment test. Hence, the overall procedure can be carried out in NP. \square

The only membership results in Table 4 remaining to be proven are the P ones in the data complexity for the classes \mathcal{L} of TGDs for which $\text{OMQA}(\mathcal{L})$ is FO-rewritable. These results are a straightforward consequence of Theorem 5.4. Indeed, when \mathcal{L} is FO-rewritable and we are in the data complexity setting, we can compute in polynomial time all the MinExes for the query, and then we can simply scan this set to look for a MinEx that does not include any of the forbidden sets. Clearly, the entire procedure is feasible in polynomial time.

Theorem 6.4. For every class \mathcal{L} of TGDs considered in this paper such that \mathcal{L} is FO-rewritable, $\text{MINEX-IRREL}(\mathcal{L})$ is in P in the data complexity.

We now show the lower bounds for MINEX-IRREL. The following theorem proves all the hardness results in Table 4, but the NP ones in the data complexity. To show these hardness results, we provide a reduction from $\text{OMQA}(\mathcal{L})$ to $\text{MINEX-IRREL}(\mathcal{L})$, which is a variation of the one in the proof of Theorem 4.6. In this case, we obtain a hardness result valid also in the data-complexity, and

not only for the *fp*-combined complexity and up, because, on the contrary of what happens in the reduction provided in the proof of Theorem 4.6, in the reduction of Theorem 6.5 the query does not vary.

Theorem 6.5. *For every class \mathcal{L} of TGDs considered in this paper, $\text{MINEX-IRREL}(\mathcal{L})$ is at least as hard as $\text{OMQA}(\mathcal{L})$ in the data (resp., *fp*-combined, *ba*-combined, and combined) complexity. Hardness holds even on instances whose queries are BCQs.*

Proof. We prove the statement by showing a logspace reduction from $\text{OMQA}(\mathcal{L})$ to $\text{MINEX-IRREL}(\mathcal{L})$. Given an $\text{OMQA}(\mathcal{L})$ instance (KB, q) , where $KB = (D, \Sigma)$ is a knowledge base, and q is a BCQ, we build an instance (KB', q', \mathcal{F}) of $\text{MINEX-IRREL}(\mathcal{L})$, where $KB' = (D', \Sigma')$ is a knowledge base, q' is a BCQ, and \mathcal{F} is a set of forbidden sets of facts.

(The database). We have $D' = D \cup D_q$, where D_q is a grounding of the query q with fresh constants.

(The program). We take the program $\Sigma' = \Sigma$.

(The query). The query is $q' = q$.

(The forbidden sets of facts). We take $\mathcal{F} = \{\{\alpha\} \mid \alpha \in D_q\}$, where α is every fact in D_q .

Observe that this instance of MINEX-IRREL can be computed in logspace. Notice that, from $\Sigma \in \mathcal{L}$ follows that $\Sigma' \in \mathcal{L}$, because $\Sigma' = \Sigma$. Moreover, as Σ' and q' do not change w.r.t. Σ and q , when we are in the data (resp., *fp*-combined, *ba*-combined, and combined) complexity settings, the classes of instances obtained via this reduction comply with the restrictions on the fixed/bounded-arity program and/or fixed query. Notice moreover that $(D', \Sigma') \models q'$, because D' contains a grounding of q , and that q' is a BCQ.

To conclude, notice that $(D, \Sigma) \models q$ if and only if there is a MinEx for $(D', \Sigma') \models q'$ not containing any set in \mathcal{F} . \square

Our next result proves that $\text{MINEX-IRREL}(\text{GF})$ is NP-hard in the data complexity. This provides all the NP-hardness results in the data complexity in Table 4. Intuitively, the NP-hardness is obtained by encoding a suitable NP-hard graph reachability problem into MINEX-IRREL , and this can be achieved thanks to the fact the GF language allows to express graph reachability via a fixed program and query.

Theorem 6.6. *For every class \mathcal{L} of TGDs considered in this paper such that $\mathcal{L} \supseteq \text{GF}$, $\text{MINEX-IRREL}(\mathcal{L})$ is NP-hard in the data complexity. Hardness holds even on instances whose queries are BCQs.*

Proof. This proof adapts the ideas adopted in the proof of [Theorem 5 in 94]. We reduce to $\text{MINEX-IRREL}(\text{GF})$ the NP-complete problem $\text{PATH-WITH-FORBIDDEN-ARC-PAIRS}$ [56] [58, Problem GT54]: for an instance (G, s, t, C) , where $G = (V, A)$ is a directed acyclic graph, $s, t \in V$ are two vertices of G , and $C \subseteq A \times A$ is a set of pairs of arcs, decide whether there exists in G a path P from s to t such that, for every pair of arcs $(e, e') \in C$, $\{e, e'\} \not\subseteq P$ (i.e., the path P does not pass through both arcs of the pair).

From an instance of $\text{PATH-WITH-FORBIDDEN-ARC-PAIRS}$, we build as follows an instance (KB, q, \mathcal{F}) of $\text{MINEX-IRREL}(\text{GF})$, where $KB = (D, \Sigma)$ is a knowledge base, q is a BCQ, and \mathcal{F} is a set of forbidden sets of facts.

(The database). We encode in the database the graph G , and the starting and destination vertices s and t :

$$D = \{\text{arc}(u, v) \mid (u, v) \in A\} \cup \{\text{reach}(s), \text{dest}(t), \text{reach}(t)\}.$$

Here, u and v in the predicate arc are constants related to the vertices u and v of the graph G , and a fact $\text{arc}(u, v)$ encodes the presence in G of an arc from u to v ; the predicate reach has the meaning that a specific vertex can be reached from s —observe that $\text{reach}(s)$ is in the database, so that, together with the TGD shown below, the predicate reach encodes the reachability in G from s and not from other vertices; the predicate $\text{dest}(t)$ states the destination vertex of the path. The fact $\text{reach}(t)$ in the database ensures that the query is entailed by the database via the program (see the TGDs below). However, we will declare $\text{reach}(t)$ among the forbidden facts, so that we will be able to test whether t can be reached from s , without using the fact $\text{reach}(t)$.

(The program). The program captures what vertices are reachable from a certain vertex (in this case from s , as $\text{reach}(s)$ is in the database; see above), and whether the destination vertex has been reached. In particular, we have two TGDs stating that: first, if a vertex x is reachable, and there is moreover an arc from x to y , then y is reachable as well; and second, if the destination vertex is reached, then we can flag this with the entailment of a nullary fact $\text{dest-reached}()$.

$$\begin{aligned} \Sigma = \{ & \text{reach}(X) \wedge \text{arc}(X, Y) \rightarrow \text{reach}(Y), \\ & \text{dest}(X) \wedge \text{reach}(X) \rightarrow \text{dest-reached}(). \end{aligned}$$

(The query). We take the query to be $q = \text{dest-reached}()$.

(The forbidden sets of facts). The forbidden sets are the pairs of arcs in C plus the reach -fact for the destination vertex:

$$\mathcal{F} = \{\{\text{arc}(u_1, v_1), \text{arc}(u_2, v_2)\} \mid ((u_1, v_1), (u_2, v_2)) \in C\} \cup \{\{\text{reach}(t)\}\}.$$

This instance of MINEX-IRREL can be computed in polynomial time, the program and the query are fixed, the program is guarded and full, and that $(D, \Sigma) \models q$, as $\{\text{dest}(t), \text{reach}(t)\} \subset D$ and $\text{'dest}(X) \wedge \text{reach}(X) \rightarrow \text{dest-reached}()' \in \Sigma$.

Table 5

Complexity results for SMALL-MINEX(\mathcal{L}). All non-“in” entries are completeness results. Membership results hold for UCQs, while hardness results hold even on BCQs. In square brackets, we report the number of the theorems providing the membership and the hardness results, respectively.

\mathcal{L}	Data	fp -combined	ba -combined	Combined
L, LF, AF	in P [7.4 / -]	NP [7.3 / 7.7]	NP [7.3 / 7.7]	PSpace [7.1 / 7.7]
S, SF	in P [7.4 / -]	NP [7.3 / 7.7]	NP [7.3 / 7.7]	EXP [7.1 / 7.7]
A	in P [7.4 / -]	NP [7.3 / 7.7]	NEXP [7.2 / 7.7]	NEXP [7.2 / 7.7]
G	NP [7.1 / 7.5]	NP [7.3 / 7.7]	EXP [7.1 / 7.7]	2EXP [7.1 / 7.7]
F, GF	NP [7.1 / 7.5]	NP [7.3 / 7.7]	NP [7.3 / 7.7]	EXP [7.1 / 7.7]
WS, WA	NP [7.1 / 7.5]	NP [7.3 / 7.7]	2EXP [7.1 / 7.7]	2EXP [7.1 / 7.7]
WG	EXP [7.1 / 7.6]	EXP [7.1 / 7.7]	EXP [7.1 / 7.7]	2EXP [7.1 / 7.7]

Algorithm 4: A general algorithm for the SMALL-MINEX problem.

Input: A knowledge base $KB = (D, \Sigma)$, a UCQ q , and an integer $n \geq 1$.

Output: accept, if there is a MinEx for $(D, \Sigma) \models q$ of size at most n ; reject, otherwise.

Procedure SmallMinex(KB, q, n):

```

1   $E \leftarrow$  guess a subset of  $D$ 
2  verify  $|E| \leq n$ 
3  verify  $(E, \Sigma) \models q$ 
4  return accept
```

We now prove that there is a path in G from s to t not passing through any pair of arcs in C if and only if there is a MinEx for q not containing any set in \mathcal{F} .

(\Rightarrow) Suppose that there is a path P in G from s to t not containing any of the forbidden arc pairs in C . For presentation convenience, assume that P is a sequence of arcs (and not a sequence of vertices). Let $E_P = \{arc(u, v) \mid (u, v) \in P\} \cup \{reach(s)\}$. Notice that E_P entails the query, because P connects s to t , $reach(s)$ is in E , and hence, via the TGDs of Σ , it is possible to derive $reach(t)$ and $dest-reached()$. Furthermore, E_P is a minimal explanation, as otherwise some arc could be removed from P ; but this is not possible, because P is a path in an acyclic graph, implying that P is simple (i.e., P has no cycles), and hence the removal of any arc from P would break the connectivity. Finally, since P does not contain any of the forbidden arc pairs in C , by construction, E_P does not contain any of the sets in \mathcal{F} .

(\Leftarrow) Suppose that there is a MinEx E for q not containing any set in \mathcal{F} . By this, the fact $reach(t)$ does not belong to E . Let $\tilde{E} \subset E$ be the set of *arc*-facts contained in E . Since $(E, \Sigma) \models q$, and since $reach(s) \in E$ and $reach(t) \notin E$, by the fact that a TGD in Σ encodes reachability in a graph, it must be the case that \tilde{E} contains, in principle among others, *arc*-facts associated with arcs in G constituting a path in G from s to t . However, since E is a minimal explanation, no fact from E can be removed without losing the property for E to entail the query. Therefore, the set \tilde{E} , not only contains facts associated with arcs constituting a path in G from s to t , but it does not contain any superfluous *arc*-fact. This implies that the arcs associated with the facts in \tilde{E} constitute a simple path (i.e., a path without cycles).

Let P_E be a sequence of arcs from G built from the *arc*-facts in E as follows: the first arc in P_E is the arc $(s, v) \in A$ for a fact $arc(s, v) \in E$; the other arcs of P_E are appended so that, if (u, v) is the last arc appended to P_E , then the arc $(v, w) \in A$ for some fact $arc(v, w) \in E$ is appended to P_E ; no more arcs are appended to P_E when no additional arc can be appended to P_E according to the above rules.

By the property of \tilde{E} (see above), the sequence P_E , built from E according to the rules above, is actually a path from s to t in the acyclic graph G . To conclude, given that E does not contain any of the forbidden sets in \mathcal{F} , the path P_E does not pass through any of the forbidden arc-pairs in C . \square

7. Small explanations

In this section, we analyze the SMALL-MINEX problem of deciding whether there is a MinEx whose size is not bigger than some given threshold: for an instance (KB, q, n) , where $KB = (D, \Sigma)$ is a knowledge base, q is an UCQ (with $KB \models q$), and n is an integer number represented in binary, decide whether there is a MinEx for $(D, \Sigma) \models q$ of size smaller than or equal to n . We first provide the membership results for UCQs and then proceed with the hardness results for BCQs. The complexity results for SMALL-MINEX are the same of those for MINEX-IRREL and are reported in Table 5.

A general approach to solve SMALL-MINEX is outlined in Algorithm 4. This algorithm allows to obtain (not necessarily tight) upper bounds for all the SMALL-MINEX problems considered.

By comparing Algorithm 4 with Algorithm 3 for MINEX-IRREL, we notice that they are quite similar. The only difference is that, to solve SMALL-MINEX, after guessing a set of facts (line 1), we test that the guessed set is small enough (line 2), instead of checking, as in the case of the MINEX-IRREL problem, that it avoids all the forbidden sets. This is done before checking that the guessed set is an explanation (line 3). As for MINEX-IRREL, also for SMALL-MINEX we do not need to test the minimality of the guessed explanation, because the existence of an explanation E of size at most n guarantees the existence, within E , of a *minimal* explanation of size at most n . The two tasks of checking the size of the guessed explanation and of checking that the guessed explanation avoids the forbidden sets

are both feasible in (deterministic) polynomial time. Hence, the arguments showing the membership results for MINEX-IRREL, with just minimal variations accounting for the test of the size of the guessed explanation, show the membership results of SMALL-MINEX as well. For this reason, we report only the statements of the results, without the detailed proofs.

The following theorem covers all the membership results for SMALL-MINEX in Table 5 apart from the NP, NEXP, and P ones (details of the proof are in the proof of Theorem 6.1).

Theorem 7.1. *For every class \mathcal{L} of TGDs considered in this paper, if $\text{OMQA}(\mathcal{L})$ is in the complexity class \mathbf{C} in the combined (resp., ba-combined, fp-combined, and data) complexity, then $\text{SMALL-MINEX}(\mathcal{L})$ can be decided by a check in $\text{NP}^{\mathbf{C}}$ in the combined (resp., ba-combined, fp-combined, and data) complexity.*

The membership results in Table 5 obtained from the previous theorem descend also from the following computational complexity facts: $\text{NP}^{\text{P}} = \text{NP}$; and for $\mathbf{C} \in \{\text{PSPACE}, \text{EXP}, \text{2EXP}\}$, $\text{NP}^{\mathbf{C}} = \mathbf{C}$.

The next theorem provides all the NEXP membership results in Table 5 (proof details are in the proof of Theorem 6.2).

Theorem 7.2. *$\text{SMALL-MINEX}(\mathbf{A})$ is in NEXP in the combined (resp., ba-combined) complexity.*

The next theorem provides all the NP membership results in Table 5 (proof details are in the proof of Theorem 6.3).

Theorem 7.3. *For every class \mathcal{L} of TGDs considered in this paper such that $\text{OMQA}(\mathcal{L})$ is in NP in the ba-combined (resp., fp-combined) complexity, $\text{SMALL-MINEX}(\mathcal{L})$ is in NP in the ba-combined (resp., fp-combined) complexity.*

The following result provides all the P membership results in Table 5 (details of the proof are provided in the comment before Theorem 6.4).

Theorem 7.4. *For every class \mathcal{L} of TGDs considered in this paper such that \mathcal{L} is FO-rewritable, $\text{SMALL-MINEX}(\mathcal{L})$ is in P in the data complexity.*

This leaves to us only to show the lower bounds. We show first the NP-hardness of SMALL-MINEX(GF) in the data complexity. We notice that the construction in the proof of Theorem 6.6 for the NP-hardness of MINEX-IRREL(GF) in the data complexity does not apply in this case. Thus, we provide a different reduction, which is from VERTEX COVER. This shows all the NP-hardness results in the data complexity in Table 5. Intuitively, we encode the problem of deciding the existence of a small vertex cover in a graph with the problem of deciding the existence of a small MinEx.

Theorem 7.5. *For every class \mathcal{L} of TGDs considered in this paper such that $\mathcal{L} \supseteq \text{GF}$, $\text{SMALL-MINEX}(\mathcal{L})$ is NP-hard in the data complexity. Hardness holds even on instances whose queries are BCQs.*

Proof. Recall that a *vertex cover* of a graph is a set of vertices intersecting every edge of the graph. The NP-complete VERTEX COVER [72] problem asks, for a pair (G, k) , where $G = (V, A)$ is a(n undirected) graph, and k is an integer, to decide whether G has a vertex cover of at most k vertices. We reduce VERTEX COVER to SMALL-MINEX(GF). For an instance (G, k) of VERTEX COVER, where $G = (V, A)$ is a graph, we build as follows an instance (KB, q, n) , where $KB = (D, \Sigma)$ is a knowledge base, q is a BCQ, and n is an integer. In what follows, a_1, \dots, a_m are the m edges in A .

(The database). We encode the graph G in the database D via an incidence graph. More specifically,

$$D_G = \{\text{vertex}(v_i) \mid v_i \in V\} \cup \{\text{inc}(v_i, a_j) \mid \text{the edge } a_j \text{ is incident on the vertex } v_i\},$$

where *vertex* is a predicate stating that v_i is a vertex (v_i is a constant associated with the respective vertex in G), and *inc* is a predicate stating that an edge a_j touches a vertex v_i (a_j is a constant associated with the respective edge in G).

In the database, there are also the *structural* facts needed to test whether a MinEx encodes a vertex cover of G :

$$D_{st} = \{\text{succ}(a_i, a_{i+1}) \mid 1 \leq i \leq |A| - 1\} \cup \{\text{succ}(a_0, a_1), \text{cov-chain}(a_0), \text{max-edge}(a_m)\},$$

where $\text{succ}(a_i, a_{i+1})$ is a predicate stating that a_{i+1} is the edge successor of a_i (“successor” from an index perspective, not that a_{i+1} comes after a_i in the graph topology), $\text{max-edge}(a_m)$ states that a_m is the last edge, $\text{succ}(a_0, a_1)$ states that a_0 (an additional dummy constant) is followed by a_1 , and $\text{cov-chain}(a_0)$ is a predicate allowing us to test whether all edges are covered by a vertex cover (encoded in a MinEx; see the TGDs below). Hence, the database is $D = D_G \cup D_{st}$.

(The program). The program consists of three rules. The first encodes that if a vertex v is in a vertex set, then all edges intersected by v are covered. The second rule is used to build the “sequence” of covered edges. The third rule captures that the fact *all-covered* is entailed only if all the edges are covered (by testing that the “sequence” covers all edges).

Table 6

Complexity results for MINEX-REL(\mathcal{L}). All non-“in” entries are completeness results. Membership results hold for UCQs, while hardness results hold even on BCQs. In square brackets, we report the number of the theorems providing the membership and the hardness results, respectively.

\mathcal{L}	Data	fp -combined	ba -combined	Combined
L, LF, AF	in P [8.2 / –]	Σ_2^P [8.1 / 8.6]	Σ_2^P [8.1 / 8.6]	PSPACE [8.1 / 8.5]
S, SF	in P [8.2 / –]	Σ_2^P [8.1 / 8.6]	Σ_2^P [8.1 / 8.6]	EXP [8.1 / 8.5]
A	in P [8.2 / –]	Σ_2^P [8.1 / 8.6]	P^{NEXP} [8.1 / 8.7]	P^{NEXP} [8.1 / 8.7s]
G	NP [8.1 / 8.3]	Σ_2^P [8.1 / 8.6]	EXP [8.1 / 8.5]	2EXP [8.1 / 8.5]
F, GF	NP [8.1 / 8.3]	Σ_2^P [8.1 / 8.6]	Σ_2^P [8.1 / 8.6]	EXP [8.1 / 8.5]
WS, WA	NP [8.1 / 8.3]	Σ_2^P [8.1 / 8.6]	2EXP [8.1 / 8.5]	2EXP [8.1 / 8.5]
WG	EXP [8.1 / 8.4]	EXP [8.1 / 8.5]	EXP [8.1 / 8.5]	2EXP [8.1 / 8.5]

$$\Sigma = \{ \text{vertex}(V) \wedge \text{inc}(V, A) \rightarrow \text{covered}(A),$$

$$\text{cov-chain}(X) \wedge \text{succ}(X, Y) \wedge \text{covered}(Y) \rightarrow \text{cov-chain}(Y),$$

$$\text{max-edge}(M) \wedge \text{cov-chain}(M) \rightarrow \text{all-covered}() \}.$$

(The query). The query is $q = \text{all-covered}()$.

(The threshold integer). We take the threshold integer to be $n = k + 2 \cdot |A| + 2$. The intuition for this number is that in a set to be a MinEx for $(D, \Sigma) \models q$ there must be exactly $|A|$ -many *inc*-facts, all $|A|$ -many *succ*-facts, *cov-chain*(a_0), and *max-edge*(a_m). Also, we expect that at most k *vertex*-facts to be in a MinEx. This adds up to the number n specified.

Observe that this instance of SMALL-MINEX can be computed in polynomial time, the program and the query are fixed, the program is guarded and full, and that $(D, \Sigma) \models q$, as the whole database encodes the set of all vertices which clearly is a vertex cover of the graph (and hence *all-covered*() can be derived from the whole database via Σ).

We show that G has a vertex cover of size at most k if and only if there is a MinEx for $(D, \Sigma) \models q$ of size at most n .

(\Rightarrow) Suppose that there is a vertex cover C of G of size at most k . Consider the set $E_C = D_{st} \cup \{ \text{vertex}(v_i) \mid v_i \in C \} \cup E'_C$, where E'_C contains, for each edge $a_j \in A$, exactly one fact *inc*(v_i, a_j) such that $v_i \in C$. Notice that E_C is of size at most n (as defined above), and that E_C entails the query by construction. Therefore, E_C is a (not necessarily minimal) explanation of size at most n for the query. However, this is enough to conclude that there exists within E_C a MinEx for the query of size at most n .

(\Leftarrow) Suppose that there is a MinEx E of size at most n . Since E entails the query, by the construction of the program Σ , it must be the case that the set of vertices $C_E = \{ v_i \in V \mid \text{vertex}(v_i) \in E \}$ is a vertex cover of G . By the discussion in the definition of n (see above), since E entails the query, there are $(|A| + 2)$ -many facts in E that are structural facts, and $|A|$ -many facts that are *inc*-facts. Since $|E| \leq n = k + 2 \cdot |A| + 2$, we have that there are at most k *vertex*-facts in E , from which it follows that the vertex cover C_E is of size at most k . \square

The other hardness results in Table 5 follow from Theorem 7.6 and Theorem 7.7 below, whose proofs are slight variations of the proofs of Theorem 4.3 and Theorem 4.6, respectively. More specifically, the reductions proving the two theorems below are from OMQA(\mathcal{L}) to the SMALL-MINEX(\mathcal{L}).

For both theorems, the reductions transform in logspace an OMQA(\mathcal{L}) instance (KB, q) , where $KB = (D, \Sigma)$ is a knowledge base, and q is a BCQ, into an instance (KB', q', n) of SMALL-MINEX(\mathcal{L}), where $KB' = (D', \Sigma')$ is a knowledge base, q' is a BCQ, and n is an integer number. In the reductions of Theorem 7.6 and Theorem 7.7, we build D' , Σ' , and q' as in the reductions of Theorem 4.3 and Theorem 4.6, respectively. Then, in the reductions of Theorem 7.6 and Theorem 7.7, we let the threshold integer n to be $2 \cdot |D| + 2$, and to be $|D|$, respectively.

These proofs end by noticing that if $(D, \Sigma) \models q$, then the only MinEx for q is $E = D' \setminus D_q$, whose size is n . On the other hand, if $(D, \Sigma) \not\models q$, then the only MinEx for q is the entire database D' , whose size is strictly bigger than n .

Theorem 7.6. *For every class \mathcal{L} of TGDs considered in this paper such that $\mathcal{L} \supseteq \text{GF}$, SMALL-MINEX(\mathcal{L}) is at least as hard as OMQA(\mathcal{L}) in the data (resp., fp -combined, ba -combined, and combined) complexity. Hardness holds even on instances whose queries are BCQs.*

Theorem 7.7. *For every class \mathcal{L} of TGDs considered in this paper, SMALL-MINEX(\mathcal{L}) is at least as hard as OMQA(\mathcal{L}) in the fp -combined (resp., ba -combined, and combined) complexity. Hardness holds even on instances whose queries are BCQs.*

8. Relevant facts for explanations

We now carry out a complexity analysis, summarized in Table 6, of the MINEX-REL problem of deciding whether a given fact is relevant for the entailment of a query: for an instance (KB, q, ψ) , where $KB = (D, \Sigma)$ is a knowledge base, q is an UCQ (with $KB \models q$), and $\psi \in D$ is a fact, decide whether there is a MinEx for $(D, \Sigma) \models q$ containing ψ .

A first insight that we get from this section is that, although MINEX-REL may seem pretty similar to MINEX-IRREL, the complexities of the two problems are rather different. Remember that, for the MINEX-IRREL problem, a procedure to answer the problem can be

Algorithm 5: A general algorithm for the MINEX-REL problem.**Input:** A knowledge base $KB = (D, \Sigma)$, a UCQ q , and a fact $\psi \in D$.**Output:** accept, if there is a MinEx for $(D, \Sigma) \models q$ including ψ ; reject, otherwise.**Procedure** MinExRel(KB, q, ψ):

```

1   $E \leftarrow$  guess a subset of  $D$ 
2  verify  $E \ni \psi$ 
3  verify IsMinEx( $KB, q, E$ )
4  return accept

```

Table 7

Complexity results for LARGE-MINEX(\mathcal{L}). All non-“in” entries are completeness results. Membership results hold for UCQs, while hardness results hold even on BCQs. In square brackets, we report the number of the theorems providing the membership and the hardness results, respectively.

\mathcal{L}	Data	fp -combined	ba -combined	Combined
L, LF, AF	in P [9.2 / –]	Σ_2^P [9.1 / 9.6]	Σ_2^P [9.1 / 9.6]	PSpace [9.1 / 9.5]
S, SF	in P [9.2 / –]	Σ_2^P [9.1 / 9.6]	Σ_2^P [9.1 / 9.6]	EXP [9.1 / 9.5]
A	in P [9.2 / –]	Σ_2^P [9.1 / 9.6]	P^{NEXP} [9.1 / 9.7]	P^{NEXP} [9.1 / 9.7s]
G	NP [9.1 / 9.3]	Σ_2^P [9.1 / 9.6]	EXP [9.1 / 9.5]	2EXP [9.1 / 9.5]
F, GF	NP [9.1 / 9.3]	Σ_2^P [9.1 / 9.6]	Σ_2^P [9.1 / 9.6]	EXP [9.1 / 9.5]
WS, WA	NP [9.1 / 9.3]	Σ_2^P [9.1 / 9.6]	2EXP [9.1 / 9.5]	2EXP [9.1 / 9.5]
WG	EXP [9.1 / 9.4]	EXP [9.1 / 9.5]	EXP [9.1 / 9.5]	2EXP [9.1 / 9.5]

as follows: guess an explanation E not including the forbidden sets, and check that E entails the query. This is enough to say that there is a minimal explanation for the query not including the forbidden sets, because within E there exists the sought after MinEx. However, a naive adaptation to MINEX-REL of the procedure for MINEX-IRREL is *not* correct. Indeed, after guessing an explanation E containing the fact ψ , and after checking that E entails the query, we cannot be sure that the MinExes within E actually contain ψ . Therefore, the procedure, to be correct for MINEX-REL, needs an additional step in which the minimality of the guessed explanation E is checked.

An interesting complexity result that we obtain is that, given a plain relational database D , a BCQ query q , and a fact $\psi \in D$, deciding whether there is a minimal subset of D including ψ that entails q is already Σ_2^P -hard (Theorem 8.6), i.e., there is no need to have a program to make the problem difficult to solve. Another complexity result interesting to look at is the P^{NEXP} -hardness of MINEX-REL(A) (Theorem 8.7), which puts this problem among the few natural ones complete for P^{NEXP} , which is the last level of the Strong Exponential Hierarchy.

Let us now delve into the complexity analysis. Again, we start by providing the membership results when UCQs are considered, and then we show the hardness results for MINEX-REL when BCQs are considered. Algorithm 5 outlines a general approach to solve MINEX-REL (details and comments about the algorithm are provided in the proof of Theorem 8.1). This algorithm provides (not necessarily tight) upper bounds for the MINEX-REL problems. In particular, in Table 6, apart from the P membership results, for which we need tighter statements that we will be discussed next, all other membership results are tight and descend from Algorithm 5 and the following statement. Intuitively, to check the existence of a MinEx that contains a distinguished fact ψ , we can guess a candidate minimal explanation containing ψ and then ask to an oracle for IS-MINEX(\mathcal{L}) to check whether E is actually a MinEx.

Theorem 8.1. *For every class \mathcal{L} of TGDs considered in this paper, if IS-MINEX(\mathcal{L}) is in the complexity class \mathbf{D} in the combined (resp., ba -combined, fp -combined, and data) complexity, then MINEX-REL(\mathcal{L}) can be decided by a check in $NP^{\mathbf{D}}$ in the combined (resp., ba -combined, fp -combined, and data) complexity.*

Proof. Let (D, Σ) be a knowledge base, let q be a UCQ, and let $\psi \in D$ be a fact in the database. A general procedure to check the existence of a MinEx for $(D, \Sigma) \models q$ containing ψ is outlined in Algorithm 5 (more comments below). We can check whether there exists such a MinEx for q by guessing (line 1) a subset $E \subseteq D$ of the database (feasible in NP), check that ψ belongs to the guessed set (line 2), and then check whether E is a MinEx for $(D, \Sigma) \models q$ by calling an oracle in the complexity class \mathbf{D} for IS-MINEX(\mathcal{L}) (line 3). \square

The membership results in Table 7 obtained from the previous theorem descend also from the following complexity facts: $NP^P = NP$; for $\mathbf{D} \in \{PSPACE, EXP, 2EXP\}$, $NP^{\mathbf{D}} = \mathbf{D}$; and $P^{NEXP} \subseteq NP^{D^{EXP}} \subseteq NP^{NP^{NEXP}}$, for which, by the collapse of the Strong Exponential Hierarchy [62], it holds $P^{NEXP} = NP^{NP^{NEXP}}$, and hence $P^{NEXP} = NP^{D^{EXP}}$; additionally, $NP^{NP} = \Sigma_2^P$.

The P membership results in the data complexity of Table 6, for the classes \mathcal{L} of FO-rewritable TGDs, are a straightforward consequence of Theorem 5.4. By this theorem, when \mathcal{L} is FO-rewritable, and we are in the data complexity setting, all the MinExes for a query can be computed in polynomial time. Hence, we can simply scan this set to look for a MinEx including the fact ψ . Clearly, the entire procedure is feasible in polynomial time.

Theorem 8.2. *For every class \mathcal{L} of TGDs considered in this paper such that \mathcal{L} is FO-rewritable, MINEX-REL(\mathcal{L}) is in P in the data complexity.*

We now proceed with the study of the complexity lower-bounds of the problem. First, we show that MINEX-REL(GF) is NP-hard in the data complexity. This shows all the NP-hardness results in the data complexity in Table 6. The idea for this reduction is similar to the NP-hardness proof of MINEX-IRREL(GF) in the data complexity, where we encode graph reachability via a fixed program and query.

Theorem 8.3. *For every class \mathcal{L} of TGDs considered in this paper such that $\mathcal{L} \supseteq \text{GF}$, MINEX-REL(\mathcal{L}) is NP-hard in the data complexity. Hardness holds even on instances whose queries are BCQs.*

Proof. This proof adapts ideas from [Theorem 7 in 94]. Consider the problem PATH-VIA-ARC: for an instance (G, s, t, a) , where G is a directed graph, s, t are two vertices of G , and a is an arc of G , decide whether there is in G a simple path from s to t via a . PATH-VIA-ARC has (implicitly) been shown NP-complete by Peñaloza and Sertkaya [94, Theorem 7].

We provide a reduction from PATH-VIA-ARC to MINEX-REL. From an instance (G, s, t, a) of PATH-VIA-ARC, we build as follows an instance (KB, q, ψ) of MINEX-REL, where $KB = (D, \Sigma)$ is a knowledge base, q is a BCQ, and $\psi \in D$ is a fact.

The database D , the program Σ , and the query q , are obtained as in the proof of Theorem 6.6.

(The candidate relevant fact). We take $\psi = \text{arc}(u, v)$.

Remember, from the proof of Theorem 6.6, that this instance of MINEX-REL can be computed in polynomial time, the program and the query are fixed, the program is guarded and full, and that $(D, \Sigma) \models q$.

The argument proving that there is a path in G from s to t via a if and only if there is a MinEx for q including ψ is very similar to the one in the proof of Theorem 6.6. The difference here is that, in the (\Rightarrow) -direction of the proof, we focus on a path P passing via the arc a and we obtain a MinEx E_P including ψ . For the (\Leftarrow) -direction, let E be a MinEx including ψ . Since includes ψ and is minimal, E does not include $\text{reach}(t)$; indeed, if $\text{reach}(t)$ were in E then ψ could be removed from E without breaking the entailment of the query, contradicting that E is a *minimal* explanation. After observing this, the proof follows the (\Leftarrow) -direction of the proof of Theorem 6.6, and we obtain a path P_E via a . \square

The other hardness results in Table 6, but the Σ_2^P and the P^{NEXP} ones, follow from Theorem 8.4 and Theorem 8.5 below, whose proofs are slight variations of the proofs of Theorem 4.3 and Theorem 4.6, respectively. More specifically, the reductions proving the two theorems below are from OMQA(\mathcal{L}) to the *complement* of MINEX-REL(\mathcal{L}).⁷ However, since the results sought after are for *deterministic* complexity classes, these classes are closed under complement, and hence the hardness obtained hold also for MINEX-REL(\mathcal{L}).

For both theorems, the reductions transform in logspace an OMQA(\mathcal{L}) instance (KB, q) , where $KB = (D, \Sigma)$ is a knowledge base, and q is a BCQ, into an instance (KB', q', ψ) of MINEX-REL(\mathcal{L}), where $KB' = (D', \Sigma')$ is a knowledge base, q' is a BCQ, and $\psi \in D$ is a fact. In the reductions of Theorem 8.4 and Theorem 8.5, we build D' , Σ' , and q' as in the reductions of Theorem 4.3 and Theorem 4.6, respectively. Then, in both reductions of Theorem 8.4 and Theorem 8.5, we let ψ to be a fact from D_q (recall that D_q is a grounding of the query q over fresh constants).

To conclude these proofs, it is enough to notice that if $(D, \Sigma) \models q$, then the only MinEx for q' is $E = D' \setminus D_q$, which does not contain ψ , hence there is *no* MinEx for q including ψ . On the other hand, if $(D, \Sigma) \not\models q$, then the only MinEx for q' is the entire database D' , which includes also ψ .

Theorem 8.4. *For every class \mathcal{L} of TGDs considered in this paper such that $\mathcal{L} \supseteq \text{GF}$, the complement of MINEX-REL(\mathcal{L}) is at least as hard as OMQA(\mathcal{L}) in the data (resp., fp-combined, ba-combined, and combined) complexity. Hardness holds even on instances whose queries are BCQs.*

Theorem 8.5. *For every class \mathcal{L} of TGDs considered in this paper, the complement of MINEX-REL(\mathcal{L}) is at least as hard as OMQA(\mathcal{L}) in the fp-combined (resp., ba-combined, and combined) complexity. Hardness holds even on instances whose queries are BCQs.*

Next, we show that MINEX-REL(\mathcal{L}) is Σ_2^P -hard in the fp-combined complexity even when the program is empty. This shows all the Σ_2^P -hardness results in Table 6. We prove this result via a reduction from the canonical Σ_2^P -complete QBF validity problem for a formula $\Phi = \exists \bar{x} \forall \bar{y} \neg \phi(\bar{x}, \bar{y})$, where $\phi(\bar{x}, \bar{y})$ is a quantifier-free Boolean formula in 3CNF, to MINEX-REL. Our reduction is different from what is proposed in the proof of the Σ_2^P -hardness of MINA-RELEVANCE in [94, Theorem 9], in which a tailored Σ_2^P -complete problem, whose shape is rather close to MINA-RELEVANCE, is used for the reduction. By using the QBF validity canonical problem, recognizing from our proof the two sources of complexity for the problem MINEX-REL is easy. More specifically, we encode the formula's structure in the database, the satisfiability condition in the query, the assignment to the existentially quantified Boolean variables of the QBF via the possible MinExes, and the assignments to the universally quantified variables of the QBF are enacted by the homomorphisms mapping the query variables to the facts in the MinExes during the query answering process. We can hence say that the two sources of complexity are finding a *minimal* explanation, and individuating the homomorphism mapping the query atoms over the explanation facts to check the entailment of the query.

⁷ Remember that the complement of a problem is defined on the very same input strings, but 'yes' and 'no' answers are interchanged.

Theorem 8.6. For every class \mathcal{L} of TGDs considered in this paper, $\text{MINEX-REL}(\mathcal{L})$ is Σ_2^P -hard in the *fp*-combined (resp., *ba*-combined) complexity. Hardness holds even on instances whose programs are empty and whose queries are BCQs.⁸

Proof. We show a reduction to $\text{MINEX-REL}(\mathcal{L})$ from the Σ_2^P -complete problem $\text{QBF}_{2,\forall,\neg}^{\text{CNF}}$ [104,112]: given a quantified Boolean formula $\Phi = \exists \bar{x} \forall \bar{y} \neg \phi(\bar{x}, \bar{y})$, where $\phi(\bar{x}, \bar{y})$ is a quantifier-free Boolean formula in 3CNF defined over the Boolean variable sets $\bar{x} = \{x_1, \dots, x_n\}$ and \bar{y} , and clauses $C = \{c_1, \dots, c_m\}$, decide whether Φ is valid.

Let $\Phi = \exists \bar{x} \forall \bar{y} \neg \phi(\bar{x}, \bar{y})$ be an instance of $\text{QBF}_{2,\forall,\neg}^{\text{CNF}}$, we build as follows an instance (KB, q, ψ) of $\text{MINEX-REL}(\mathcal{L})$, where $KB = (D, \Sigma)$ is a knowledge base, q is a BCQ, and $\psi \in D$ is a fact.

(The database). The database D contains facts giving the possibility to encode in the candidate MinExes truth assignments to the existentially quantified variables $x_i \in \bar{x}$. More specifically, for each $x_i \in \bar{x}$, D contains the facts:

$$\text{val}(x_i, f) \quad \text{val}(x_i, t),$$

where x_i is a constant representing the respective Boolean variable in \bar{x} , and f and t are constants associated with the Boolean values *false* and *true*, respectively.

The database D also contains facts used to impose the consistency of truth assignments to the *literals* of $\phi(\bar{x}, \bar{y})$:

$$\begin{array}{llll} \text{simlit}(f, f), & \text{simlit}(f, \star), & \text{opplit}(f, t), & \text{simlit}(\star, \star), \\ \text{simlit}(t, t), & \text{simlit}(t, \star), & \text{opplit}(t, f), & \text{opplit}(\star, \star), \end{array}$$

where f and t are constants with the same meanings as above, and \star is an additional constant used as a “jolly”. Intuitively, the “jolly” constant will allow to bypass the Boolean formula satisfiability check.

Further, the database D contains facts capturing the satisfying assignments to the *literals* for a clause:

$$\text{clsat}(\tau_1, \tau_2, \tau_3), \quad \text{clsat}(\star, \star, \star),$$

where each $\tau_i \in \{f, t\}$ and at least one of τ_i for $i \in \{1, 2, 3\}$ is t . Note that in D there are 7 *clsat*-facts. Intuitively, the facts with the “jolly” constant allow to bypass the satisfiability checks of ϕ .

(The program). We take Σ to be the empty set.

(The query). For presentation purposes, we gradually introduce the query q , by giving intuitions for each piece defined.

A first piece of the query “reads” the assignment for the Boolean variables in \bar{x} encoded in the MinEx onto the query variables T_i , which represent in the query the value of the Boolean variable $x_i \in \bar{x}$, and also imposes that, for each variable $x_i \in \bar{x}$, at least one fact between $\text{val}(x_i, f)$ and $\text{val}(x_i, t)$ is present in the candidate MinExes:

$$\text{assignX} \equiv \bigwedge_{x_i \in \bar{x}} \text{val}(x_i, T_i).$$

In what follows, $\ell_{j,k}$ denotes the k^{th} literal in the j^{th} clause in the formula ϕ , and $v_{j,k}$ is the Boolean variable of the literal $\ell_{j,k}$. A second piece of the query “copies” the truth assignment to each Boolean variable x_i (and stored in the query variable T_i ; see above) onto a suitable query variable $T_{j,k}$, which represents in the query the value of the literal $\ell_{j,k} = x_i$ (this “copy” is captured, in the query evaluation task, by the homomorphism mapping the query variable $T_{j,k}$ to the respective constants in the database via the *simlit*-predicate). Notice here that the query variable $T_{j,k}$ must be one associated with a *positive* literal $\ell_{j,k}$ for which $v_{j,k} = x_i$. We take this part of the query to be

$$\text{copy} \equiv \bigwedge_{x_i \in \bar{x}} \text{simlit}(T_i, T_{j,k}).$$

For *copy* to properly work, each variable $x_i \in \bar{x}$ must appear as a positive literal at least once in the formula ϕ . This can be assumed w.l.o.g., because, if x_i always appears as a negative literal in all the clauses of ϕ , then we can replace all the occurrences of the negative literal $\neg x_i$ with the positive literal x_i without altering the satisfiability property of ϕ .

A third query piece imposes that the various query variables $T_{j,k}$, which are the query variables associated with the formula literals $\ell_{j,k}$, are mapped to consistent constants; in this way, we simulate that the assignments to the formula literals, derived by the mappings of the query atoms to the database facts, are consistent. This part of the query is:

$$\text{consist} \equiv \bigwedge_{\substack{\text{for all pairs of literals} \\ (\ell_{j,k}, \ell_{j',k'}) \text{ in } \phi \text{ s.t.} \\ \ell_{j,k} = \ell_{j',k'}}} \text{simlit}(T_{j,k}, T_{j',k'}) \wedge \bigwedge_{\substack{\text{for all pairs of literals} \\ (\ell_{j,k}, \ell_{j',k'}) \text{ in } \phi \text{ s.t.} \\ \ell_{j,k} = \neg \ell_{j',k'}}} \text{opplit}(T_{j,k}, T_{j',k'}).$$

⁸ Notice that this hardness result holds already over plain relational databases: given a database D , a BCQ q , and a fact $\psi \in D$ (both D and q are part of the input and may vary), deciding whether there is a minimal subset E of D containing ψ and entailing q is Σ_2^P -hard.

The last piece of the query checks the satisfiability of $\phi(\bar{x}, \bar{y})$:

$$\text{satisfied} \equiv \bigwedge_{\substack{\text{for all clauses} \\ c_j \text{ of } \phi}} \text{clsat}(T_{j,1}, T_{j,2}, T_{j,3}).$$

We define the query as $q = D_{st} \wedge \text{assignX} \wedge \text{copy} \wedge \text{consist} \wedge \text{satisfied}$, where with the notation D_{st} in the query we mean the conjunction of all the *structural facts* from $D_{st} = \{\text{simlit}(\mathbf{a}) \mid \text{simlit}(\mathbf{a}) \in D\} \cup \{\text{opplit}(\mathbf{a}) \mid \text{opplit}(\mathbf{a}) \in D\} \cup (\{\text{clsat}(\mathbf{a}) \mid \text{clsat}(\mathbf{a}) \in D\} \setminus \{\text{clsat}(\star, \star, \star)\})$.

(The candidate relevant fact ψ). We choose the distinguished fact $\psi = \text{clsat}(\star, \star, \star)$.

The reduction can be computed in polynomial time. Moreover, the reduction is such that $(D, \Sigma) \models q$, because $\text{clsat}(\star, \star, \star)$, $\text{simlit}(f, \star)$, $\text{simlit}(t, \star)$, $\text{simlit}(\star, \star)$, $\text{opplit}(\star, \star)$, and the *val*-facts are in D ; indeed, assigning the value \star to all the query variables $T_{j,k}$, and any value f or t to all the query variables T_i , satisfies the query. Furthermore, the empty program Σ is vacuously linear (and hence guarded), acyclic, sticky, and full; such a program is clearly also fixed.

We now show that Φ is valid if and only if there is a MinEx for $(D, \Sigma) \models q$ containing ψ . Observe that the formula $\Phi = \exists \bar{x} \forall \bar{y} \neg \phi(\bar{x}, \bar{y})$ is valid if and only if there exists a truth assignment $\sigma_{\bar{x}}$ for \bar{x} such that $\phi(\bar{x}/\sigma_{\bar{x}}, \bar{y})$ is not satisfiable.

(\Rightarrow) Assume that Φ is valid. Hence, there exists a truth assignment $\sigma_{\bar{x}}$ for the variables \bar{x} such that $\phi(\bar{x}/\sigma_{\bar{x}}, \bar{y})$ is not satisfiable. Consider the set $E_{\sigma_{\bar{x}}} = D_{st} \cup \{\text{val}(x_i, f) \mid \sigma_{\bar{x}}(x_i) = \text{false}\} \cup \{\text{val}(x_i, t) \mid \sigma_{\bar{x}}(x_i) = \text{true}\} \cup \{\text{clsat}(\star, \star, \star)\}$. By construction, $E_{\sigma_{\bar{x}}}$ entails the query and contains the fact $\psi = \{\text{clsat}(\star, \star, \star)\}$ (the reason why this set entails the query is the same for which the whole database entails the query; see above). Moreover, we argue that $E_{\sigma_{\bar{x}}}$ is also a minimal explanation. In particular, removing any fact from D_{st} results in the respective part of the query not being entailed, and removing any *val*-fact results in the *assignX* part of the query not being entailed. Moreover, since $\phi(\bar{x}/\sigma_{\bar{x}}, \bar{y})$ is not satisfiable and the *val*-facts in $E_{\sigma_{\bar{x}}}$ encode the truth assignment $\sigma_{\bar{x}}$ for \bar{x} , removing the fact $\text{clsat}(\star, \star, \star)$ from $E_{\sigma_{\bar{x}}}$ breaks the entailment of the query. Therefore, $E_{\sigma_{\bar{x}}}$ is a MinEx for $(D, \Sigma) \models q$ containing $\psi = \text{clsat}(\star, \star, \star)$.

(\Leftarrow) Assume that there is a MinEx E for q containing $\text{clsat}(\star, \star, \star)$. First, we claim that, since E is a minimal explanation, it contains exactly one fact between $\text{val}(x_i, f)$ and $\text{val}(x_i, t)$ for each Boolean variable $x_i \in \bar{x}$. Indeed, since the query variables T_i are existentially quantified in q , the presence of exactly one fact between $\text{val}(x_i, f)$ and $\text{val}(x_i, t)$ in the MinEx is enough for the satisfaction of the *assignX* part of the query. Therefore, E encodes a proper truth assignment σ_E to \bar{x} . Because E is a minimal explanation, the fact $\text{clsat}(\star, \star, \star)$ cannot be removed from E without breaking the entailment of the query. Therefore, $\phi(\bar{x}/\sigma_E, \bar{y})$ is not satisfiable, which implies that Φ is valid. \square

We now study the complexity lower bound for the language A in the *ba*-combined and combined complexity, and show that MINEX-REL(A) is P^{NEXP} -hard in the *ba*-combined complexity; this result provides the two P^{NEXP} -hardness results in Table 6. The result is obtained via a reduction from a P^{NEXP} -complete EXTENDED-TILING problem that is a variation of the exponential tiling problem.

Theorem 8.7. MINEX-REL(A) is P^{NEXP} -hard in the *ba*-combined (resp., combined) complexity. Hardness holds even on instances whose queries are BCQs.

Proof. To prove the statement, we show a reduction from the P^{NEXP} -complete EXTENDED-TILING problem [51, Theorem 6] and [81, Theorem 5.2]: for an instance $(\mathcal{T}_1, \mathcal{T}_2, n, m)$, where \mathcal{T}_1 and \mathcal{T}_2 are two tiling systems, and n and m are two integers represented in unary, decide whether there exists an initial tiling condition s of length m such that \mathcal{T}_1 admits a tiling of the exponential square $2^n \times 2^n$ with initial condition s and \mathcal{T}_2 does not admit a tiling of the exponential square $2^n \times 2^n$ with initial condition s . From an instance $(\mathcal{T}_1, \mathcal{T}_2, n, m)$ of the EXTENDED-TILING problem, we build as follows an instance (KB, q, ψ) of MINEX-REL(A), where $KB = (D, \Sigma)$ is a knowledge base, q is a BCQ, and $\psi \in D$ is a fact.

For the two tiling systems \mathcal{T}_1 and \mathcal{T}_2 , we use the TGD and database encoding presented in [51, Theorems 6 and 15], [52, extended proof of Theorem 6, Lemma 23], and [81, Lemma 3.2, Theorem 5.2]. More specifically, we have two sets of bounded-arity TGDs $\Sigma_{\mathcal{T}_i, n, m}$, for $i = 1, 2$, to encode the structure of the tiling task, that is, describe how a correct tiling looks like when covering the exponential square $2^n \times 2^n$ and has an initial condition of length m ; and two sets $D_{\mathcal{T}_i}$ of facts, for $i = 1, 2$, to encode the adjacency rules of \mathcal{T}_1 and \mathcal{T}_2 , respectively. In this reduction, we do not use the set of facts “ D_w ” (see [51, 81]; this would be the database encoding the initial tiling condition), as we need a reduction simulating a test for all the initial conditions s of length m . The two sets of TGDs and the database facts for \mathcal{T}_1 and \mathcal{T}_2 are made disjoint by using disjoint predicates (we index them with superscript i). That is, we have $\Sigma^1 = \Sigma_{\mathcal{T}_1, n, m}^1$, $D^1 = D_{\mathcal{T}_1}^1$, and $\Sigma^2 = \Sigma_{\mathcal{T}_2, n, m}^2$, $D^2 = D_{\mathcal{T}_2}^2$. The rules Σ^i and the database D^i , for $i = 1, 2$, are such that the tiling system \mathcal{T}_i admits a tiling of the exponential square $2^n \times 2^n$ with an initial tiling condition s , encoded via the *init_j*-facts (see their meaning below), if and only if D^i entails the fact *yesⁱ*() via Σ^i [51, Theorem 15], [52, Lemma 23], and [81, Lemma 3.2]. We can now provide the details of the reduction.

(The database). The database is $D = D^1 \cup D^2 \cup \{\text{diff-tiles}(t', t'') \mid \text{for all pairs of tile-type constants } t' \neq t''\} \cup \{\text{init}_j(t) \mid \text{for all } 0 \leq j < m \text{ and for all tile-type constants } t\} \cup \{\text{yes}^2()\}$, where *diff-tiles*(t', t'') is a predicate stating that the tile constants t' and t'' refer to different tile-types, and *init_j*(t) is a predicate stating that, at the j^{th} position of the initial tiling condition, there is a tile of type t .

(The program). We take the program to be $\Sigma = \Sigma^1 \cup \Sigma^2 \cup \Sigma_{init}$, where Σ_{init} is defined as follows:

$$\begin{aligned}\Sigma_{init} = & \{ \text{diff-tiles}(X, Y) \wedge \text{init}_j(X) \wedge \text{init}_j(Y) \rightarrow \text{yes}^1(), \\ & \text{diff-tiles}(X, Y) \wedge \text{init}_j(X) \wedge \text{init}_j(Y) \rightarrow \text{yes}^2(), \\ & \text{diff-tiles}(X, Y) \wedge \text{init}_j(X) \wedge \text{init}_j(Y) \rightarrow \text{init-all}() \mid \text{for all } 0 \leq j < m \} \cup \\ & \{ \text{init}_0(S_0) \wedge \dots \wedge \text{init}_{m-1}(S_{m-1}) \rightarrow \text{init-all}() \}.\end{aligned}$$

Intuitively, these TGDs state that, if in a candidate MinEx, two different types of tiles are assigned to the same initial location, then $\text{yes}^1()$, $\text{yes}^2()$, and $\text{init-all}()$ are derived; the last rule says that, if in a set of facts, there is at least a fact assigning a tile-type to each location of the initial condition, then we can derive $\text{init-all}()$.

(The query). The query is $q = D^1 \wedge D^2 \wedge \text{init-all}() \wedge \text{yes}^1() \wedge \text{yes}^2()$, where the symbols D^1 and D^2 in the query denote the conjunction of all database facts from D^1 and D^2 , respectively.

(The candidate relevant fact). We take $\psi = \text{yes}^2()$.

Observe that the reduction can be computed in polynomial time. Moreover, the reduction is such that the instances obtained have a bounded-arity program, and such that $(D, \Sigma) \models q$, because $\text{yes}^1()$, $\text{yes}^2()$, and $\text{init-all}()$ are obtained by the presence in D of all the init_j -facts and the TGDs in Σ_{init} .

Before showing that $(\mathcal{T}_1, \mathcal{T}_2, n, m)$ is a ‘yes’-instance of the EXTENDED-TILING problem if and only if there exists a MinEx for q including ψ , let us focus on the possible (minimal) explanations for the query.

A set of facts $E \subseteq D$, to be an explanation, needs to contain all facts in D^1 and D^2 . If besides these facts, E additionally contains, for a single $0 \leq j < m$ and two distinct tiles types t' and t'' , precisely the facts $\text{init}_j(t')$ and $\text{init}_j(t'')$, then we claim that E is a *minimal* explanation. Indeed, due to the presence in E of $\text{init}_j(t')$ and $\text{init}_j(t'')$, from E it is possible to derive $\text{yes}^1()$, $\text{yes}^2()$, and $\text{init-all}()$, besides all the facts in D^1 and D^2 that are assumed to be in E ; therefore, E is an explanation. The set E is moreover a minimal explanation, because removing any fact from E would break the entailment of the query. For this reason, any set of facts that is a *proper* superset of $D^1 \cup D^2 \cup \{\text{init}_j(t'), \text{init}_j(t'')\}$, for a single $0 \leq j < m$ and two distinct tiles types t' and t'' , is a *non-minimal* explanation.

Hence, a necessary condition for a set of facts containing $\psi = \text{yes}^2()$ to be a MinEx is the one of *not* including, for every $0 \leq j < m$ and every pair of different tile-types t' and t'' , the pair of facts $\{\text{init}_j(t'), \text{init}_j(t'')\}$.

Moreover, a MinEx including ψ must also satisfy $\text{init-all}()$ in the query, and hence it has to include at least one fact $\text{init}_j(t)$, for each $0 \leq j < m$. Thus, a necessary condition for a fact set to be a MinEx including ψ is to include exactly one fact $\text{init}_j(t)$, for each $0 \leq j < m$. These sets hence encode a proper initial tiling condition s for the tiling problems.

We now prove that there exists a initial tiling condition s of length m such that \mathcal{T}_1 admits a tiling of the exponential square $2^n \times 2^n$ with the initial condition s , and \mathcal{T}_2 does not admit a tiling of the exponential square $2^n \times 2^n$ with initial condition s , if and only if there exists a MinEx for $(D, \Sigma) \models q$ including ψ .

(\Rightarrow) Assume that there exists an initial condition \tilde{s} of length m such that \mathcal{T}_1 has a solution with \tilde{s} , and \mathcal{T}_2 has no solution with \tilde{s} . Consider the following set of facts $E_{\tilde{s}} = D^1 \cup D^2 \cup \{\text{init}_j(t) \mid \text{the } j^{\text{th}} \text{ tile of } \tilde{s} \text{ is of type } t\} \cup \{\text{yes}^2()\}$.

First, we show that $(E_{\tilde{s}}, \Sigma)$ entails q . Clearly, $E_{\tilde{s}}$ satisfies the parts D^1 , D^2 , and $\text{yes}^2()$ of q , as they are contained in $E_{\tilde{s}}$. Moreover, $E_{\tilde{s}}$ contains exactly one fact $\text{init}_j(t)$ for each $0 \leq j < m$, hence also the part $\text{init-all}()$ of the query is satisfied by $E_{\tilde{s}}$. To conclude, since we are assuming that \mathcal{T}_1 has a solution with \tilde{s} , also the part $\text{yes}^1()$ of q is satisfied via the mediation of the program Σ^1 .

We claim that $E_{\tilde{s}}$ is moreover a minimal explanation. Indeed, none of the facts in $D^1 \cup D^2$ or init_j -facts can be removed from $E_{\tilde{s}}$, otherwise, the query would not be entailed. Regarding $\text{yes}^2()$, since we are assuming that \mathcal{T}_2 has no solution with s , the fact $\text{yes}^2()$ cannot be entailed from $E_{\tilde{s}}$ via the mediation of the program Σ^2 , and hence the presence of $\text{yes}^2()$ in $E_{\tilde{s}}$ is essential for the entailment of the query.

(\Leftarrow) Assume that there exists a MinEx E including $\psi = \text{yes}^2()$. We show that there exists an initial tiling condition s_E of length m such that \mathcal{T}_1 has solution with s_E and \mathcal{T}_2 has no solution with s_E .

Since E is a MinEx containing $\text{yes}^2()$, it must be the case that E encodes a proper initial tiling condition s_E of length m (see the discussion above regarding the presence or not of the pairs of facts $\{\text{init}_j(t'), \text{init}_j(t'')\}$ in a candidate MinEx). Observe that the set E does not contain the fact $\text{yes}^1()$, as the fact $\text{yes}^1()$ does not belong to the database D by construction, but E anyway entails the query, which means that the fact $\text{yes}^1()$ is entailed from E via the mediation of the program Σ^1 , implying that \mathcal{T}_1 has a solution with s_E . Moreover, the minimality of E implies that the fact $\text{yes}^2()$ cannot be removed from E without breaking the entailment of the query. Hence, \mathcal{T}_2 has no solution with s_E . \square

9. Large explanations

In this section, we conclude by carrying out a complexity analysis for the LARGE-MINEX problem of deciding whether there is a MinEx whose size is not smaller than some given threshold: for an instance (KB, q, n) , where $KB = (D, \Sigma)$ is a knowledge base, q is an UCQ (with $KB \models q$), and n is an integer number represented in binary, decide whether there is a MinEx for $(D, \Sigma) \models q$ of size larger than or equal to n . We first provide the membership results for UCQs and proceed with the hardness results for BCQs. The complexity results for LARGE-MINEX are the same of those for MINEX-REL and are reported in Table 7.

Algorithm 6: A general algorithm for the LARGE-MINEX problem.

Input: A knowledge base $KB = (D, \Sigma)$, a UCQ q , and an integer $n \geq 1$.
Output: accept, if there is a MinEx for $(D, \Sigma) \models q$ of size at least n ; reject, otherwise.

Procedure LargeMinex(KB, q, n):

```

1    $E \leftarrow$  guess a subset of  $D$ 
2   verify  $|E| \geq n$ 
3   verify IsMinex( $KB, q, E$ )
4   return accept
```

We outline a general approach solving LARGE-MINEX in Algorithm 6. This algorithm allows us to obtain (not necessarily tight) upper bounds for all the LARGE-MINEX problems considered. Also in this case, if we compare Algorithm 6 with Algorithm 5 for the MINEX-REL problem, we can see that they are very similar. The only difference is that, for LARGE-MINEX, after guessing a set of facts (line 1), we check that the guessed set has size at least n (line 2), instead of checking, as for MINEX-REL, that the guessed set contains the distinguished fact ψ . Then, in both cases, we perform an oracle call to check that the guessed set is a MinEx (line 3)—like for the MINEX-REL case and unlike for the SMALL-MINEX case, we need to check the minimality of the guessed explanation. The two tasks of checking the size of the guessed explanation and checking that the guessed explanation includes the distinguished fact are both feasible in (deterministic) polynomial time. Hence, the arguments showing the membership results for MINEX-REL, with just minimal variations accounting for the test of the size of the guessed explanation, show the membership results for LARGE-MINEX as well. For this reason, we report only the statements of the results, without the detailed proofs. Also in this case, similarly to MINEX-REL, an interesting complexity result that we obtain is that, given a relational database D , a BCQ query q , and an integer $n \geq 1$, deciding whether there is a minimal subset of D of size at least n entailing q is already Σ_2^P -hard (Theorem 9.6), i.e., again there is no need to have a program to make the problem difficult to solve.

The following theorem covers all the membership results for LARGE-MINEX in Table 7, but the P ones (details of the proof are in the proof of Theorem 8.1).

Theorem 9.1. *For every class \mathcal{L} of TGDs considered in this paper, if $\text{IS-MINEX}(\mathcal{L})$ is in the complexity class \mathbf{D} in the combined (resp., ba-combined, fp-combined, and data) complexity, then $\text{LARGE-MINEX}(\mathcal{L})$ can be decided by a check in $\text{NP}^{\mathbf{D}}$ in the combined (resp., ba-combined, fp-combined, and data) complexity.*

The membership results in Table 7 obtained from the previous theorem descend also from the following complexity facts: $\text{NP}^P = \text{NP}$; for $\mathbf{D} \in \{\text{PSPACE}, \text{EXP}, 2\text{EXP}\}$, $\text{NP}^{\mathbf{D}} = \mathbf{D}$; and $\text{P}^{\text{NEXP}} \subseteq \text{NP}^{\text{D}^{\text{EXP}}} \subseteq \text{NP}^{\text{NP}^{\text{NEXP}}}$, for which, by the collapse of the Strong Exponential Hierarchy [62], it holds $\text{P}^{\text{NEXP}} = \text{NP}^{\text{NP}^{\text{NEXP}}}$, and hence $\text{P}^{\text{NEXP}} = \text{NP}^{\text{D}^{\text{EXP}}}$; additionally, $\text{NP}^{\text{NP}} = \Sigma_2^P$.

The following theorem provides all the P membership results in Table 7 (details of the proof are provided in the comment before Theorem 8.2).

Theorem 9.2. *For every class \mathcal{L} of TGDs considered in this paper such that \mathcal{L} is FO-rewritable, $\text{LARGE-MINEX}(\mathcal{L})$ is in P in the data complexity.*

We now proceed with the hardness results. First, we show in the following theorem that LARGE-MINEX is NP-hard for guarded full TGDs in the data complexity. This proves all the NP-hardness results in the data complexity in Table 7. We note that the construction, provided for MINEX-REL is not directly applicable here, and thus we provide a slightly different reduction. In this case, we encode the Hamiltonian path problem into LARGE-MINEX.

Theorem 9.3. *For every class \mathcal{L} of TGDs considered in this paper such that $\mathcal{L} \supseteq \text{GF}$, $\text{LARGE-MINEX}(\mathcal{L})$ is NP-hard in the data complexity. Hardness holds even on instances whose queries are BCQs.*

Proof. Recall that a *Hamiltonian path* in a graph is a *path* traversing all vertices of the graph without passing twice through the same vertex. We reduce to LARGE-MINEX(GF) the NP-complete HAMILTONIAN-PATH problem defined as follows [58, Problem GT39]: for an instance (G, s, t) , where $G = (V, A)$ is a directed graph, and $s, t \in V$ are two vertices, decide whether there exists a Hamiltonian path in G starting from s and terminating in t —notice here that G cannot be assumed acyclic, for otherwise the problem would be polynomial [58, Problem GT39].

From an instance (G, s, t) of HAMILTONIAN-PATH, where $G = (V, A)$ is a directed graph assumed w.l.o.g. to be such that $|V| \geq 2$, we build as follows an instance (KB, q, n) of LARGE-MINEX(GF), where $KB = (D, \Sigma)$ is a knowledge base, q is a BCQ, and n is an integer number. The reduction is an adaptation of the one proposed in the proof of Theorem 6.6. We comment here only the additional elements, database facts and program rules, specific to the current reduction. More details on the meaning of the other elements can be found in the proof of Theorem 6.6.

(*The database*). We encode in the database the graph G , and the starting and destination vertices s and t :

$$\{ \text{arc}(u, v) \mid (u, v) \in A \} \cup \{ \text{reach}(s), \text{dest}(t), \text{reach}(t) \}.$$

The database also contains facts declaring pairs of different vertices and facts over a predicate to have guarded rules:

$$\{diff(v, w) \mid v, w \in V \wedge v \neq w\}, \quad \{guard(u, v, w) \mid u, v, w \in V\}.$$

(The program). The program is defined as

$$\begin{aligned} \Sigma = \{ & reach(X) \wedge arc(X, Y) \rightarrow reach(Y), \\ & dest(X) \wedge reach(X) \rightarrow dest-reached(), \\ & guard(X, Y, Z) \wedge arc(X, Y) \wedge arc(X, Z) \wedge diff(Y, Z) \rightarrow dest-reached() \}, \end{aligned}$$

where the last rule, when triggered, bypasses the test of reachability of t from s when in a set of facts there are facts encoding two different outgoing arcs from a same source vertex.

(The query). We take the query to be $q = dest-reached() \wedge D_{st}$, where with the notation D_{st} in the query we mean the conjunction of all structural facts from the set D_{st} containing all the *diff*-facts and *guard*-facts.

(The threshold integer). We take $n = |D_{st}| + |V| + 1$.

This instance of LARGE-MINEX can be computed in polynomial time, the program and the query are fixed, the program is guarded and full, and $(D, \Sigma) \models q$, as $\{dest(t), reach(t)\} \subset D$ and $'dest(X) \wedge reach(X) \rightarrow dest-reached()'$ $\in \Sigma$.

We show that there is a Hamiltonian path from s to t in G if and only if there is a MinEx for $(D, \Sigma) \models q$ of size at least n .

(\Rightarrow) Suppose that there is a Hamiltonian path P from s to t in G . For presentation convenience, let us assume that P is a sequence of arcs (and not a sequence of vertices). Notice that the path P is of size $|V| - 1$, as it traverses all vertices exactly once. Take the set of facts $E_P = \{arc(u, v) \mid (u, v) \in P\} \cup \{reach(s), dest(t)\} \cup D_{st}$. Observe that $(E_P, \Sigma) \models q$, as t is reachable from s via the path P and the *arc*-facts associated with the arcs in P are in E_P ; hence, E_P is an explanation. Moreover, we claim that E_P is a minimal explanation. Indeed, we cannot remove $reach(s)$ from E_P , for otherwise no *reach*-fact other than $reach(s)$ could be entailed, and neither $reach(t)$. Also, no *arc*-fact can be removed from E_P , because P is a simple path, and this would result in $reach(t)$ not being entailed. Furthermore, no fact of D_{st} can be removed from E , for otherwise the corresponding query part would not be satisfied. Observe that the size of E_P is n .

(\Leftarrow) First, notice that the fact $dest(t)$ must be in every MinEx for q , otherwise the TGD $'dest(X) \wedge reach(X) \rightarrow dest-reached()'$ could not be triggered and the fact $dest-reached() = q$ would not be entailed.

Observe now that there are two kinds of “trivial” MinExes for q : (i) the set $D_{st} \cup \{dest(t), reach(t)\}$, and (ii) the sets $D_{st} \cup \{arc(u, v), arc(u, w)\}$, for a pair of arcs $\{(u, v), (u, w)\} \subseteq A$ with $v \neq w$. The set of kind (i) is a MinEx for the presence of the rule $'dest(X) \wedge reach(X) \rightarrow dest-reached()'$, whereas the sets of kind (ii) are MinExes for the presence the rule $'guard(X, Y, Z) \wedge arc(X, Y) \wedge arc(X, Z) \wedge diff(Y, Z) \rightarrow dest-reached()'$.

Let us consider non-trivial MinExes for q . These MinExes have to be different from trivial ones, and cannot be strict superset of trivial MinExes. Since a non-trivial MinEx is not a strict superset of the trivial MinEx of kind (i), a non-trivial MinEx must include $dest(t)$ (see above) and has to exclude $reach(t)$. By this, a non-trivial MinEx has instead to include $reach(s)$, because it is a necessary fact to start the successive applications of the rule $'reach(X) \wedge arc(X, Y) \rightarrow reach(Y)'$, that eventually entail the fact $reach(t)$ and trigger the TGD $'dest(X) \wedge reach(X) \rightarrow dest-reached()'$. Moreover, since a non-trivial MinEx is neither a strict superset of trivial MinExes of kind (ii), a non-trivial MinEx cannot include two (distinct) *arc*-facts associated with two different outgoing arcs from each given vertex.

Suppose now that there is a MinEx E for $(D, \Sigma) \models q$ of size at least $n = |D_{st}| + |V| + 1$. As we are assuming $|V| \geq 2$, the size of E is at least $|D_{st}| + 3$, and hence E is a non-trivial MinEx. From our observations above, we hence have that:

- $E \supseteq \{reach(s), dest(t)\}$ and $reach(t) \notin E$, and
- E includes at most one *arc*-fact associated with an outgoing arc from each given vertex.

Because E entails the query and $reach(t) \notin E$, among the *arc*-facts in E there are some encoding a path from s to t . However, since E includes at most one *arc*-fact associated with an outgoing arc from each given vertex, all the *arc*-facts in E encode altogether a simple path from s to t .

To conclude, notice that E , to be an explanation, must include all facts from D_{st} , together with the facts $dest(t)$ and $reach(t)$ (see above). Since the size of E is at least $n = |D_{st}| + |V| + 1$, and there is in E at most one *arc*-fact for each given vertex, it must be the case the number of *arc*-facts in E is $|V| - 1$. By this, the *arc*-facts in E must encode a Hamiltonian path in G from s to t . \square

The other hardness results in Table 7, but the Σ_2^P and the P^{NEXP} ones, follow from Theorem 9.4 and Theorem 9.5 below, whose proofs are slight variations of the proofs of Theorem 4.3 and Theorem 4.6, respectively. More specifically, the reductions proving the two theorems below are from $OMQA(\mathcal{L})$ to the complement of $LARGE-MINEX(\mathcal{L})$.⁹ However, since the results sought after are for deterministic complexity classes, these classes are closed under complement, and hence the hardness obtained hold also for $LARGE-MINEX(\mathcal{L})$.

⁹ See Footnote 7 on page 26.

For both theorems, the reductions transform in logspace an OMQA(\mathcal{L}) instance (KB, q) , where $KB = (D, \Sigma)$ is a knowledge base, and q is a BCQ, into an instance (KB', q', n) of LARGE-MINEX(\mathcal{L}), where $KB' = (D', \Sigma')$ is a knowledge base, q' is a BCQ, and n is an integer. In the reductions of Theorem 9.4 and Theorem 9.5, we build D' , Σ' , and q' as in the reductions of Theorem 4.3 and Theorem 4.6, respectively. Then, in the reductions of Theorem 9.4 and Theorem 9.5, we let the threshold integer n to be $2 \cdot |D| + 3$, and to be $|D| + 1$, respectively.

To conclude these proofs, it is enough to notice that if $(D, \Sigma) \models q$, then the only MinEx for q is $E = D' \setminus D_q$, whose size is $n - 1$ (so, not big enough). On the other hand, if $(D, \Sigma) \not\models q$, then the only MinEx for q is the entire database D' , which includes also the facts from D_q , whose size is at least n .

Theorem 9.4. *For every class \mathcal{L} of TGDs considered in this paper such that $\mathcal{L} \supseteq \text{GF}$, the complement of LARGE-MINEX(\mathcal{L}) is at least as hard as OMQA(\mathcal{L}) in the data (resp., fp-combined, ba-combined, and combined) complexity. Hardness holds even on instances whose queries are BCQs.*

Theorem 9.5. *For every class \mathcal{L} of TGDs considered in this paper, the complement of LARGE-MINEX(\mathcal{L}) is at least as hard as OMQA(\mathcal{L}) in the fp-combined, (resp., ba-combined, and combined) complexity. Hardness holds even on instances whose queries are BCQs.*

The Σ_2^P -hardness of LARGE-MINEX(\mathcal{L}) in the fp-combined complexity can be shown by slightly modifying the reduction in the proof of Theorem 8.6. The theorem below provides all Σ_2^P -hardness results in Table 7 for LARGE-MINEX.

Theorem 9.6. *For every class \mathcal{L} of TGDs considered in this paper, LARGE-MINEX(\mathcal{L}) is Σ_2^P -hard in the fp-combined (resp., ba-combined) complexity. Hardness holds even on instances whose programs are empty and whose queries are BCQs.¹⁰*

Proof. We show a reduction to LARGE-MINEX(\mathcal{L}) from the Σ_2^P -complete problem $\text{QBF}_{2, \forall, \neg}^{\text{CNF}}$. From an instance $\Phi = \exists \bar{x} \forall \bar{y} \neg \phi(\bar{x}, \bar{y})$ of $\text{QBF}_{2, \forall, \neg}^{\text{CNF}}$, we obtain an instance (KB, q, n) of LARGE-MINEX as follows, where $KB = (D, \Sigma)$ is a knowledge base, q is a BCQ, and n is an integer number.

The database D , the program Σ , and the query q are obtained as in the proof of Theorem 8.6. (The threshold integer). We take $n = |D_{st}| + |\bar{x}| + 1$.

Recall, from the proof of Theorem 8.6, that this instance of LARGE-MINEX can be computed in polynomial time, the program is fixed, as it is empty, and it is linear (and guarded), acyclic, sticky, and full. Recall also that $(D, \Sigma) \models q$.

(\Rightarrow) This direction easily follows, as in the (\Rightarrow)-direction of the proof of Theorem 8.6, the MinEx $E_{\bar{\sigma}_{\bar{x}}}$ has size n .

(\Leftarrow) Let us assume that there exists a MinEx E for $(D, \Sigma) \models q$ of size at least $n = |D_{st}| + |\bar{x}| + 1$.

We know that every MinEx contains exactly one fact between $\text{val}(x_i, f)$ and $\text{val}(x_i, t)$, for each $x_i \in \bar{x}$ (see the proof of Theorem 8.6). Therefore, E encodes a truth assignment σ_E for the Boolean variables in \bar{x} . As there are $|\bar{x}|$ -many val-facts in E , the other facts in E are at least $(|D_{st}| + 1)$ -many. However, the non-val-facts in D are precisely the facts in $D_{st} \cup \text{clsat}(\star, \star, \star)$, which clearly are $(|D_{st}| + 1)$ -many, therefore all of them are contained in E .

Since E is a minimal explanation, none of the facts in E can be removed without breaking the entailment of the query. Hence, neither $\text{clsat}(\star, \star, \star)$ can be removed from E , implying that this fact is necessary for the entailment of the query. Therefore, by the definition of the query, the truth assignment σ_E for the variables in \bar{x} is such that $\phi(\bar{x}/\sigma_E, \bar{y})$ is not satisfiable. Thus, the formula Φ is valid. \square

The P^{NEXP} -hardness results in Table 7 for the language A in the ba-combined and combined complexity provided by the statement below are obtained by adapting the reduction shown in the proof of Theorem 8.7. This provides all the P^{NEXP} -hardness results in Table 7 for LARGE-MINEX.

Theorem 9.7. *LARGE-MINEX(A) is P^{NEXP} -hard in the ba-combined (resp., combined) complexity. Hardness holds even on instances whose queries are BCQs.*

Proof. We show a reduction to LARGE-MINEX(A) from the P^{NEXP} -hard problem EXTENDED-TILING. From an instance $(\mathcal{T}_1, \mathcal{T}_2, n, m)$ of the EXTENDED-TILING problem, where here we assume w.l.o.g. that $m \geq 2$, we build an instance (KB, q, n') of LARGE-MINEX(A), where $KB = (D, \Sigma)$ is a knowledge base, q is a BCQ, and n' is an integer number.

The database D , the program Σ , and the query q are obtained as in the proof of Theorem 8.7. (The threshold integer). We take $n' = |D^1| + |D^2| + m + 1$.

Recall, from the proof of Theorem 8.7, that this instance of LARGE-MINEX can be computed in polynomial time, the program has bounded arity, and that $(D, \Sigma) \models q$.

¹⁰ Notice that this hardness result holds already over plain relational databases: given a database D , a BCQ q , and an integer number $n \geq 1$ (both D and q are part of the input and may vary), deciding whether there is a minimal subset E of D whose size is at least n and entailing q is Σ_2^P -hard.

(\Rightarrow) This direction easily follows, as in the (\Rightarrow)-direction of the proof of Theorem 8.7, the MinEx $E_{\bar{s}}$ has size n .

(\Leftarrow) Let us assume that there is a MinEx E for $(D, \Sigma) \models q$ of size at least $n = |D^1| + |D^2| + m + 1$, which, by our assumption on m , is at least $|D^1| + |D^2| + 3$. Notice that MinExes of the form $D^1 \cup D^2 \cup \{init_j(t'), init_j(t'')\}$, for some $0 \leq j < m$ and two distinct tile-types t' and t'' , have only size $|D^1| + |D^2| + 2$. Hence, it must be the case that E is a MinEx *not* containing any pair of facts $\{init_j(t'), init_j(t'')\}$, for each $0 \leq j < m$ and two distinct tile-types t' and t'' .

Since E is an explanation and entails the query, it entails also the *init-all*() part of the query. Therefore, E contains exactly one *init_j*-fact, for each $0 \leq j < m$, and hence E encodes an initial tiling condition s_E . From this, we have that the other facts in E are at least $(|D^1| + |D^2| + 1)$ -many. Since the facts in D that are not *init_j*-facts are precisely the facts in $D^1 \cup D^2 \cup \{yes^2()\}$, and clearly they are $(|D^1| + |D^2| + 1)$ -many, it follows that E contains all of them. Therefore, by the fact that E entails the query, it means that $yes^1()$ is entailed by E , which implies that T_1 admits a tiling of the exponential square with initial condition s_E .

On the other hand, since E is a MinEx, none of the facts in E can be removed without breaking the entailment of the q , and this holds for the fact $yes^2()$ as well. Therefore, the presence of $yes^2()$ in E is necessary to ensure the entailment of the query, which implies that T_2 does *not* admit a tiling of the exponential square with initial condition s_E . \square

10. Related work

Abstracting away from subtle differences, the study of explanations for ontological query answering can be classified according to the underlying logical formalism (e.g., description logics or existential rules) and the reasoning task to be explained (e.g., query answering or concept subsumption). In this section, we review the existing literature on explanations for logical formalisms and present the larger context which our work settles in.

Logical abduction. From a broader perspective, we may say that our work deals with *abductive reasoning*, since we seek to find the causes for some observations. Abductive reasoning has been studied in the AI community since the 1970s [39,89,96] with applications in fault diagnosis, belief revision, automated planning, and others. Abductive reasoning has been studied for several formalisms, such as propositional logic [48], logic programs [49], default theories [50], probabilistic temporal logic [85], and DLs [15,94,100]. The study of explanations and diagnosis in logical formalisms dates back to the work of Reiter [98]. He studied the problem of explaining the discrepancy between the observed and the correct system behaviour by employing first-order logic.

An early relevant work to ours is that on *propositional abduction* by Eiter and Gottlob [48], which analyses the complexity of various problems in this setting. In their work, a propositional theory T formalises a particular application domain, a set M of atomic formulas describes the observed manifestations, and a set H of formulas contains possible hypotheses. Then, an *explanation* for M is a subset of hypotheses $S \subseteq H$ such that $T \cup S$ is consistent and logically entails M . This work contains the complexity analysis of a number of decision problems, including the problems of deciding whether an explanation exists, and whether a particular hypothesis is relevant or necessary, which have inspired some of the problems studied in this paper, like MINEX-REL. In their complexity analysis, Eiter and Gottlob [48] looked into the two cases when T is an arbitrary propositional theory, and when T consists only of Horn formulas. Their work also studied a number of different minimality criteria, including, subset-minimality, minimum-cardinality, and weight-minimality (penalisation).

Explaining axiom consequences. Early works dealing with explanations in the context of ontologies primarily focused on DLs as the underlying ontology language and, interestingly, they investigated this topic from an ontology “debugging” perspective. This was because, since devising an ontology is an error-prone task, people were interested in having tools supporting the analysis of ontologies and their consequences. In particular, they were focused in discovering which parts of an ontology bolster *concepts subsumption* or cause *concepts unsatisfiability*: In the former task, given an ontology \mathcal{T} (a TBox, using a DL terminology), it is asked why two given concepts C and D (intuitively, unary predicates, in the terminology of the present paper) are such that $\mathcal{T} \models (\forall x)(C(x) \rightarrow D(x))$ (i.e., D subsumes C)—this was needed to double-check in the ontology the explicitly and implicitly defined taxonomies; whereas in the latter task it is asked why a given concept C is such that $\mathcal{T} \models \neg(\exists x)(C(x))$ (i.e., C is unsatisfiable)—this was regarded as an important sign of a badly-designed ontology.

Different works that dealt with these problems proposed different notions of explanations. The earliest works considered formal proofs as explanations, while later works considered subsets of the TBox as explanations. Providing formal proofs for concept subsumption was first considered by McGuinness and Borgida [84] and Borgida et al. [20], who provided algorithms to obtain such proofs based on different deductive calculi. This approach greatly differs from ours, as we abstract away from a particular proof by considering subsets that support the entailment.

A subsequent line of research focused on *axiom pinpointing* [67,69,100]. In axiom pinpointing, explanations are taken to be minimal subsets of the ontology (TBox). Such explanations are called *justifications* [64,65], or *MinAs*, which stands for “minimal axiom set” [7, 94]. Early works in axiom pinpointing focused on explaining concept unsatisfiability. The first such work, to our knowledge, was by Schlobach and Cornet [100], which was inspired by medical applications. The authors provided an algorithm to pinpoint the axioms in an unfoldable \mathcal{ALC} TBox that cause unsatisfiability of a concept by extending the standard tableau algorithm [101] with labels. Later, this problem was addressed in the Semantic Web setting for repairing unsatisfiable concepts in OWL ontologies [70], which are based on the expressive DL $\mathcal{SHOIN}(\mathcal{D})$. Justifications have also been considered to explain inconsistencies in ontologies [65], where inconsistency means that the ontology does not admit any model and entails everything. Various types of justifications have also been considered depending on different notions of semantic minimality [64]. Later works in axiom pinpointing mainly focused on a more general task of explaining any concept subsumption [7,65,94]. Axiom pinpointing has been extensively studied for the

Semantic Web ontologies, OWL [64,71,105], and for the SNOMED CT medical ontology [7]. Also, systems were developed to compute justifications [71,102].

From the context of axiom pinpointing, the work by Peñaloza and Sertkaya [94] is closely related to ours. They analyse the complexity of decision, counting, and enumeration problems, associated with axiom pinpointing for concepts subsumption, by incorporating and extending many of their previous works [91–93]. The authors give a nearly complete picture of the complexity of axiom pinpointing for lightweight DLs \mathcal{EL} and $DL\text{-}Lite$ family of languages. In particular, they studied the problems of recognising a MinA, recognising the set of all MinAs, deciding whether a particular axiom is relevant for explaining, and deciding whether there is an explanation that avoids all of some forbidden sets. By adapting these problems, we have introduced the problems IS-MINEX, ALL-MINEX, MINEX-REL, and MINEX-IRREL, when studying explanations for query answers. For more expressive description logics, the complexity of the problems associated with axiom pinpointing were considered more recently as well [90].

Some other works on axiom pinpointing focused on analysing the problems associated with computing a pinpointing formula [5,6] that provides an encoding of all MinAs. However, it is known that extracting the MinAs from such representations is already a hard problem. Beside computing justifications efficiently, several works addressed the problem of making them understandable for the user, either by studying their cognitive complexity [66], or by grouping justifications that have a similar structure to help to handle a large number of justifications [9].

Explaining query entailment. Closer to our work are those explicitly dealing with explanations of query entailment (or non-entailment) from knowledge bases. An extensive portion of research on explanations of query entailment has been carried out under the umbrella of *provenance*. Query provenance has widely been studied in the database community [25,40]. The notions of provenance can be divided into the three main categories, namely, why-, how-, and where-provenance. The *why*-provenance, which is the notion closest to our minimal explanation, provides a witness that is a subset of the database for a query [26,41], the *how*-provenance is more general than the *why*-provenance, as it also provides *how* the answer is derived from a witness [53,60], and the *where*-provenance captures where the output data came from in the input database [26,110]. The classical database provenance has recently been extended for ontology-based data access to capture *why*-provenance [23,33]. The main difference between our approach and the work on provenance is that, in our work, we seek *minimal* database subsets supporting the query entailment, while, in the context of provenance, all possible explanations are relevant, irrespective of whether they are minimal or not.

Focusing now on query entailment explanations in knowledge base settings, works in the literature on this topic mainly dealt with query entailment under inconsistency-tolerant semantics or explaining query *non*-entailment. Surprisingly, prior to the initial appearance of our work in its conference version, very little work investigated explanations of query entailment under standard semantics.

To our knowledge, prior to our work, explanations for ontology-mediated query answering under standard semantics have only been studied for the $DL\text{-}Lite$ and $DL\text{-}Lite_A$ families of languages [21,103]. These works provide formal proofs as explanations for query answers, and thus substantially differ from our approach. The only other work on query entailment explanations under existential rules preceding our work is relative to *probabilistic databases* and hence of a very different flavor [34]. Our initial work on this topic was then followed up by some focusing on explaining query answers in DLs [22,37], where the explanations are subsets of the ABox (intuitively, the database, in the terminology of the present paper), another in which the explanations are instead proofs [1], and another focusing on explanations with preferences in the context of existential rules [38].

Explanations for query entailment under inconsistency tolerant semantics have been studied both for description logics and existential rules. In the DL setting, the work that is most similar to ours is that by Bienvenu et al. [18,19], which studies explanations for both positive and negative query answers over inconsistent $DL\text{-}Lite_{core}$ knowledge bases. This work defines explanations, named *causes*, as subsets of the ABox, which makes them very similar to ours, for positive and negative query answers under brave, AR, and IAR semantics. They study the problems of recognizing a minimal explanation, deciding if a particular assertion (intuitively, a fact, in the terminology of the present paper) is relevant, and if a particular assertion is necessary. Despite the fact that their work considers similar problems, the fact that a different semantics is considered makes their work and the analysis in the present paper substantially different. Explanations for query answers under the inconsistency-tolerant semantics ICR was considered by Arioua et al. [3], where an argumentation framework was used as the basis for the explanations. Another argumentation framework was provided for BCQ explanations under the IAR and brave semantics in [2], which was also implemented in the DALEK prototype [4]. Preferred explanations for negative query answers were considered under the inconsistency-tolerant semantics IAR by Du et al. [47]. In the existential rule setting, explanations for query entailment under the AR, IAR, and ICR semantics were studied in [80], and in [82] explanations for query non-entailment under inconsistency-tolerant semantics were studied. Empirical evaluation was provided in [61] for assessing different methods for computing explanations under the ICR semantics.

Explanations for negative query answers have been studied for DLs, under the name of *ABox abduction* [32,45,46,74], and in the context of databases [63]. The closest work to ours is that by Calvanese et al. [31,32]. In this work, as it is common in the ABox abduction research, they start with a set of abducible *predicates*, called *signature*. Then, given a query that is *not* entailed by the knowledge base, the idea is to find a set of assertions such that, when added to the ABox, the entailment holds (avoiding inconsistencies). This formulation is suitable when abducible predicates are known, but relevant constants may not be known. They analysed the computational complexity of the problems associated with negative explanations in $DL\text{-}Lite_{core}$, and in particular they focused on the problems of recognising an explanation, deciding whether an explanation exists, whether an assertion is relevant, and whether an assertion is necessary. They considered arbitrary inclusion-minimal and cardinality-minimal explanations. A similar approach to the problem was also investigated in [43,103]. Koopmann et al. [76] (see also [75]) considered an extended scenario, in which the abduction is not limited to assertions to add to the ABox, but they also consider rules to add to the TBox. Du et al. [46] introduced a kind of minimal explanations, called *representative* explanations, from which all minimal explanations can be retrieved.

Different works focused on the efficient computation of explanations for ABox abduction; see, e.g., [43,45,46,111]. For existential rules, explanations of query non-entailment were investigated in [36].

Explanations for negative query answers have also been considered for database, as minimal sets to be added to support the entailment [63], or, via a different approach, by considering subqueries causing an answer to be missing [14].

11. Conclusion

In this paper, we have defined the notion of minimal explanations for ontological query entailment under the existential rules formalism. We have carried out a thorough complexity analysis of various problems associated with minimal explanations for the most prominent classes of TGDs, and from different complexity measures, with the aim of understanding how the complexity is affected when key parameters of the input are considered to be fixed. We have found that in various circumstances, the complexity of dealing with minimal explanations increases compared to the complexity of query entailment, as it is needed to ensure the minimality of the explanation sets. Among the complexity results obtained, we have proven a few D^{Exp} -completeness ones, which characterize some of the problems studied here among the very few natural ones that are complete for this complexity class.

There are many interesting directions for future work, apart from studying other ontology languages, such as considering richer notions of explanations in which also the rules play a role. In fact, in some inconsistency-tolerant semantics, also ontological rules can be repaired, e.g., in the generalized repair semantics [51,79], and hence richer notions of explanations might be needed, in which also the repair conducted on the ontology has to be taken into account. Other problems can be considered, as counting and enumerating explanations, as it was done, e.g., by Peñaloza and Sertkaya [94]. The case in which preferences can be expressed over explanations, especially when enumerating them, can be enriched by more expressive preference languages [24,77,78], as discussed in [27], where a notion of preferred repairs is introduced in the context of inconsistency-tolerant query answering. We can extend the notion of explanations for the inconsistency-tolerant framework to take into account user preferences, as discussed there, or use richer notions of order to select repairs in the inconsistency-tolerant framework [16,83]. Also investigating more general definitions of explanations encompassing positive and negative query answering, both from the perspective of a consistent and inconsistent setting, is interesting.

CRedit authorship contribution statement

Ismail İlkan Ceylan: Writing – review & editing, Writing – original draft, Validation, Methodology, Investigation, Funding acquisition, Conceptualization. **Thomas Lukasiewicz:** Writing – review & editing, Validation, Supervision, Project administration, Methodology, Funding acquisition, Conceptualization. **Enrico Malizia:** Writing – review & editing, Writing – original draft, Validation, Methodology, Investigation, Funding acquisition, Conceptualization. **Andrius Vaicenavičius:** Writing – review & editing, Writing – original draft, Validation, Methodology, Investigation, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

We thank Marco Calautti and Cristian Molinaro for their useful comments during the preparation of this manuscript.

This work was supported by the Alan Turing Institute under the UK EPSRC grant EP/N510129/1, by the AXA Research Fund, by the UK EPSRC grants EP/R013667/1, EP/L012138/1, and EP/M025268/1, by the EU TAILOR grant 952215, and by the European Union – NextGenerationEU programme, through the Italian Ministry of University and Research (MUR) PRIN 2022-PNRR grant P2022KHTX7 “DISTORT”—CUP: J53D23015000001, under the Italian “National Recovery and Resilience Plan” (PNRR), Mission 4 Component 1.

Data availability

No data was used for the research described in the article.

References

- [1] C. Alrabbaa, S. Borgwardt, P. Koopmann, A. Kovtunova, Explaining ontology-mediated query answers using proofs over universal models, in: Proceedings of the 6th International Joint Conference on Rules and Reasoning (RuleML+RR 2022), 2022, pp. 167–182, https://doi.org/10.1007/978-3-031-21541-4_11.
- [2] A. Arioua, M. Croitoru, Dialectical characterization of consistent query explanation with existential rules, in: Proceedings of the 29th International Florida Artificial Intelligence Research Society Conference (FLAIRS-16), 2016, pp. 621–625, <https://www.aaai.org/ocs/index.php/FLAIRS/FLAIRS16/paper/view/12800>.
- [3] A. Arioua, N. Tamani, M. Croitoru, Query answering explanation in inconsistent Datalog+/- knowledge bases, in: Proceedings of the 26th International Conference on Database and Expert Systems Applications (DEXA-15), 2015, pp. 203–219, https://doi.org/10.1007/978-3-319-22849-5_15.
- [4] A. Arioua, M. Croitoru, P. Buche, DALEK: a tool for dialectical explanations in inconsistent knowledge bases, in: Proceedings of the 6th International Conference on Computational Models of Argument (COMMA-16), 2016, pp. 461–462, <https://doi.org/10.3233/978-1-61499-686-6-461>.
- [5] F. Baader, R. Peñaloza, Automata-based axiom pinpointing, J. Autom. Reason. 45 (2) (2010) 91–129, <https://doi.org/10.1007/s10817-010-9181-2>.
- [6] F. Baader, R. Peñaloza, Axiom pinpointing in general tableaux, J. Log. Comput. 20 (1) (2010) 5–34, <https://doi.org/10.1093/logcom/exn058>.

- [7] F. Baader, B. Suntisrivaraporn, Debugging SNOMED CT using axiom pinpointing in the description logic \mathcal{EL}^+ , in: Proceedings of the 3rd International Conference on Knowledge Representation in Medicine (KR-MED-08), 2008, pp. 1:1–1:7, <http://ceur-ws.org/Vol-410/Paper01.pdf>.
- [8] F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, P.F. Patel-Schneider (Eds.), The Description Logic Handbook: Theory, Implementation, and Applications, 2nd edition, Cambridge University Press, Cambridge, UK, 2007, <https://doi.org/10.1017/CBO9780511711787>.
- [9] S. Bail, B. Parsia, U. Sattler, The logical diversity of explanations in OWL ontologies, in: Proceedings of the 22nd ACM International Conference on Information and Knowledge Management (CIKM-13), 2013, pp. 559–568, <https://doi.org/10.1145/2505515.2505536>.
- [10] P. Barceló, G. Berger, C. Lutz, A. Pieris, First-order rewritability of frontier-guarded ontology-mediated queries, in: Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI-18), 2018, pp. 1707–1713, <https://doi.org/10.24963/IJCAI.2018/236>.
- [11] J.B.L. Bard, S.Y. Rhee, Ontologies in biology: design, applications and future challenges, Nat. Rev. Genet. 5 (2004) 213–222, <https://doi.org/10.1038/nrg1295>.
- [12] C. Beeri, M.Y. Vardi, A proof procedure for data dependencies, J. ACM 31 (4) (1984) 718–741, <https://doi.org/10.1145/1634.1636>.
- [13] V. Bertaud-Gounot, R. Duvauferrier, A. Burgun, Ontology and medical diagnosis, Inform. Health Soc. Care 37 (2) (2012) 51–61, <https://doi.org/10.3109/17538157.2011.590258>.
- [14] N. Bidoit, M. Herschel, K. Tzompanaki, Query-based why-not provenance with NedExplain, in: Proceedings of the 17th International Conference on Extending Database Technology (EDBT-14), 2014, pp. 145–156, <https://doi.org/10.5441/002/edbt.2014.14>.
- [15] M. Bienvenu, Complexity of abduction in the \mathcal{EL} family of lightweight description logics, in: Proceedings of the 11th International Conference on Principles of Knowledge Representation and Reasoning (KR 2008), 2008, pp. 220–230, <https://www.aaai.org/Library/KR/2008/kr08-022.php>.
- [16] M. Bienvenu, C. Bourgaux, F. Goasdoué, Querying inconsistent description logic knowledge bases under preferred repair semantics, in: Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI-14), 2014, pp. 996–1002, <https://doi.org/10.1609/AAAI.V28i1.8855>.
- [17] M. Bienvenu, B. ten Cate, C. Lutz, F. Wolter, Ontology-based data access: a study through disjunctive Datalog, CSP, and MMSNP, ACM Trans. Database Syst. 39 (4) (2014) 33:1–33:44, <https://doi.org/10.1145/2661643>.
- [18] M. Bienvenu, C. Bourgaux, F. Goasdoué, Explaining inconsistency-tolerant query answering over description logic knowledge bases, in: Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI-16), 2016, pp. 900–906, <https://doi.org/10.1609/aaai.v30i1.10092>.
- [19] M. Bienvenu, C. Bourgaux, F. Goasdoué, Computing and explaining query answers over inconsistent DL-Lite knowledge bases, J. Artif. Intell. Res. 64 (2019) 563–644, <https://doi.org/10.1613/jair.1.11395>.
- [20] A. Borgida, E. Franconi, I. Horrocks, Explaining \mathcal{ALC} subsumption, in: Proceedings of the 14th European Conference on Artificial Intelligence (ECAI 2000), 2000, pp. 209–213, <https://frontiersinai.com/ecai/ecai2000/p0209.html>.
- [21] A. Borgida, D. Calvanese, M. Rodríguez-Muro, Explanation in the DL-Lite family of description logics, in: Proceedings of the 2008 OTM Confederated International Conference “On the Move to Meaningful Internet Systems” (OTM-08), 2008, pp. 1440–1457, https://doi.org/10.1007/978-3-540-88873-4_35.
- [22] S. Borgwardt, S. Breuer, A. Kovtunova, Computing ABox justifications for query answers via Datalog rewriting, in: Proceedings of the 36th International Workshop on Description Logics (DL 2023), 2023, pp. 8:1–8:13, <https://ceur-ws.org/Vol-3515/paper-8.pdf>.
- [23] C. Bourgaux, A. Ozaki, R. Peñaloza, L. Predoiu, Provenance for the description logic \mathcal{ELH}' , in: Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI-20), 2020, pp. 1862–1869, <https://doi.org/10.24963/ijcai.2020/258>.
- [24] C. Boutilier, R.I. Brafman, C. Domshlak, H.H. Hoos, D. Poole, CP-nets: a tool for representing and reasoning with conditional ceteris paribus preference statements, J. Artif. Intell. Res. 21 (2004) 135–191, <https://doi.org/10.1613/jair.1234>.
- [25] P. Buneman, The providence of provenance, in: Proceedings of the 29th British National Conference on Databases (BNCOD 2013), 2013, pp. 7–12, https://doi.org/10.1007/978-3-642-39467-6_3.
- [26] P. Buneman, S. Khanna, W.C. Tan, Why and where: a characterization of data provenance, in: Proceedings of the 8th International Conference on Database Theory (ICDT-01), 2001, pp. 316–330, https://doi.org/10.1007/3-540-44503-x_20.
- [27] M. Calautti, S. Greco, C. Molinaro, I. Trubitsyna, Preference-based inconsistency-tolerant query answering under existential rules, Artif. Intell. 312:art (2022) 103772, <https://doi.org/10.1016/j.artint.2022.103772>.
- [28] A. Cali, G. Gottlob, T. Lukasiewicz, A general Datalog-based framework for tractable query answering over ontologies, J. Web Semant. 14 (2012) 57–83, <https://doi.org/10.2139/ssrn.3198957>.
- [29] A. Cali, G. Gottlob, A. Pieris, Towards more expressive ontology languages: the query answering problem, Artif. Intell. 193 (2012) 87–128, <https://doi.org/10.1016/j.artint.2012.08.002>.
- [30] A. Cali, G. Gottlob, M. Kifer, Taming the infinite chase: query answering under expressive relational constraints, J. Autom. Reason. 48 (2013) 115–174, <https://doi.org/10.1613/jair.3873>.
- [31] D. Calvanese, M. Ortiz, M. Simkus, G. Stefanoni, The complexity of explaining negative query answers in DL-Lite, in: Proceedings of the 13th International Conference on Principles of Knowledge Representation and Reasoning (KR 2012), 2012, pp. 583–587, <https://www.aaai.org/ocs/index.php/KR/KR12/paper/view/4537>.
- [32] D. Calvanese, M. Ortiz, M. Simkus, G. Stefanoni, Reasoning about explanations for negative query answers in DL-Lite, J. Artif. Intell. Res. 48 (2013) 635–669, <https://doi.org/10.1613/jair.3870>.
- [33] D. Calvanese, D. Lanti, A. Ozaki, R. Peñaloza, G. Xiao, Enriching ontology-based data access with provenance, in: Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI-19), 2019, pp. 1616–1623, <https://doi.org/10.24963/ijcai.2019/224>.
- [34] I.Ī. Ceylan, S. Borgwardt, T. Lukasiewicz, Most probable explanations for probabilistic database queries, in: Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI-17), 2017, pp. 950–956, <https://doi.org/10.24963/ijcai.2017/132>.
- [35] I.Ī. Ceylan, T. Lukasiewicz, E. Malizia, A. Vaicenaivičius, Explanations for query answers under existential rules, in: Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI-19), 2019, pp. 1639–1646, <https://doi.org/10.24963/ijcai.2019/227>.
- [36] I.Ī. Ceylan, T. Lukasiewicz, E. Malizia, C. Molinaro, A. Vaicenaivičius, Explanations for negative query answers under existential rules, in: Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning (KR 2020), 2020, pp. 223–232, <https://doi.org/10.24963/kr.2020/23>.
- [37] I.Ī. Ceylan, T. Lukasiewicz, E. Malizia, A. Vaicenaivičius, Explanations for ontology-mediated query answering in description logics, in: Proceedings of the 24th European Conference on Artificial Intelligence (ECAI 2020), 2020, pp. 672–679, <https://doi.org/10.3233/FAIA200153>.
- [38] I.Ī. Ceylan, T. Lukasiewicz, E. Malizia, C. Molinaro, A. Vaicenaivičius, Preferred explanations for ontology-mediated queries under existential rules, in: Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI-21), 2021, pp. 6262–6270, <https://doi.org/10.1609/aaai.v35i7.16778>.
- [39] E. Charniak, D. McDermott, Introduction to Artificial Intelligence, Addison-Wesley, Reading, MA, USA, 1985.
- [40] J. Cheney, L. Chiticariu, W.-C. Tan, Provenance in databases: why, how, and where, Found. Trends® Databases 1 (4) (2009) 379–474, <https://doi.org/10.1561/1900000006>.
- [41] Y. Cui, J. Widom, J.L. Wiener, Tracing the lineage of view data in a warehousing environment, ACM Trans. Database Syst. 25 (2) (2000) 179–227, <https://doi.org/10.1145/357775.357777>.
- [42] A. Dawar, G. Gottlob, L. Hella, Capturing relativized complexity classes without order, Math. Log. Q. 44 (1998) 109–122, <https://doi.org/10.1002/malq.19980440108>.
- [43] J. Dolby, A. Fokoue, A. Kalyanpur, A. Kershenbaum, E. Schonberg, K. Srinivas, L. Ma, Scalable semantic retrieval through summarization and refinement, in: Proceedings of the 22nd AAAI Conference on Artificial Intelligence (AAAI-07), 2007, pp. 299–304, <http://www.aaai.org/Library/AAAI/2007/aaai07-046.php>.
- [44] F.K. Došliović, M. Brčić, N. Hlupić, Explainable artificial intelligence: a survey, in: Proceedings of the 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO-18), 2018, pp. 210–215, <https://doi.org/10.23919/mipro.2018.8400040>.

- [45] J. Du, G. Qi, Y. Shen, J.Z. Pan, Towards practical ABox abduction in large OWL DL ontologies, in: Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI-11), 2011, pp. 1160–1165, <https://doi.org/10.1609/aaai.v25i1.8070>.
- [46] J. Du, K. Wang, Y. Shen, A tractable approach to ABox abduction over description logic ontologies, in: Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI-14), 2014, pp. 1034–1040, <https://doi.org/10.1609/aaai.v28i1.8852>.
- [47] J. Du, K. Wang, Y. Shen, Towards tractable and practical ABox abduction over inconsistent description logic ontologies, in: Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI-15), 2015, pp. 1489–1495, <https://doi.org/10.1609/aaai.v29i1.9393>.
- [48] T. Eiter, G. Gottlob, The complexity of logic-based abduction, *J. ACM* 42 (1) (1995) 3–42, <https://doi.org/10.1145/200836.200838>.
- [49] T. Eiter, G. Gottlob, N. Leone, Semantics and complexity of abduction from default theories, *Artif. Intell.* 90 (1–2) (1997) 177–223, [https://doi.org/10.1016/s0004-3702\(96\)00040-9](https://doi.org/10.1016/s0004-3702(96)00040-9).
- [50] T. Eiter, G. Gottlob, N. Leone, Abduction from logic programs: semantics and complexity, *Theor. Comput. Sci.* 189 (1–2) (1997) 129–177, [https://doi.org/10.1016/s0304-3975\(96\)00179-x](https://doi.org/10.1016/s0304-3975(96)00179-x).
- [51] T. Eiter, T. Lukasiewicz, L. Predoiu, Generalized consistent query answering under existential rules, in: Proceedings of the 15th International Conference on Principles of Knowledge Representation and Reasoning (KR 2016), 2016, pp. 359–368, <http://www.aaai.org/ocs/index.php/KR/KR16/paper/view/12888>.
- [52] T. Eiter, T. Lukasiewicz, L. Predoiu, Generalized consistent query answering under existential rules, Manuscript (extended version of [51]), 2016.
- [53] A. Elhalawati, M. Krötzsch, S. Mennicke, An existential rule framework for computing why-provenance on-demand for datalog, in: Proceedings of the 6th International Joint Conference on Rules and Reasoning (RuleML+RR 2022), 2022, pp. 146–163, https://doi.org/10.1007/978-3-031-21541-4_10.
- [54] R. Fagin, P.G. Kolaitis, R.J. Miller, L. Popa, Data exchange: semantics and query answering, *Theor. Comput. Sci.* 336 (1) (2005) 89–124, <https://doi.org/10.1016/j.tcs.2004.10.033>.
- [55] M. Fürer, The computational complexity of the unconstrained limited domino problem (with implications for logical decision problems), in: Logic and Machines: Decision Problems and Complexity (LaM 1983), 1983, pp. 312–319, https://doi.org/10.1007/3-540-13331-3_48.
- [56] H.N. Gabow, S.N. Maheswari, L.J. Osterweil, On two problems in the generation of program test paths, *IEEE Trans. Softw. Eng.* 2 (3) (1976) 227–231, <https://doi.org/10.1109/tse.1976.233819>.
- [57] A. Gainer-Dewar, P. Vera-Licona, The minimal hitting set generation problem: algorithms and computation, *SIAM J. Discrete Math.* 31 (1) (2017) 63–100, <https://doi.org/10.1137/15m1055024>.
- [58] M.R. Garey, D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman & Co., New York, NY, USA, 1979.
- [59] G. Gottlob, E. Malizia, Achieving new upper bounds for the hypergraph duality problem through logic, in: Proceedings of the Joint Meeting of the 23rd EACSL Annual Conference on Computer Science Logic (CSL) and the 29th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), (CSL-LICS '14), 2014, pp. 43:1–43:10, <https://doi.org/10.1145/2603088.2603103>.
- [60] T.J. Green, G. Karvounarakis, V. Tannen, Provenance semirings, in: Proceedings of the 26th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS-07), 2007, pp. 31–40, <https://doi.org/10.1145/1265530.1265535>.
- [61] A. Hecham, A. Arioua, G. Stapleton, M. Croitoru, An empirical evaluation of argumentation in explaining inconsistency-tolerant query answering, in: Proceedings of the 30th International Workshop on Description Logics (DL 2017), 2017, pp. 3:1–3:13, <http://ceur-ws.org/Vol-1879/paper3.pdf>.
- [62] L.A. Hemachandra, The strong exponential hierarchy collapses, *J. Comput. Syst. Sci.* 39 (3) (1989) 299–322, [https://doi.org/10.1016/0022-0000\(89\)90025-1](https://doi.org/10.1016/0022-0000(89)90025-1).
- [63] M. Herschel, M.A. Hernández, Explaining missing answers to SPJUA queries, *Proc. VLDB Endow.* 3 (1) (2010) 185–196, <https://doi.org/10.14778/1920841.1920869>.
- [64] M. Horridge, B. Parsia, U. Sattler, Laconic and precise justifications in OWL, in: Proceedings of the 7th International Semantic Web Conference (ISWC-08), 2008, pp. 323–338, https://doi.org/10.1007/978-3-540-88564-1_21.
- [65] M. Horridge, B. Parsia, U. Sattler, Explaining inconsistencies in OWL ontologies, in: Proceedings of the 3rd International Conference on Scalable Uncertainty Management (SUM-09), 2009, pp. 124–137, https://doi.org/10.1007/978-3-642-04388-8_11.
- [66] M. Horridge, S. Bail, B. Parsia, U. Sattler, The cognitive complexity of OWL justifications, in: Proceedings of the 10th International Semantic Web Conference (ISWC-11), 2011, pp. 241–256, https://doi.org/10.1007/978-3-642-25073-6_16.
- [67] M. Horridge, B. Parsia, U. Sattler, Extracting justifications from bioprotal ontologies, in: Proceedings of the 11th International Semantic Web Conference (ISWC-12), 2012, pp. 287–299, https://doi.org/10.1007/978-3-642-35173-0_19.
- [68] D.S. Johnson, A catalog of complexity classes, in: *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity*, Elsevier and the MIT Press, 1990, pp. 69–161, <https://doi.org/10.1016/B978-0-444-88071-0.50007-2>.
- [69] A. Kalyanpur, B. Parsia, E. Sirin, J.A. Hendler, Debugging unsatisfiable classes in OWL ontologies, *J. Web Semant.* 3 (4) (2005) 268–293, <https://doi.org/10.2139/ssrn.3199261>.
- [70] A. Kalyanpur, B. Parsia, E. Sirin, B. Cuenca-Grau, Repairing unsatisfiable concepts in OWL ontologies, in: Proceedings of the 3rd European Semantic Web Conference (ESWC-06), 2006, pp. 170–184, https://doi.org/10.1007/11762256_15.
- [71] A. Kalyanpur, B. Parsia, M. Horridge, E. Sirin, Finding all justifications of OWL DL entailments, in: Proceedings of the 6th International Semantic Web Conference and the 2nd Asian Semantic Web Conference (ISWC/ASWC-07), 2007, pp. 267–280, https://doi.org/10.1007/978-3-540-76298-0_20.
- [72] R.M. Karp, Reducibility among combinatorial problems, in: *Complexity of Computer Computations*, Springer, 1972, pp. 85–103, https://doi.org/10.1007/978-1-4684-2001-2_9.
- [73] S. Klamt, U. Haus, F.J. Theis, Hypergraphs and cellular networks, *PLoS Comput. Biol.* 5 (5) (2009) e1000385, <https://doi.org/10.1371/journal.pcbi.1000385>.
- [74] S. Klarman, U. Endriss, S. Schlobach, ABox abduction in the description logic \mathcal{ALC} , *J. Autom. Reason.* 46 (1) (2011) 43–80, <https://doi.org/10.1007/s10817-010-9168-z>.
- [75] P. Koopmann, LETHE-abduction: a tool for signature-based abduction for \mathcal{ALC} knowledge bases, <https://lat.inf.tu-dresden.de/~koopmann/LETHE-Abduction/>, 2023.
- [76] P. Koopmann, W. Del-Pinto, S. Tournet, R.A. Schmidt, Signature-based abduction for expressive description logics, in: Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning (KR 2020), 2020, pp. 592–602, <https://doi.org/10.24963/KR.2020/59>.
- [77] T. Lukasiewicz, E. Malizia, On the complexity of preference aggregation over (m) CP-nets: Pareto and Majority voting, *Artif. Intell.* 272 (2019) 101–142, <https://doi.org/10.1016/j.artint.2018.12.010>.
- [78] T. Lukasiewicz, E. Malizia, On the complexity of preference aggregation over (m) CP-nets: max and rank voting, *Artif. Intell.* 303:art (2022) 103636, <https://doi.org/10.1016/j.artint.2021.103636>.
- [79] T. Lukasiewicz, E. Malizia, C. Molinaro, Complexity of approximate query answering under inconsistency in Datalog+/, in: Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI-18), 2018, pp. 1921–1927, <https://doi.org/10.24963/ijcai.2018/265>.
- [80] T. Lukasiewicz, E. Malizia, C. Molinaro, Explanations for inconsistency-tolerant query answering under existential rules, in: Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI-20), 2020, pp. 2909–2916, <https://doi.org/10.1609/aaai.v34i03.5682>.
- [81] T. Lukasiewicz, E. Malizia, M.V. Martinez, C. Molinaro, A. Pieris, G.I. Simari, Inconsistency-tolerant query answering for existential rules, *Artif. Intell.* 307:art (2022) 103685, <https://doi.org/10.1016/j.artint.2022.103685>.
- [82] T. Lukasiewicz, E. Malizia, C. Molinaro, Explanations for negative query answers under inconsistency-tolerant semantics, in: Proceedings of the 31st International Joint Conference on Artificial Intelligence (IJCAI-22), 2022, pp. 2705–2711, <https://doi.org/10.24963/ijcai.2022/375>.
- [83] T. Lukasiewicz, E. Malizia, C. Molinaro, Complexity of inconsistency-tolerant query answering in Datalog+/- under preferred repairs, in: Proceedings of the 20th International Conference on Principles of Knowledge Representation and Reasoning (KR 2023), 2023, pp. 472–481, <https://doi.org/10.24963/KR.2023/46>.

- [84] D.L. McGuinness, A. Borgida, Explaining subsumption in description logics, in: Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95), 1995, pp. 816–821, <https://www.ijcai.org/Proceedings/95-1/Papers/105.pdf>.
- [85] C. Molinaro, A. Sliva, V.S. Subrahmanian, Super-solutions: succinctly representing solutions in abductive annotated probabilistic temporal logic, *ACM Trans. Comput. Log.* 15 (3) (2014) 18:1–18:35, <https://doi.org/10.1145/2627354>.
- [86] A. Onet, The chase procedure and its applications in data exchange, in: P.G. Kolaitis, M. Lenzerini, N. Schweikardt (Eds.), *Data Exchange, Integration, and Streams*, in: Dagstuhl Follow-Ups, vol. 5, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2013, pp. 1–37, <https://doi.org/10.4230/DFU.Vol5.10452.1>.
- [87] C.H. Papadimitriou, *Computational Complexity*, Addison-Wesley, Reading, MA, USA, 1994.
- [88] C.H. Papadimitriou, M. Yannakakis, The complexity of facets (and some facets of complexity), *J. Comput. Syst. Sci.* 28 (2) (1984) 244–259, [https://doi.org/10.1016/0022-0000\(84\)90068-0](https://doi.org/10.1016/0022-0000(84)90068-0).
- [89] G. Paul, Approaches to abductive reasoning: an overview, *Artif. Intell. Rev.* 7 (2) (1993) 109–152, <https://doi.org/10.1007/bf00849080>.
- [90] R. Peñaloza, Axiom pinpointing, in: *Applications and Practices in Ontology Design, Extraction, and Reasoning*, IOS Press, 2020, pp. 162–177, <https://doi.org/10.3233/SSW200042>.
- [91] R. Peñaloza, B. Sertkaya, Axiom pinpointing is hard, in: *Proceedings of the 22nd International Workshop on Description Logics (DL 2009)*, 2009, pp. 39:1–39:12, http://ceur-ws.org/Vol-477/paper_39.pdf.
- [92] R. Peñaloza, B. Sertkaya, On the complexity of axiom pinpointing in the \mathcal{EL} family of description logics, in: *Proceedings of the 12th International Conference on Principles of Knowledge Representation and Reasoning (KR 2010)*, 2010, pp. 280–289, <https://lat.inf.tu-dresden.de/research/papers/2010/PeSe-KR10.pdf>.
- [93] R. Peñaloza, B. Sertkaya, Complexity of axiom pinpointing in the DL-Lite family of description logics, in: *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI 2010)*, 2010, pp. 29–34, <https://doi.org/10.3233/978-1-60750-606-5-29>.
- [94] R. Peñaloza, B. Sertkaya, Understanding the complexity of axiom pinpointing in lightweight description logics, *Artif. Intell.* 250 (2017) 80–104, <https://doi.org/10.1016/j.artint.2017.06.002>.
- [95] A. Poggi, D. Lembo, D. Calvanese, G.D. Giacomo, M. Lenzerini, R. Rosati, Linking data to ontologies, *J. Data Semant.* 10 (2008) 133–173, https://doi.org/10.1007/978-3-540-77688-8_5.
- [96] H.E. Pople, On the mechanization of abductive logic, in: *Proceedings of the 3rd International Joint Conference on Artificial Intelligence (IJCAI-73)*, 1973, pp. 147–152, <http://ijcai.org/Proceedings/73/Papers/017.pdf>.
- [97] E. Ramadan, A. Tarafdar, A. Pothen, A hypergraph model for the yeast protein complex network, in: *Proceedings of the 18th International Symposium on Parallel & Distributed Processing (IPDPS-04)*, 2004, pp. 189–196, <https://doi.org/10.1109/IPDPS.2004.1303205>.
- [98] R. Reiter, A theory of diagnosis from first principles, *Artif. Intell.* 32 (1) (1987) 57–95, [https://doi.org/10.1016/0004-3702\(87\)90062-2](https://doi.org/10.1016/0004-3702(87)90062-2).
- [99] B. Rossman, Homomorphism preservation theorems, *J. ACM* 55 (3) (2008) 15:1–15:53, <https://doi.org/10.1145/1379759.1379763>.
- [100] S. Schlobach, R. Cornet, Non-standard reasoning services for the debugging of description logic terminologies, in: *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-03)*, 2003, pp. 355–360, <http://ijcai.org/Proceedings/03/Papers/053.pdf>.
- [101] M. Schmidt-Schauß, G. Smolka, Attributive concept descriptions with complements, *Artif. Intell.* 48 (1) (1991) 1–26, [https://doi.org/10.1016/0004-3702\(91\)90078-x](https://doi.org/10.1016/0004-3702(91)90078-x).
- [102] R. Sebastiani, M. Vescovi, Axiom pinpointing in lightweight description logics via Horn-SAT encoding and conflict analysis, in: *Proceedings of the 22nd International Conference on Automated Deduction (CADE-09)*, 2009, pp. 84–99, https://doi.org/10.1007/978-3-642-02959-2_6.
- [103] G. Stefanoni, *Explaining Query Answers in Lightweight Ontologies: The DL-Lite Case*, MSC Thesis, Technische Universität Wien, 2011, <http://hdl.handle.net/20.500.12708/161189>.
- [104] L.J. Stockmeyer, The polynomial-time hierarchy, *Theor. Comput. Sci.* 3 (1) (1976) 1–22, [https://doi.org/10.1016/0304-3975\(76\)90061-x](https://doi.org/10.1016/0304-3975(76)90061-x).
- [105] B. Suntisrivaraporn, G. Qi, Q. Ji, P. Haase, A modularization-based approach to finding all justifications for OWL DL entailments, in: *Proceedings of the 3rd Asian Semantic Web Conference (ASWC 2008)*, 2008, pp. 1–15, https://doi.org/10.1007/978-3-540-89704-0_1.
- [106] E. Tsamoura, D. Carral, E. Malizia, J. Urbani, Materializing knowledge bases via trigger graphs, *Proc. VLDB Endow.* 14 (6) (2021) 943–956, <https://doi.org/10.14778/3447689.3447699>.
- [107] E. Tsamoura, T.M. Hospedales, L. Michael, Neural-symbolic integration: a compositional perspective, in: *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI-21)*, 2021, pp. 5051–5060, <https://doi.org/10.1609/AAAI.V35I6.16639>.
- [108] M.Y. Vardi, The complexity of relational query languages, in: *Proceedings of the 14th Annual ACM Symposium on Theory of Computing (STOC-82)*, 1982, pp. 137–146, <https://doi.org/10.1145/800070.802186>.
- [109] M.Y. Vardi, On the complexity of bounded-variable queries, in: *Proceedings of the 14th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS-95)*, 1995, pp. 266–276, <https://doi.org/10.1145/212433.212474>.
- [110] Y.R. Wang, S.E. Madnick, A polygen model for heterogeneous database systems: the source tagging perspective, in: *Proceedings of the 16th International Conference on Very Large Data Bases (VLDB-90)*, 1990, pp. 519–538, <http://www.vldb.org/conf/1990/P519.PDF>.
- [111] Z. Wang, M. Chitsaz, K. Wang, J. Du, Towards scalable and complete query explanation with OWL 2 EL ontologies, in: *Proceedings of the 24th ACM International Conference on Information and Knowledge Management (CIKM-15)*, 2015, pp. 743–752, <https://doi.org/10.1145/2806416.2806547>.
- [112] C. Wrathall, Complete sets and the polynomial-time hierarchy, *Theor. Comput. Sci.* 3 (1) (1976) 23–33, [https://doi.org/10.1016/0304-3975\(76\)90062-1](https://doi.org/10.1016/0304-3975(76)90062-1).