Explanations for Inconsistency-Tolerant Query Answering under Existential Rules

Thomas Lukasiewicz,[1] **Enrico Malizia,**[2] **Cristian Molinaro**[3]
[1]University of Oxford, UK, [2]University of Exeter, UK, [3]University of Calabria, Italy
thomas.lukasiewicz@cs.ox.ac.uk, e.malizia@exeter.ac.uk, cmolinaro@dimes.unical.it

Abstract

Querying inconsistent knowledge bases is a problem that has attracted a great deal of interest over the last decades. While several semantics of query answering have been proposed, and their complexity is rather well-understood, little attention has been paid to the problem of explaining query answers. Explainability has recently become a prominent problem in different areas of AI. In particular, explaining query answers allows users to understand not only what is entailed by an inconsistent knowledge base, but also *why*. In this paper, we address the problem of explaining query answers for existential rules under three popular inconsistency-tolerant semantics, namely, the ABox repair, the intersection of repairs, and the intersection of closed repairs semantics. We provide a thorough complexity analysis for a wide range of existential rule languages and for different complexity measures.

Introduction

Existential rules from the context of Datalog$^{\pm}$ and description logics (DLs) are popular ontology languages. In real-world ontology-based applications, it may very well be the case that the data are inconsistent with the ontology. To provide meaningful answers to users' queries in the presence of inconsistency, different inconsistency-tolerant semantics of query answering have been proposed over the years.

One of the most popular is the *ABox repair (AR)* semantics, first developed for relational databases (Arenas, Bertossi, and Chomicki 1999) and then generalized for several DLs (Lembo et al. 2010). Its basic idea is to consider a query answer valid if it can be inferred from each of the *repairs* of the knowledge base, that is, the inclusion-maximal consistent subsets of the database.

The *intersection of repairs (IAR)* (Lembo et al. 2010) and the *intersection of closed repairs (ICR)* (Bienvenu 2012) semantics have been introduced as approximations of the AR semantics. An answer is considered to be valid under the IAR (resp., ICR) semantics if it can be inferred from the intersection of the repairs (resp., the intersection of the closure of the repairs), along with the ontology. Besides being natural under-approximations of the AR semantics, they come with additional advantages of practical relevance. For instance, they are amenable to preprocessing, since the intersection of the (closed) repairs can be computed offline, and then standard query answering algorithms can be employed online. Indeed, the latter approach has been adopted in the implementation of the IAR semantics (Lembo et al. 2015), while for the ICR semantics, it has been already remarked in (Bienvenu and Bourgaux 2016).

While the complexity of the above semantics is rather well-understood (see, e.g., (Lembo et al. 2010; Bienvenu 2012; Bienvenu and Rosati 2013; Bienvenu, Bourgaux, and Goasdoué 2014; Lembo et al. 2015; Bienvenu and Bourgaux 2016) for inconsistency-tolerant query answering in DLs, and, e.g., (Lukasiewicz, Martinez, and Simari 2012; 2013; Lukasiewicz et al. 2015; Eiter, Lukasiewicz, and Predoiu 2016; Lukasiewicz, Malizia, and Molinaro 2018; Lukasiewicz, Malizia, and Vaicenavičius 2019) for inconsistency-tolerant query answering under existential rules), less attention has been paid to the problem of explaining query answers under such semantics. Explainability has recently become a prominent problem in different areas of AI. In our setting, explaining query answers allows users to understand not only what is entailed by an inconsistent knowledge base under a particular semantics, but also *why* it is entailed.

In this paper, we study explanations of query entailment under inconsistency-tolerant semantics in the presence of existential rules. Although DLs are popular formalisms for modeling ontologies, it is generally agreed that rule-based ontologies are well-suited for data intensive applications, since they allow us to conveniently deal with higher-arity relations, which naturally occur in standard relational databases.

Explaining query answers under inconsistency-tolerant semantics has been recently addressed in the literature (Arioua, Tamani, and Croitoru 2015; Hecham et al. 2017; Bienvenu, Bourgaux, and Goasdoué 2015; 2016; 2019). Specifically, Arioua, Tamani, and Croitoru (2015) addressed the problem of explaining query entailment under the ICR semantics in the presence of existential rules for which the Skolemized chase is finite. Their definition of explanation is based on abstract argumentation. Their approach along with interactive explanation methods based on dialectical

approaches has been experimentally evaluated by Hecham et al. (2017). In this paper, we also consider the AR and IAR semantics for several classes of existential rules (including classes for which the chase may not terminate, such as guarded and sticky existential rules), and provide a thorough complexity analysis under different complexity measures.

Bienvenu, Bourgaux, and Goasdoué (2015; 2016; 2019) considered the lightweight description logic DL-Lite$_{\mathcal{R}}$. They defined explanations for positive and negative answers under the brave, AR, and IAR semantics, and investigated the data complexity of different related problems. In this paper, in contrast, we consider a different formalism based on existential rules, define explanations under the ICR semantics, and carry out a complexity analysis also under the combined, bounded-arity-combined, and fixed-program-combined complexities, besides the data complexity.

Ceylan et al. (2019) studied the problem of explaining query answers under existential rules, restricting to *consistent* knowledge bases (consisting only of TGDs).

The contribution of this paper is a thorough complexity analysis for query explanations under the AR, IAR, and ICR semantics (as customary, we will focus on the decision variant of the problem), for a wide spectrum of Datalog$^{\pm}$ languages, and under the data, fixed-program-combined, bounded-arity-combined, and combined complexity measures.

Preliminaries

In this section, we briefly recall some basics on existential rules from the context of Datalog$^{\pm}$ (Calì, Gottlob, and Lukasiewicz 2012).

General. We assume a set $\mathbf{C}$ of *constants*, a set $\mathbf{N}$ of *labeled nulls*, and a set $\mathbf{V}$ of *variables*. A *term* t is a constant, null, or variable. We also assume a set of *predicates*, each associated with an arity, i.e., a non-negative integer. An *atom* has the form $p(t_1, \ldots, t_n)$, where p is an n-ary predicate, and $t_1, \ldots, t_n$ are terms. An atom containing only constants is also called a *fact*. Conjunctions of atoms are often identified with the sets of their atoms. An *instance* I is a (possibly infinite) set of atoms $p(\mathbf{t})$, where $\mathbf{t}$ is a tuple of constants and nulls. A *database* D is a finite instance that contains only constants. A *homomorphism* is a substitution $h\colon \mathbf{C} \cup \mathbf{N} \cup \mathbf{V} \to \mathbf{C} \cup \mathbf{N} \cup \mathbf{V}$ that is the identity on $\mathbf{C}$ and maps $\mathbf{N}$ to $\mathbf{C} \cup \mathbf{N}$. With a slight abuse of notation, homomorphisms are applied also to (sets/conjunctions of) atoms. A *conjunctive query* (CQ) q has the form $\exists \mathbf{Y} \phi(\mathbf{X}, \mathbf{Y})$, where $\phi(\mathbf{X}, \mathbf{Y})$ is a conjunction of atoms without nulls. The *answer* to q over an instance I, denoted $q(I)$, is the set of all tuples $\mathbf{t}$ over $\mathbf{C}$ for which there is a homomorphism h such that $h(\phi(\mathbf{X}, \mathbf{Y})) \subseteq I$ and $h(\mathbf{X}) = \mathbf{t}$. A *Boolean CQ* (BCQ) q is a CQ $\exists \mathbf{Y} \phi(\mathbf{Y})$, i.e., all variables are existentially quantified; q is *true* over I, denoted $I \models q$, if $q(I) \neq \emptyset$, i.e., there is a homomorphism h with $h(\phi(\mathbf{Y})) \subseteq I$.

Dependencies. A *tuple-generating dependency (TGD)* σ is a first-order formula $\forall \mathbf{X} \forall \mathbf{Y}\, \varphi(\mathbf{X}, \mathbf{Y}) \rightarrow \exists \mathbf{Z}\, p(\mathbf{X}, \mathbf{Z})$, where $\mathbf{X}$, $\mathbf{Y}$, and $\mathbf{Z}$ are pairwise disjoint sets of variables, $\varphi(\mathbf{X}, \mathbf{Y})$ is a conjunction of atoms, and $p(\mathbf{X}, \mathbf{Z})$ is an atom, all without nulls; $\varphi(\mathbf{X}, \mathbf{Y})$ is the *body* of σ, denoted $body(\sigma)$, while $p(\mathbf{X}, \mathbf{Z})$ is the *head* of σ, denoted $head(\sigma)$. For clarity, we consider single-atom-head TGDs; however, our results extend to TGDs with a conjunction of atoms in the head. An instance I satisfies σ, written $I \models \sigma$, if the following holds: whenever there exists a homomorphism h such that $h(\varphi(\mathbf{X}, \mathbf{Y})) \subseteq I$, then there exists $h' \supseteq h|_{\mathbf{X}}$, where $h|_{\mathbf{X}}$ is the restriction of h on $\mathbf{X}$, such that $h'(p(\mathbf{X}, \mathbf{Z})) \in I$. A *negative constraint (NC)* ν is a first-order formula $\forall \mathbf{X}\, \varphi(\mathbf{X}) \rightarrow \bot$, where $\mathbf{X} \subseteq \mathbf{V}$, $\varphi(\mathbf{X})$ is a conjunction of atoms without nulls, called the *body* of ν and denoted $body(\nu)$, and $\bot$ denotes the truth constant *false*. An instance I satisfies ν, written $I \models \nu$, if there is no homomorphism h such that $h(\varphi(\mathbf{X})) \subseteq I$. Given a set Σ of TGDs and NCs, I satisfies Σ, written $I \models \Sigma$, if I satisfies each TGD and NC of Σ. For brevity, we omit the universal quantifiers in front of TGDs and NCs, and use the comma (instead of $\wedge$) for conjoining atoms. Given a class of TGDs $\mathbb{C}$, we denote by $\mathbb{C}_{\bot}$ the formalism obtained by combining $\mathbb{C}$ with arbitrary NCs. Finite sets of TGDs and NCs are also called *programs*, and TGDs are also called *existential rules*.

Knowledge Bases. A *knowledge base* is a pair (D, Σ), where D is a database, and Σ is a program. For programs Σ, Σ_T and Σ_{NC} are the subsets of Σ containing the TGDs and NCs of Σ, respectively. The set of *models* of $KB = (D, \Sigma)$, denoted $mods(KB)$, is the set of instances $\{I \mid I \supseteq D \wedge I \models \Sigma\}$. We say that KB is *consistent* if $mods(KB) \neq \emptyset$, otherwise KB is *inconsistent*. The *answer* to a CQ q relative to KB is the set of tuples $ans(q, KB) = \bigcap\{q(I) \mid I \in mods(KB)\}$. The answer to a BCQ q is *true*, denoted $KB \models q$, if $ans(q, KB) \neq \emptyset$. The decision version of the *CQ answering* problem is: given a knowledge base KB, a CQ q, and a tuple of constants $\mathbf{t}$, decide whether $\mathbf{t} \in ans(q, KB)$. Since CQ answering can be reduced in LOGSPACE to BCQ answering, we focus on BCQs. Following Vardi (1982), the *combined complexity* of BCQ answering considers the database, the set of dependencies, and the query as part of the input. The *bounded-arity-combined* (or *ba-combined*) complexity assumes that the arity of the underlying schema is bounded by an integer constant. The *fixed-program-combined* (or *fp-combined*) *complexity* considers the sets of TGDs and NCs as fixed; the *data complexity* also assumes the query fixed.

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

We refer to (Eiter, Lukasiewicz, and Predoiu 2016) for a more detailed overview. Table 1 recalls complexity results of BCQ answering for the above languages, which are used in the complexity analysis done in this paper.

Complexity Classes. We briefly recall the complexity classes that we encounter. The complexity class AC^0 is the class of all decision problems that can be solved by uniform families of Boolean circuits of polynomial size and constant depth. PSPACE (resp., P, EXP, 2EXP) is the class of all problems that can be decided in polynomial space (resp., polynomial time, exponential time, double exponential time) on a deterministic Turing machine. NP and NEXP are the classes of all problems that are decidable in polynomial and exponential time on a nondeterministic Turing machine, respectively, and co-NP and co-NEXP are their complementary classes, where 'yes' and 'no' instances are interchanged. P^{NEXP} is the class of all problems that are decidable in deterministic polynomial time using a NEXP oracle. The class Σ_2^{P} is the class of all problems that can be decided in nondeterministic polynomial time using an NP oracle, and Π_2^{P} is the complement of Σ_2^{P}. The class $\text{D}^{\text{P}} = \text{NP} \wedge \text{co-NP}$ (resp., $\text{D}_2^{\text{P}} = \Sigma_2^{\text{P}} \wedge \Pi_2^{\text{P}}$) is the class of all problems that are the conjunction of a problem in NP (resp., Σ_2^{P}) and a problem in co-NP (resp., Π_2^{P}). The above complexity classes and their inclusion relationships (which are all currently believed to be strict) are: $\text{AC}^0 \subseteq \text{P} \subseteq \text{NP}, \text{co-NP} \subseteq \text{D}^{\text{P}} \subseteq \Sigma_2^{\text{P}}, \Pi_2^{\text{P}} \subseteq \text{D}_2^{\text{P}} \subseteq \text{PSPACE} \subseteq \text{EXP} \subseteq \text{NEXP}, \text{co-NEXP} \subseteq \text{P}^{\text{NEXP}} \subseteq \text{2EXP}$.

Inconsistency-Tolerant Semantics for Query Answering. We now recall three prominent inconsistency-tolerant semantics for ontology-based query answering under existential rules (Lukasiewicz, Malizia, and Molinaro 2018), namely, the *ABox repair (AR)* semantics, its approximation by the *intersection of repairs (IAR)*, and the *intersection of closed repairs (ICR)* semantics (Lembo et al. 2010; Bienvenu 2012); all three are based on the notion of *repair*, which is a maximal consistent subset of the given database.

Let $KB = (D, \Sigma)$ be a knowledge base. A *repair* of KB is an inclusion-maximal subset R of D such that $mods((R, \Sigma)) \neq \emptyset$. We use $Rep(KB)$ to denote the set of all repairs of KB. The *closure* $Cn(KB)$ of KB is the set of all atoms without variables, built from constants in D and Σ, entailed by D and the TGDs of Σ. Let q be a BCQ.

- KB entails q under the *ABox repair* (AR) *semantics* if, for all $R \in Rep(KB)$, it holds that $(R, \Sigma) \models q$.
- KB entails q under the *intersection of repairs* (IAR) *semantics* if $(D_I, \Sigma) \models q$, where $D_I = \bigcap\{R \mid R \in Rep(KB)\}$.
- KB entails q under the *intersection of closed repairs* (ICR) *semantics* if $(D_C, \Sigma) \models q$, where $D_C = \bigcap\{Cn((R, \Sigma)) \mid R \in Rep(KB)\}$.

We refer to (Lukasiewicz et al. 2015) and (Lukasiewicz, Malizia, and Molinaro 2018) for an overview of the complexity of AR- and IAR-/ICR-query answering, respectively, for different existential rule languages and complexity measures.

Explanations for Query Answers

In this section, we introduce the notions of explanations and minimal explanations, both under standard BCQ answering and under the AR, IAR, and ICR semantics. In the rest of this section, KB is a knowledge base (D, Σ), and q is a BCQ.

	Data	fp**-comb.**	ba**-comb.**	**Comb.**
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

Table 1: Complexity of BCQ answering (Lukasiewicz et al. 2015). All non-"in" entries are completeness results.

An *explanation* for q w.r.t. KB is a subset E of D such that (E, Σ) is consistent and $(E, \Sigma) \models q$. A *minimal explanation* E, or *MinEx*, for q w.r.t. KB is an explanation for q w.r.t. KB that is inclusion-minimal, i.e., there is no $E' \subsetneq E$ that is an explanation for q w.r.t. KB.[1] Unlike the definition of (minimal) explanations provided by Ceylan et al. (2019), we require a (minimal) explanation to be consistent, because in our setting the knowledge base can be inconsistent.

We now introduce the notions of explanation and minimal explanation under the AR, IAR, and ICR semantics.

Definition 1. An *AR-explanation* for q w.r.t. KB is a set of explanations $\mathcal{E} = \{E_1, \ldots, E_n\}$ for q w.r.t. KB such that every repair of KB contains some E_i.

An *IAR-explanation* for q w.r.t. KB is a singleton set of explanations $\mathcal{E} = \{E\}$ for q w.r.t. KB such that $E \subseteq R$ for every repair $R \in Rep(KB)$.

An *ICR-explanation* for q w.r.t. KB is a set of explanations $\mathcal{E} = \{E_1, \ldots, E_n\}$ for q w.r.t. KB such that *(i)* every repair of KB contains some E_i and *(ii)* $(E_C, \Sigma) \models q$, where $E_C = \bigcap\{Cn(E_i) \mid E_i \in \mathcal{E}\}$.

In the previous definition, an IAR-explanation could be analogously defined simply as an explanation E for q w.r.t. KB such that E is included in every repair of KB, rather than modeling it as the singleton set $\{E\}$. We chose the latter to treat all the three types of explanations in a uniform way (i.e., as sets of sets of facts) and thus ease presentation.

Definition 2. For any $S \in \{AR, IAR, ICR\}$, an S-explanation $\mathcal{E} = \{E_1, \ldots, E_n\}$ for q w.r.t. KB is an *S-minimal explanation*, or *S-MinEx*, if every $E_i \in \mathcal{E}$ is a MinEx for q w.r.t. KB, and no $\mathcal{E}' \subsetneq \mathcal{E}$ is an S-explanation for q w.r.t. KB.

Intuitively, S-minimal explanations are succinct summaries: besides explaining why a query is entailed under a particular inconsistency-tolerant semantics, they do not contain more information than actually needed for entailing the query. We point out that our definitions of AR-minimal and IAR-minimal explanations are equivalent to positive explanations under the AR and IAR semantics of (Bienvenu, Bourgaux, and Goasdoué 2016; 2019).

Example 3. Consider the database

$$D = \{Prof(p, cs), Postdoc(p, math), Group(g)\},$$

[1] Note that the concept of minimal explanations in this paper is equivalent to the concept of *causes* in (Bienvenu, Bourgaux, and Goasdoué 2016; 2019).

asserting that p is a professor working in the cs department, p is a postdoc working in the $math$ department, and g is a research group. Consider also the program Σ consisting of the following dependencies:

$$\begin{array}{lcl} Prof(X,Y) & \rightarrow & Researcher(X), \\ Postdoc(X,Y) & \rightarrow & Researcher(X), \\ Prof(X,Y) & \rightarrow & Dept(Y), \\ Postdoc(X,Y) & \rightarrow & Dept(Y), \\ Prof(X,Y), Postdoc(X,Z) & \rightarrow & \bot, \end{array}$$

expressing that $Prof$ and $Postdoc$ have $Researcher$ as domain and $Dept$ as range, and one cannot be both a professor and a postdoc.

It is easy to see that the knowledge base $KB = (D, \Sigma)$ is inconsistent, since p violates the negative constraint.

The knowledge base admits the following two repairs:

$$\begin{array}{ll} D' & = \{Prof(p, cs), Group(g)\}, \\ D'' & = \{Postdoc(p, math), Group(g)\}. \end{array}$$

Notice that the intersection of the two repairs is $D_I = \{Group(g)\}$, while the intersection of their closures is $D_C = \{Group(g), Researcher(p)\}$.

The Boolean query $\exists X\, Group(X)$ is entailed by KB under the IAR (and thus also under the ICR and AR) semantics. The set $\{\{Group(g)\}\}$ is an IAR-minimal (as well as ICR- and AR-minimal) explanation for the query w.r.t. KB. Indeed, $Group(g)$ is the fact in D_I that entails the query.

The Boolean query $\exists X\, Researcher(X)$ is entailed by KB under the ICR (and thus also under the AR) semantics, but not under the IAR semantics. The set $\{\{Prof(p, cs)\}, \{Postdoc(p, math)\}\}$ is an ICR-minimal (as well as AR-minimal) explanation for the query w.r.t. KB. Indeed, $Researcher(p)$ is the fact in D_C that entails the query, and the reason why $Researcher(p)$ belongs to the closures of D' and D'' are the facts $Prof(p, cs)$ and $Postdoc(p, math)$ of D' and D'', respectively.

The Boolean query $\exists X\, Dept(X)$ is entailed by KB only under the AR semantics. An AR-minimal explanation for the query w.r.t. KB is $\{\{Prof(p, cs)\}, \{Postdoc(p, math)\}\}$. Indeed, $Prof(p, cs)$ is the fact of D' entailing the query, while $Postdoc(p, math)$ is the fact of D'' entailing the query.

We point out that a natural decision counterpart of the problem of computing S-minimal explanations is deciding the existence of an S-minimal explanation. Such explanation exists iff the query is entailed under the S semantics, and hence the complexity of deciding the existence of an S-minimal explanation equals the complexity of inconsistency-tolerant reasoning, which has already been investigated in the literature. Thus, in this paper, we focus on the following decision problems: deciding whether a set of sets of facts is an S-MinEx for q w.r.t. KB, for each $S \in \{AR, IAR, ICR\}$.

Problem: S-MinEx, with $S \in \{AR, IAR, ICR\}$.
Input: A knowledge base $KB = (D, \Sigma)$, a BCQ q, and $\mathcal{E} \subseteq \mathcal{P}(D)$, with $\mathcal{P}(D)$ being the powerset of D.
Question: Is $\mathcal{E}$ an S-MinEx for q w.r.t. KB?

By the inconsistency-tolerant semantics definitions, if KB entails q under the ICR (resp., IAR) semantics, then KB entails q also under the AR (resp., ICR) semantics. Thus, a natural property is that an ICR (resp., IAR)-explanation should be also an AR (resp., ICR)-explanation (for the same query and knowledge base), and the same should hold when talking of minimal explanations. The following lemma states that this property is indeed enjoyed by our definitions of S-(minimal) explanations.

Lemma 4. *An ICR-(minimal) explanation for q w.r.t. KB is also an AR-(minimal) explanation for q w.r.t. KB.*

An IAR-(minimal) explanation for q w.r.t. KB is also an ICR-(minimal) explanation for q w.r.t. KB.

Overview of Complexity Results

We give a precise picture of the complexity of S-MinEx, for each $S \in \{AR, IAR, ICR\}$. An overview of the results is reported in Table 2 (AR and ICR semantics) and Table 3 (IAR semantics). Our results range from membership in P to 2EXP-completeness.

An interesting question is whether the complexity drops when moving from the AR semantics to the IAR and ICR semantics. Unfortunately, this is not the case for ICR, as the complexity of ICR-MinEx and AR-MinEx is the same in all cases. In contrast, the IAR semantics offers some computational benefits. In the data complexity, there is a decrease in complexity for all languages. Specifically, the complexity decreases from D^{P}-completeness to membership in P for linear, sticky, and acyclic existential rules (as well as their sublanguages), and to co-NP-completeness for the remaining languages. In the ba-combined complexity, there is a decrease in complexity only for linear, sticky, and full existential rules (and their sublanguages), with the complexity going from D_2^{P}- to Π_2^{P}-completeness. The complexity of IAR-MinEx remains the same as that of AR-MinEx for all fragments of existential rules in the *fp*-combined and combined complexities.

Another question is whether explaining query answering in the presence of inconsistency increases the complexity w.r.t. the consistent case, which was studied by Ceylan et al. (2019). Table 4 recalls complexity results for the IS-MINEX problem, that is, deciding whether a subset of the database is a MinEx for a query w.r.t. a knowledge base—notice that the languages reported in Table 4 do not include negative constraints, as the problem concerns *consistent* knowledge bases.

As expected, each semantics incurs an increase of complexity in some cases, even though there are cases where the complexity remains the same as in the consistent case.

More specifically, in the data complexity, moving from the consistent case to the IAR semantics, the complexity goes from P-completeness to co-NP-completeness only for guarded full existential rules and their generalizations ($\mathsf{G}_\bot$, $\mathsf{F}_\bot$, $\mathsf{WS}_\bot$, and $\mathsf{WA}_\bot$). When we move to the AR or ICR semantics, the complexity increases for all languages to D^{P}-completeness.

In the ba-combined complexity, when moving from IS-MINEX to IAR-MinEx (resp., AR- and ICR-MinEx), there is an increase of complexity from D^{P}-completeness to Π_2^{P}-completeness (resp., D_2^{P}-completeness) for linear, full, and sticky existential rules, and their sublanguages.

For acyclic existential rules, all the three inconsistency-tolerant semantics increase the ba-combined and combined

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

Table 2: Complexity of AR- and ICR-MinEx. All entries are completeness results. Hardness results in the data and fp-combined complexity also follow from (Bienvenu, Bourgaux, and Goasdoué 2019).

	Data	fp-comb.	ba-comb.	Comb.
$\mathsf{L}_\perp$, $\mathsf{LF}_\perp$, $\mathsf{AF}_\perp$	in P	D^{P}	Π_2^{P}	PSPACE
$\mathsf{S}_\perp$, $\mathsf{SF}_\perp$	in P	D^{P}	Π_2^{P}	EXP
$\mathsf{A}_\perp$	in P	D^{P}	$\mathrm{P}^{\mathrm{NEXP}}$	$\mathrm{P}^{\mathrm{NEXP}}$
$\mathsf{G}_\perp$	co-NP	D^{P}	EXP	2EXP
$\mathsf{F}_\perp$, $\mathsf{GF}_\perp$	co-NP	D^{P}	Π_2^{P}	EXP
$\mathsf{WS}_\perp$, $\mathsf{WA}_\perp$	co-NP	D^{P}	2EXP	2EXP

Table 3: Complexity of IAR-MinEx. All entries without "in" are completeness results.

complexities from $\mathrm{D}^{\mathrm{EXP}}$-completeness to $\mathrm{P}^{\mathrm{NEXP}}$-completeness.

In all the remaining cases, the complexity of the consistent and inconsistent cases is the same.

Deciding whether $\mathcal{E}$ is an S-MinEx involves checking whether $\mathcal{E}$ is an S-explanation and checking whether $\mathcal{E}$ is minimal as per Definition 2. The former problem is somehow related to the query answering problem under the S semantics (because we need to check whether $\mathcal{E}$ witnesses entailment under the S semantics), while the latter is related to the Is-MinEx problem (because it in turn requires checking if each element of $\mathcal{E}$ is a MinEx). Since, as discussed above, the complexity of S-MinEx is always at least that of Is-MinEx, an interesting question is: when the complexity of query answering under the S semantics is greater than that of Is-MinEx, does the complexity of S-MinEx get to that of query answering under the S semantics? The answer is that this is not always the case. For example, in the fp-combined complexity, Is-MinEx and ICR reasoning are D^{P}-complete and Θ_2^{p}-complete, respectively, in all the Datalog$^{\pm}$ languages considered. Nonetheless, ICR-MinEx is D^{P}-complete. Intuitively, the reason is that checking whether $\mathcal{E}$ witnesses entailment is somehow easier than checking whether entailment holds, because $\mathcal{E}$ is given, and thus we do not need to look for something witnessing entailment.

Complexity Analysis

In this section, we first discuss membership results and then hardness results.

Membership Results

The following theorem proves all the upper bounds in Table 2 for the AR semantics and in Table 3, but the ones in the fp-combined complexity and the P results in Table 3, for which we need tighter statements.

	Data	fp-comb.	ba-comb.	Comb.
L, LF, AF	in P	D^{P}	D^{P}	PSPACE
S, SF	in P	D^{P}	D^{P}	EXP
A	in P	D^{P}	$\mathrm{D}^{\mathrm{EXP}}$	$\mathrm{D}^{\mathrm{EXP}}$
G	P	D^{P}	EXP	2EXP
F, GF	P	D^{P}	D^{P}	EXP
WS, WA	P	D^{P}	2EXP	2EXP

Table 4: Complexity of Is-MinEx (Ceylan et al. 2019). All non-"in" entries are completeness results.

Theorem 5. *Let L be one of the Datalog$^{\pm}$ languages of this paper. If BCQ answering from knowledge bases over L is in $\mathbf{C}$, then AR-/IAR-MinEx can be answered by the following sequence of checks:*

(a) a co-$(\mathrm{NP}^{\mathbf{C}})$ check,

(b) (only for AR-MinEx) an $\mathrm{NP}^{\mathbf{C}}$ check,

(c) (only for AR-MinEx) a linear number of $\mathbf{C}$ checks/ (only for IAR-MinEx) a $\mathbf{C}$ check, and

(d) a linear number of co-$\mathbf{C}$ checks.

Proof sketch. Verifying that $\mathcal{E}$ is an S-MinEx for q w.r.t. KB requires checking the following conditions (we recall that for IAR-MinEx $\mathcal{E} = \{E\}$): (1) verify that all $E_i \in \mathcal{E}$ are MinExs of q w.r.t. KB (which implies verifying that: (1a) all $E_i \in \mathcal{E}$ are consistent; (1b) all $E_i \in \mathcal{E}$ entail q; and (1c) all $E_i \in \mathcal{E}$ are minimal); (2) verify that all the repairs are "covered" by $\mathcal{E}$, i.e., for each repair R there is a MinEx $E_i \in \mathcal{E}$ such that $R \supseteq E_i$; and (3) (only for AR-MinEx) verify that the "cover by $\mathcal{E}$ is minimal", i.e., there is no $\mathcal{E}' \subsetneq \mathcal{E}$ that "covers" all the repairs (for IAR-MinEx, we simply check that $|\mathcal{E}| = 1$).

We start by checking conditions (2) and (3), which give rise to the checks (a) and (b) in the statement, respectively. Checking these conditions at the beginning ensures that all $E_i \in \mathcal{E}$ are consistent (condition (1a)), and hence the subsequent checks of conditions (1b) and (1c) will be meaningful.

The complement of condition (2) is in $\mathrm{NP}^{\mathbf{C}}$: an NP machine M guesses a set $R \subseteq D$ such that, for all $E_i \in \mathcal{E}$, $R \not\supseteq E_i$. Then, via oracle calls in $\mathbf{C}$, M checks that R is a repair.

(*Only for AR-MinEx*) Condition (3) is in $\mathrm{NP}^{\mathbf{C}}$: minimality of $\mathcal{E}$ is proven by showing the "criticality" in $\mathcal{E}$ of all $E_i \in \mathcal{E}$, i.e., that there is a repair R_i such that $R_i \supseteq E_i$ and $R_i \not\supseteq E_j$, for all $j \neq i$ (see the notion of "critical vertex" in minimal transversals, Gottlob and Malizia 2014; 2018). Hence, an NP machine, for each $E_i \in \mathcal{E}$, guesses a set $E_i \subseteq R_i \subseteq D$, then, it checks that $R_i \not\supseteq E_j$ for all $j \neq i$, and via oracle calls in $\mathbf{C}$, R_i is checked to be a repair.

Checking conditions (1b) and (1c) give rise to checks (c) and (d) in the statement, respectively. To check (1b), with $|\mathcal{E}|$-many checks in $\mathbf{C}$, one for each $E_i \in \mathcal{E}$, we check that all $E_i \in \mathcal{E}$ entail the query. Condition (1c) can be verified by a linear number of checks in co-$\mathbf{C}$: for each set $E_i \in \mathcal{E}$, we remove each single fact in E_i in turn, and verify with a check in co-$\mathbf{C}$ that the set obtained does *not* entail the query. □

The following theorem proves all the upper-bounds in Table 2 for the ICR semantics in the data, *ba*-combined, and combined complexities. The *fp*-combined setting requires a tighter statement. The basic idea of the proof of this result is as follows. Verifying that a set $\mathcal{E} = \{E_1, \ldots, E_n\}$ is an ICR-MinEx for a query w.r.t. a knowledge base requires to check conditions (1), (2), and (3) in the proof of Theorem 5, and the additional condition that the intersection of the closure of all the E_i's has to entail the query. Verifying the latter can be reduced to ICR reasoning over a suitable knowledge base.

Theorem 6. *Let L be one of the Datalog$^{\pm}$ languages of this paper. If BCQ answering (resp., inconsistency-tolerant BCQ answering under the ICR semantics) from knowledge bases over L is in $\mathbf{C}$ (resp., in $\mathbf{D}$), then ICR-MinEx can be answered by the following sequence of checks:*

(a) a co-($\text{NP}^{\mathbf{C}}$) *check,*

(b) an $\text{NP}^{\mathbf{C}}$ *check,*

(c) a $\mathbf{D}$ *check, and*

(d) a linear number of co-$\mathbf{C}$ *checks.*

The following theorem proves the *fp*-combined complexity upper-bounds in Tables 2 and 3. The result is obtained from the proofs of Theorems 5 and 6 with two additional remarks. First, in the *fp*-combined setting, for the Datalog$^{\pm}$ languages considered, checking whether a set of facts is a repair is in P. Second, for the ICR case, we also need to notice that, in the *fp*-combined setting, checking whether the intersection of the closure of the E_i's entails the query is in NP.

Theorem 7. *AR-/IAR-/ICR-MinEx is in D^{P} in the fp-combined complexity for the Datalog$^{\pm}$ languages of this paper.*

The following theorem proves the P upper-bounds in Table 3. A key observation is that the intersection of the repairs in the stated fragments is computable in polynomial time (Lukasiewicz, Martinez, and Simari 2012; 2013).

Theorem 8. *IAR-MinEx from knowledge bases over $\mathsf{L}_\bot$, $\mathsf{A}_\bot$, and $\mathsf{S}_\bot$ is in P in the data complexity.*

Hardness Results

The hardness results not explicitly proven in this section follow from the hardness of deciding whether a set of facts is a MinEx over consistent KBs (Ceylan et al. 2019).

The two following theorems provide the co-NP-hardness and the Π^{p}_2-hardness results in Table 3, respectively. For the first, we use a reduction from the problem of deciding the unsatisfiability of a 3CNF Boolean formula $\phi(X)$, and for the second, a reduction from the problem of deciding the validity of a quantified Boolean formula $\forall X \exists Y \phi(X, Y)$, where $\phi(X, Y)$ is in 3CNF. The idea at the base of the reductions is to employ the repairs to encode assignments to the Boolean variables X, and then check whether the formulas $\phi(X)$ and $\phi(X, Y)$ are satisfi*ed* and satisfi*able*, respectively, in the given repair. The two constructions are considerably different, because the first result is in the data complexity and hence the program must be independent from $\phi(X)$, while in the second we can encode the satisfiability of $\phi(X, Y)$ in the TGDs of the program. The second reduction is a simplification of the one in the proof of Theorem 13.

Theorem 9. *IAR-MinEx from knowledge bases over $\mathsf{GF}_\bot$ is* co-NP-hard *in the data complexity.*

Theorem 10. *IAR-MinEx from knowledge bases over $\mathsf{LF}_\bot$, $\mathsf{AF}_\bot$, and $\mathsf{SF}_\bot$ is Π^{p}_2-hard in the ba-combined complexity.*

The following theorem provides the P^{NEXP}-hardness results in Tables 2 and 3. The result can be shown via a reduction from the problem ETP defined by Eiter, Lukasiewicz, and Predoiu (2016): given a triple (m, TP_1, TP_2), where m is a number in unary, and TP_1 and TP_2 are two tiling problems for the exponential square $2^n \times 2^n$, decide whether, for all initial tiling conditions w of length m, TP_1 has no solution with w *or* TP_2 has a solution with w. The idea of the reduction is to have the various repairs encoding the possible initial tiling conditions. Then, there are rules that allow to derive the query if TP_1 has no solution with the initial condition w encoded in the repair or TP_2 has a solution with w.

Theorem 11. *For any $S \in \{AR, IAR, ICR\}$, S-MinEx from knowledge bases over $\mathsf{A}_\bot$ is P^{NEXP}-hard in the ba-combined complexity.*

The following theorem provides the D^{P}-hardness results in Table 2. The results are obtained via a reduction from the D^{P}-complete problem MINIMAL UNSATISFIABILITY (Papadimitriou and Wolfe 1988): given a Boolean formula ϕ, decide whether ϕ is *minimally unsatisfiable*, which means decide whether ϕ is unsatisfiable, *and* removing any clause from ϕ makes the formula satisfiable.

In this case, a repair encodes an assignment for ϕ. We devise a reduction in which the candidate AR-/ICR-MinEx contains MinExs E_i encoding ways of not satisfying the clauses of ϕ. The various E_i cover all the repairs if ϕ is unsatisfiable, and they are also minimal if removing any clause from ϕ makes the formula satisfiable. We point out that an alternative proof can be obtained by translating the knowledge base used in the reduction of the proof of Proposition 5.12 in (Bienvenu, Bourgaux, and Goasdoué 2019) to existential rule languages.

Theorem 12. *For $S \in \{AR, ICR\}$, S-MinEx from knowledge bases over $\mathsf{LF}_\bot$, $\mathsf{AF}_\bot$, and $\mathsf{SF}_\bot$ is* D^{P}-hard *in the data complexity.*

The theorem below provides the D^{P}_2-hardness in Table 2.

Theorem 13. *For $S \in \{AR, ICR\}$, S-MinEx from knowledge bases over $\mathsf{LF}_\bot$, $\mathsf{AF}_\bot$, and $\mathsf{SF}_\bot$, is* D^{P}_2-hard *in the ba-combined complexity.*

Proof sketch. We reduce to S-MinEx, with $S \in \{AR, ICR\}$ the D^{P}_2-hard problem $\text{QBF}^{CNF}_{2,\forall,\neg} \wedge \text{QBF}^{CNF}_{2,\exists}$: given two quantified Boolean formulas $\Phi = \exists X \forall Y \neg \phi(X, Y)$ and $\Psi = \forall X \exists Y \psi(X, Y)$, decide whether Φ *and* Ψ are valid. The D^{P}_2-hardness holds even if ϕ and ψ are in 3CNF.

From an instance (Φ, Ψ) of $\text{QBF}^{CNF}_{2,\forall,\neg} \wedge \text{QBF}^{CNF}_{2,\exists}$, we build the following instance $(KB = (D, \Sigma), q, \mathcal{E})$ of S-MinEx. Arguing similarly as in the proof of Theorem 3.9 in (Lukasiewicz and Malizia 2017), we can assume w.l.o.g. that the variables X and Y of Φ and Ψ are the same.

From (Φ, Ψ), we build the database D as follows. For each variable $x_i \in X$, in D, there are facts:

$$Val(x_i, f) \qquad\qquad Val(x_i, t),$$

where x_i, f, and t are constants representing the variables in X and the Boolean values false and true, respectively.

There are facts in D that are used to impose the consistency of the truth assignments to the literals:

$$SimLit(f,f) \quad OppLit(f,t) \quad SimLit(t,t) \quad OppLit(t,f).$$

The predicate $SimLit(\cdot,\cdot)$ will be used to impose that when a variable appears twice as a positive or a negative literal in two different places in the formulas, then the two literals must have the same truth value. On the other hand, the predicate $OppLit(\cdot,\cdot)$ will be used to impose that when a variable appears as a positive literal in one place in the formula and as a negative literal in another place of the formula, then the two literals must have different truth values.

There are facts in D that are used to select possible ways of satisfying the clauses in the formulas:

$$ClSat(f,t,t) \quad ClSat(t,t,t) \quad ClSat(t,f,t) \quad ClSat(f,f,t)$$
$$ClSat(f,t,f) \quad ClSat(t,t,f) \quad ClSat(t,f,f).$$

The predicate $ClSat(\cdot,\cdot,\cdot)$ states what truth assignments to the *literals* (and not to the variables) satisfy a clause.

$SimLit$, $OppLit$, and $ClSat$, are the *structural* facts, and we denote by D^{St} the sets of structural facts of D.

To conclude, the following atoms are part of D: $Sat^{\phi}()$, $NonSat^{\phi}()$, $Sat_1^{\psi}()$, $Sat_2^{\psi}()$, and $NonSat^{\psi}()$.

We now describe the rules Σ. We start with the NCs, and then show the TGDs. The NCs' aim is to verify the validity of Φ and Ψ. An NC selects in a repair a meaningful assignment to the variables X, and other NCs encode the formulas.

The first NC above mentioned is as follows:

$$Val(X,f),\ Val(X,t) \rightarrow \bot.$$

The NCs encoding the formulas are built by various pieces. Some of these NCs check the validity of Φ, while others the validity of Ψ. These NCs can be told apart by the superscript ϕ and ψ, for formulas Φ and Ψ, respectively, used in some predicates. We now describe these various pieces. A first piece checks the presence of all the structural facts in a repair:

$$Config \equiv \bigwedge_{p(\mathbf{c}) \in D^{St}} p(\mathbf{c}).$$

A second piece "reads" onto the variables T_i the assignment on the variables in X encoded in the repair:

$$AssignX \equiv \bigwedge_{i=1}^{n} Val(x_i, T_i).$$

Below, we use this notation: $\ell_{j,k}^{\phi}$ (resp., $\ell_{j,k}^{\psi}$) is the k^{th} literal in the j^{th} clause of formula $\phi(X,Y)$ (resp., $\psi(X,Y)$), and $v_{j,k}^{\phi}$ (resp., $v_{j,k}^{\psi}$) is the variable of $\ell_{j,k}^{\phi}$ (resp., $\ell_{j,k}^{\psi}$).

A third piece "copies" the assignment of each variable x_i onto a positive occurrence of x_i in $\phi(X,Y)$ and $\psi(X,Y)$:

$$Copy^{\alpha} \equiv \bigwedge_{i=1}^{n} SimLit(T_i, T_{j,k}^{\alpha}),$$

where the superscript α is one among $\{\phi,\psi\}$, and in each predicate $SimLit(T_i^{\alpha}, T_{j,k}^{\alpha})$, $T_{j,k}^{\alpha}$ is a variable for the Boolean value of the literal $\ell_{j,k}^{\phi} = x_i$ (resp., $\ell_{j,k}^{\psi} = x_i$) in $\phi(X,Y)$ (resp., $\psi(X,Y)$).

A fourth piece forces the values f and t assigned to the variables $T_{j,k}^{\alpha}$, simulating the assignments to the literals to be consistent. Below, $\ell_{j,k}^{\alpha} \sim \ell_{j',k'}^{\alpha}$ means that literals $\ell_{j,k}^{\alpha}$ and $\ell_{j',k'}^{\alpha}$ are both positive or negative, while $\ell_{j,k}^{\alpha} \not\sim \ell_{j',k'}^{\alpha}$ means that one literal is positive and the other is negative.

$$Consist^{\alpha} \equiv \bigwedge_{\substack{\forall(\ell_{j,k}^{\alpha},\ell_{j',k'}^{\alpha}) \\ \text{s.t. } v_{j,k}^{\alpha}=v_{j',k'}^{\alpha} \wedge \\ \ell_{j,k}^{\alpha} \sim \ell_{j',k'}^{\alpha}}} SimLit(T_{j,k}^{\alpha}, T_{j',k'}^{\alpha}) \bigwedge_{\substack{\forall(\ell_{j,k}^{\alpha},\ell_{j',k'}^{\alpha}) \\ \text{s.t. } v_{j,k}^{\alpha}=v_{j',k'}^{\alpha} \wedge \\ \ell_{j,k}^{\alpha} \not\sim \ell_{j',k'}^{\alpha}}} OppLit(T_{j,k}^{\alpha}, T_{j',k'}^{\alpha}),$$

where the superscript α is a one among $\{\phi,\psi\}$, and $T_{j,k}^{\alpha}$ is a variable with the same meaning as above.

A last piece checks $\phi(X,Y)$ and $\psi(X,Y)$'s satisfiability:

$$Satisfied^{\alpha} \equiv \bigwedge_{j=1}^{m} ClSat(T_{j,1}^{\alpha}, T_{j,2}^{\alpha}, T_{j,3}^{\alpha}),$$

where the superscript α is one among $\{\phi,\psi\}$.

We can now state the NCs. The NCs for Φ are:

$$Config, AssignX, Copy^{\phi}, Consist^{\phi}, Satisfied^{\phi}, NonSat^{\phi}() \rightarrow \bot$$
$$Config, AssignX, NonSat^{\phi}(), Sat^{\phi}() \rightarrow \bot.$$

The NCs for Ψ are:

$$Config, AssignX, Copy^{\psi}, Consist^{\psi}, Satisfied^{\psi}, NonSat^{\psi}() \rightarrow \bot$$
$$Config, AssignX, NonSat^{\psi}(), Sat_1^{\psi}() \rightarrow \bot$$
$$Config, AssignX, NonSat^{\psi}(), Sat_2^{\psi}() \rightarrow \bot.$$

A last NC is:

$$Config, AssignX, NonSat^{\phi}(), Sat_2^{\psi}() \rightarrow \bot.$$

The TGDs are:

$$NonSat^{\phi}() \rightarrow Aux() \qquad Sat_2^{\psi}() \rightarrow Aux(),$$

which are linear (and hence guarded), acyclic, sticky, and full. The query is $q = Sat_1^{\psi}(), Aux()$. To conclude, the explanation is: $\mathcal{E} = \{E_1, E_2\}$, where $E_1 = \{Sat_1^{\psi}(), Sat_2^{\psi}()\}$ and $E_2 = \{Sat_1^{\psi}(), NonSat^{\phi}()\}$. It can be shown that (Φ,Ψ) is a 'yes'-instance iff $\mathcal{E}$ is a "minimal cover" of the repairs. □

Summary and Outlook

We have presented a complexity analysis of the problem of explaining query answering under three popular inconsistency-tolerant semantics, for a wide range of existential rules, and under different complexity measures. Explanations are a succinct summary why certain query answers hold. In some scenarios, the existential rules used to derive query answers from the explanation may be useful as well. Furthermore, visualization techniques may help to present all this in a more

illustrative way to users. Also, explanations may not necessarily be intended to users only, but also be used by AI and domain experts to develop, verify, and debug AI systems.

This paper opens up several interesting avenues for further research. An interesting problem is to analyze the complexity of other related problems, such as deciding whether a fact is *necessary* (i.e., belonging to every S-minimal explanation) or *relevant* (i.e., belonging to at least one S-minimal explanation), as done by Bienvenu, Bourgaux, and Goasdoué (2016; 2019) for DL-Lite$_{\mathcal{R}}$. Another interesting problem is explaining why a query is *not* entailed under an inconsistency-tolerant semantics—e.g., see (Bienvenu, Bourgaux, and Goasdoué 2016; 2019). Also, it would be interesting to study properties and complexity issues when natural ranking criteria among S-minimal explanations are defined.

Acknowledgments. This work was supported by the Alan Turing Institute under the UK EPSRC grant EP/N510-129/1, the AXA Research Fund, and the EPSRC grants EP/R013667/1, EP/L012138/1, and EP/M025268/1.

References

Arenas, M.; Bertossi, L. E.; and Chomicki, J. 1999. Consistent query answers in inconsistent databases. In *Proc. PODS*, 68–79.

Arioua, A.; Tamani, N.; and Croitoru, M. 2015. Query answering explanation in inconsistent Datalog+/– knowledge bases. In *Proc. DEXA*, 203–219.

Bienvenu, M., and Bourgaux, C. 2016. Inconsistency-tolerant querying of description logic knowledge bases. In *Reasoning Web*, 156–202.

Bienvenu, M., and Rosati, R. 2013. Tractable approximations of consistent query answering for robust ontology-based data access. In *Proc. IJCAI*, 775–781.

Bienvenu, M.; Bourgaux, C.; and Goasdoué, F. 2014. Querying inconsistent description logic knowledge bases under preferred repair semantics. In *Proc. AAAI*, 996–1002.

Bienvenu, M.; Bourgaux, C.; and Goasdoué, F. 2015. Explaining query answers under inconsistency-tolerant semantics over description logic knowledge bases (extended abstract). In *Proc. DL*.

Bienvenu, M.; Bourgaux, C.; and Goasdoué, F. 2016. Explaining inconsistency-tolerant query answering over description logic knowledge bases. In *Proc. AAAI*, 900–906.

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

Bienvenu, M. 2012. On the complexity of consistent query answering in the presence of simple ontologies. In *Proc. AAAI*, 705–711.

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

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

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

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

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

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

Gottlob, G., and Malizia, E. 2014. Achieving new upper bounds for the hypergraph duality problem through logic. In *Proc. LICS*, 43:1–43:10.

Gottlob, G., and Malizia, E. 2018. Achieving new upper bounds for the hypergraph duality problem through logic. *SIAM J. Comput.* 47(2):456–492.

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

Lembo, D.; Lenzerini, M.; Rosati, R.; Ruzzi, M.; and Savo, D. F. 2010. Inconsistency-tolerant semantics for description logics. In *Proc. RR*, 103–117.

Lembo, D.; Lenzerini, M.; Rosati, R.; Ruzzi, M.; and Savo, D. F. 2015. Inconsistency-tolerant query answering in ontology-based data access. *J. Web Sem.* 33:3–29.

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

Lukasiewicz, T.; Martinez, M. V.; Pieris, A.; and Simari, G. I. 2015. From classical to consistent query answering under existential rules. In *Proc. AAAI*, 1546–1552.

Lukasiewicz, T.; Malizia, E.; and Molinaro, C. 2018. Complexity of approximate query answering under inconsistency in Datalog+/–. In *Proc. IJCAI*, 1921–1927.

Lukasiewicz, T.; Malizia, E.; and Vaicenavičius, A. 2019. Complexity of inconsistency-tolerant query answering in Datalog+/– under cardinality-based repairs. In *Proc. AAAI*, 2962–2969.

Lukasiewicz, T.; Martinez, M. V.; and Simari, G. I. 2012. Inconsistency-tolerant query rewriting for linear Datalog+/–. In *Proc. Datalog 2.0*, 123–134.

Lukasiewicz, T.; Martinez, M. V.; and Simari, G. I. 2013. Complexity of inconsistency-tolerant query answering in Datalog+/–. In *Proc. OTM*, 488–500.

Papadimitriou, C. H., and Wolfe, D. 1988. The complexity of facets resolved. *J. Comput. Syst. Sci.* 37(1):2–13.

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