Complexity of Inconsistency-Tolerant Query Answering in Datalog+/– under Cardinality-Based Repairs

Thomas Lukasiewicz
Department of Computer Science
University of Oxford, UK

Enrico Malizia
Department of Computer Science
University of Exeter, UK

Andrius Vaicenavičius
Department of Computer Science
University of Oxford, UK

Abstract

Querying inconsistent ontological knowledge bases is an important problem in practice, for which several inconsistency-tolerant query answering semantics have been proposed, including query answering relative to all repairs, relative to the intersection of repairs, and relative to the intersection of closed repairs. In these semantics, one assumes that the input database is erroneous, and the notion of repair describes a maximally consistent subset of the input database, where different notions of maximality (such as subset and cardinality maximality) are considered. In this paper, we give a precise picture of the computational complexity of inconsistency-tolerant (Boolean conjunctive) query answering in a wide range of Datalog$^{\pm}$ languages under the cardinality-based versions of the above three repair semantics.

Introduction

In many ontology-based applications in practice, such as ontology-based data extraction from the Web, or ontology-based integration of different data sources, it is very likely that the data are inconsistent with the ontology, and thus inconsistency-tolerant semantics for ontology-based query answering are urgently needed. Among the most prominent ontology languages are description logics (DLs) and existential rules from the context of Datalog$^{\pm}$.

The most widely accepted semantics for querying inconsistent ontological knowledge bases is perhaps *consistent query answering*, which was first developed for relational databases (Arenas, Bertossi, and Chomicki 1999) and then generalized as the ABox repair (AR) semantics for several DLs (Lembo et al. 2010). Consistent query answering is based on the concept of repair, which is a maximal consistent subset of the input database. A fact/query is entailed by an ontological knowledge base in consistent query answering, if it is (classically) entailed by all the repairs (under the ontology). Several other repair semantics for querying inconsistent ontological knowledge bases have recently been developed as alternatives to consistent query answering (Lembo et al. 2010; Bienvenu 2012; Lukasiewicz, Martinez, and Simari 2012; Bienvenu and Rosati 2013). In particular, in (Lembo et al. 2010), besides consistent query answering, three other inconsistency-tolerant query answering semantics are proposed, including the *intersection of repairs (IAR)* semantics, in which an answer is considered to be valid, if it can be inferred from the intersection of the repairs (and the ontology). The *intersection of closed repairs (ICR)* (Bienvenu 2012) is another semantics, in which an answer is valid, if it can be inferred from the intersection of the closure of the repairs (and the ontology).

There are several reasons for the practical relevance of the IAR and the ICR semantics, and thus for motivating an in-depth analysis of their computational properties. First, they are two natural semantics that identify "surer" answers than consistent query answering, and so they can also be seen as under-approximations of the latter. Investigating their complexity helps to understand whether such approximations have actually lower complexities, which is the case for different languages and one complexity measure considered in this paper. Second, recent work on explanations in the context of inconsistency-tolerant query answering shows that explanations are much easier to define and compute for the IAR semantics (Bienvenu, Bourgaux, and Goasdoué 2016). Third, a crucial advantage of the IAR and the ICR semantics is that their intersection of (closed) repairs can be materialized, while the consistent query answering semantics exists only virtually (the intersection of (closed) repairs can be computed offline, and then standard querying algorithms can be employed online)—indeed, this has been used to implement the IAR semantics (Lembo et al. 2015), and for ICR, it has been remarked in (Bienvenu and Bourgaux 2016).

The complexity of consistent query answering when the ontology is described via one of the main DLs is well-understood. Rosati (2011) studied the data and combined complexity for a wide spectrum of DLs, while Bienvenu (2012) identified cases for simple ontologies (within the *DL-Lite* family) for which tractable data complexity results can be obtained. In (Lukasiewicz, Martinez, and Simari 2012; 2013; Lukasiewicz et al. 2015), the data and different types of combined complexity of consistent query answering have been studied for ontologies described via existential rules and negative constraints, and such investigation has recently been extended to the IAR and ICR semantics in (Lukasiewicz, Malizia, and Molinaro 2018a).

Alternative notions of maximality for repairs, however, such as cardinality-maximal repairs (Lopatenko and Bertossi 2007), rather than subset-maximal ones, have been

explored less. Bienvenu, Bourgaux, and Goasdoué (2014) analyzed the data and the combined complexity of query answering under the AR and IAR semantics over the language *DL-Lite*$_{\mathcal{R}}$ for different notions of maximal repairs, among which the notion of maximum cardinality repairs.

This paper continues this line of research on cardinality-maximal consistent query answering, and we analyze the complexity of the above three inconsistency-tolerant query answering semantics for a wide range of Datalog$^{\pm}$ languages and for several different complexity measures:

- ▷ We consider different popular inconsistency-tolerant semantics, namely, the AR, the IAR, and the ICR semantics, with maximum cardinality database repairs.
- ▷ We consider the most popular Datalog$^{\pm}$ languages: linear, guarded, sticky, and acyclic existential rules, along with "weak" generalizations, as well as full (i.e., non-existential) restrictions, and full rules in general.
- ▷ We analyze the data, fixed-program combined, bounded-arity combined, and combined complexity.

Detailed proofs are given in a forthcoming extended paper.

Preliminaries

We now briefly recall some basics on Datalog$^{\pm}$ (Calì, Gottlob, and Lukasiewicz 2012) and the complexity classes that we will encounter in our analysis in this paper.

General. We assume sets $\mathbf{C}$, $\mathbf{N}$, and $\mathbf{V}$ of *constants*, *labeled nulls*, and *variables*, respectively. A *term* t is a constant, null, or variable. We also assume a set of *predicates*, each 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. 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 without nulls. A *homomorphism* is a mapping $h\colon \mathbf{C} \cup \mathbf{N} \cup \mathbf{V} \rightarrow \mathbf{C} \cup \mathbf{N} \cup \mathbf{V}$ that is the identity on $\mathbf{C}$ and maps $\mathbf{N}$ to $\mathbf{C} \cup \mathbf{N}$. 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 $\mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z} \subseteq \mathbf{V}$, $\varphi(\mathbf{X}, \mathbf{Y})$ 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 can be extended 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, 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 body 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 $\{I \mid I \supseteq D, I \models \Sigma\}$. 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 as follows: 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.

Decidability Paradigms. The main (syntactic) conditions on TGDs that guarantee the decidability of BCQ answering are guardedness (Calì, Gottlob, and Kifer 2013), stickiness (Calì, Gottlob, and Pieris 2012), and acyclicity, each having a "weak" counterpart: weak guardedness (Calì, Gottlob, and Kifer 2013), weak stickiness (Calì, Gottlob, and Pieris 2012), and weak acyclicity (Fagin et al. 2005).

A TGD σ is *guarded*, if there exists an atom in the body that contains (or "guards") all the body variables of σ. The class of guarded TGDs, denoted G, is the family of all possible sets of guarded TGDs. A key subclass of guarded TGDs are *linear* TGDs with just one body atom (which is a guard), and the corresponding class is denoted L. *Weakly guarded* TGDs extend guarded TGDs by requiring only "harmful" body variables to appear in the guard, and the associated class is denoted WG. It is easy to verify that $\mathsf{L} \subset \mathsf{G} \subset \mathsf{WG}$.

Stickiness is inherently different from guardedness, and its central property is as follows: variables that appear more than once in a 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. Weak stickiness is a relaxation of stickiness where only "harmful" variables are taken into account. A set of TGDs that enjoys weak stickiness is *weakly sticky*, and the associated class is denoted WS. Observe that $\mathsf{S} \subset \mathsf{WS}$.

A set Σ of TGDs is *acyclic*, if its predicate graph is acyclic, and the underlying class is denoted A. Σ is *weakly acyclic*, if its dependency graph enjoys a certain acyclicity condition, which actually guarantees the existence of a finite canonical model; the associated class is denoted WA. Clearly, $\mathsf{A} \subset \mathsf{WA}$. Observe also that $\mathsf{WA} \subset \mathsf{WS}$.

Another key fragment of TGDs which deserves our attention are the so-called *full* TGDs, i.e., TGDs without existentially quantified variables, and the corresponding class is denoted F. If full TGDs enjoy linearity, guardedness, stickiness, or acyclicity, then we obtain the classes LF, GF, SF, and AF, respectively. Observe that $\mathsf{F} \subset \mathsf{WA}$ and $\mathsf{F} \subset \mathsf{WG}$.

For a summary of the complexity of (classical) BCQ answering over these Datalog$^{\pm}$ languages, see Table 1.

Complexity Classes. 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. 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 Θ_2^{p} (resp., Θ_3^{p}) is the class of all problems that can be decided in polynomial time by a deterministic Turing machine with either a logarithmic number of calls to an NP (resp., Σ_2^{p}) oracle, or (equivalently) a constant number of rounds of polynomially many parallel calls to an NP (resp., Σ_2^{p}) oracle. P^{NEXP} is the class of all problems that are decidable in deterministic polynomial time using a NEXP oracle. The above complexity classes and their inclusion relationships (which are all currently believed to be strict) are: $\text{AC}^0 \subseteq \text{P} \subseteq \text{NP}, \text{co-NP} \subseteq \Theta_2^{\text{p}} \subseteq \Sigma_2^{\text{p}}, \Pi_2^{\text{p}} \subseteq \Theta_3^{\text{p}} \subseteq \text{PSPACE} \subseteq \text{EXP} \subseteq \text{NEXP}, \text{co-NEXP} \subseteq \text{P}^{\text{NEXP}} \subseteq \text{2EXP}$.

Inconsistency-Tolerant Semantics

In classical BCQ answering, for an inconsistent knowledge base KB (i.e., $mods(KB) = \emptyset$), every query is entailed, as everything follows from a contradiction. Clearly, the answers obtained in such cases are not meaningful.

Several inconsistency-tolerant semantics have been proposed. We now recall three prominent ones for ontology-based query answering, namely, the *ABox repair (AR)*, 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. Below we define these inconsistency-tolerant semantics for a generic concept of repair maximality.

Given a knowledge base $KB = (D, \Sigma)$, a *selection* D' of KB is a database such that $D' \subseteq D$. A selection D' of KB is *consistent*, if $mods((D', \Sigma)) \neq \emptyset$. Consistent selections of knowledge bases can be ordered according to some criteria to select the more desired ones. Given a preorder $\preccurlyeq$ over a set $\mathcal{S}$ of databases, for two elements D' and $D'' \in \mathcal{S}$, $D' \prec D''$ denotes that $D' \preccurlyeq D''$ and $D'' \not\preccurlyeq D'$. A database $D \in \mathcal{S}$ is $\preccurlyeq$-maximal in $\mathcal{S}$ iff there is no $D' \in \mathcal{S}$ such that $D \prec D'$.

Definition 1. A *$\preccurlyeq$-repair* of a knowledge base KB is a consistent selection of KB that is $\preccurlyeq$-maximal in the set of all the consistent selections of KB.

We now define the three different inconsistency-tolerant semantics for BCQ answering. In what follows, $Rep_{\preccurlyeq}(KB)$ denotes the set of all $\preccurlyeq$-repairs of a knowledge base KB. For a knowledge base $KB = (D, \Sigma)$, the *closure* $Cn(KB)$ of KB is the set of all ground atoms, built from constants in D and Σ, entailed by D and the TGDs of Σ.

Definition 2. Let KB be a knowledge base, let Q be a BCQ, and let $\preccurlyeq$ be an order over the consistent selections of KB.

- KB entails Q under the *ABox repair semantics and order $\preccurlyeq$ ($\preccurlyeq$-AR)*, denoted by $KB \models_{\preccurlyeq\text{-}AR} Q$, if, for all $D' \in Rep_{\preccurlyeq}(KB)$, $(D', \Sigma) \models Q$.
- KB entails Q under the *intersection of repairs semantics and order $\preccurlyeq$ ($\preccurlyeq$-IAR)*, denoted by $KB \models_{\preccurlyeq\text{-}IAR} Q$, if $(D^*, \Sigma) \models Q$, where $D^* = \bigcap\{D' \mid D' \in Rep_{\preccurlyeq}(KB)\}$.
- KB entails Q under the *intersection of closed repairs semantics and order $\preccurlyeq$ ($\preccurlyeq$-ICR)*, denoted by $KB \models_{\preccurlyeq\text{-}ICR} Q$, if $(D_I, \Sigma) \models Q$, where $D_I = \bigcap\{Cn((D', \Sigma)) \mid D' \in Rep_{\preccurlyeq}(KB)\}$.

Different orders over the set of consistent selections give rise to different inconsistency-tolerant semantics for BCQ answering, because they select different repairs.

Inclusion-maximal repairs have often been considered in the literature. An interesting class of repairs are those selected by the cardinality order '$\leq$' (Bienvenu, Bourgaux, and Goasdoué 2014). In this order, consistent selections of larger cardinality are preferred over ones of smaller cardinality (without looking at inclusion-wise relationships between the selections: only the cardinality counts). Hence, a $\leq$-repair of a knowledge base KB is a consistent selection of KB of maximum cardinality. In this paper, we consider only the '$\leq$' order, therefore, we often call $\leq$-repairs simply repairs, and by $Rep(KB)$, we mean $Rep_{\leq}(KB)$.

Cardinality-maximal repairs are very appropriate when it is known (or believed) that all the facts in the database have the same (possibly small) probability of being erroneous. In these cases, larger repairs are preferred, because fewer facts are dropped (Bienvenu, Bourgaux, and Goasdoué 2014). When facts in the database have different likelihoods of being erroneous, then other concepts of repairs can also be taken into consideration (Bienvenu 2012).

It can be shown that the following semantic relationships hold between the above three different semantics.

Proposition 3. *Let KB be a knowledge base, let Q a BCQ.* $(KB \models_{\leq\text{-}IAR} Q) \Rightarrow (KB \models_{\leq\text{-}ICR} Q) \Rightarrow (KB \models_{\leq\text{-}AR} Q)$.

Overview of Complexity Results

We give a precise picture of the complexity of BCQ answering from existential rules under the $\leq$-AR, $\leq$-IAR, and $\leq$-ICR inconsistency-tolerant semantics. Our results are summarized in Tables 2 and 3, and they range from Θ_2^{p}-completeness to 2EXP-completeness.

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

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

	Data	*fp*-comb.	*ba*-comb.	Comb.
$\mathsf{L}_\perp$, $\mathsf{LF}_\perp$, $\mathsf{AF}_\perp$	$\Theta_2^{\text{P}+*}$	Π_2^{P}	Θ_3^{P}	PSPACE
$\mathsf{G}_\perp$	$\Theta_2^{\text{P}+*}$	Π_2^{P}	EXP	2EXP
$\mathsf{WG}_\perp$	EXP	EXP	EXP	2EXP
$\mathsf{S}_\perp$, $\mathsf{SF}_\perp$	$\Theta_2^{\text{P}+*}$	Π_2^{P}	Θ_3^{P}	EXP
$\mathsf{F}_\perp$, $\mathsf{GF}_\perp$	$\Theta_2^{\text{P}+}$	Π_2^{P}	Θ_3^{P}	EXP
$\mathsf{A}_\perp$	$\Theta_2^{\text{P}+}$	Π_2^{P}	P^{NEXP}	P^{NEXP}
$\mathsf{WS}_\perp$, $\mathsf{WA}_\perp$	$\Theta_2^{\text{P}+*}$	Π_2^{P}	2EXP	2EXP

Table 2: Complexity of $\leq$-AR BCQ answering. All entries are completeness results. $^+$Different proof of membership in (Bienvenu, Bourgaux, and Goasdoué 2014). *Different proof of hardness for $\mathsf{L}_\perp$, $\mathsf{G}_\perp$, $\mathsf{S}_\perp$, and $\mathsf{WS}_\perp$ in (Bienvenu, Bourgaux, and Goasdoué 2014).

Compared to (Lukasiewicz, Malizia, and Molinaro 2018a; 2018b), where subset maximality is considered for the inconsistency-tolerant semantics, we observe that using the maximum cardinality comes at a cost in several cases. This increased cost is due to the computational effort needed to evaluate the size of the largest consistent selections of the database. This extra effort does not always influence the overall complexity of the problem. Indeed, in some circumstances it is masked out by the complexity of classical BCQ answering or the complexity associated with the evaluation of the inconsistency-tolerant semantics.

In detail, $\leq$-AR-BCQ answering (see Table 2) is complete for Θ_2^{p} (resp., Π_2^{p}) in the data (resp., *fp*-combined) complexity for all languages of existential rules, but for $\mathsf{WG}_\perp$, where it is EXP-complete. In the data complexity, except for the $\mathsf{WG}_\perp$ case, for which already classical BCQ answering over consistent ontologies is EXP-complete, the complexity of computing the size of the maximal-cardinality repairs dominates the complexity of the task. For the *fp*-combined complexity case, on the other hand, there is no increase in the complexity compared to the case of AR semantics with subset-maximal repairs (see Lukasiewicz et al. 2015).

The *ba*-combined complexity for $\leq$-AR-BCQ answering is among Θ_3^{p} (for $\mathsf{L}_\perp$, $\mathsf{LF}_\perp$, $\mathsf{AF}_\perp$, $\mathsf{S}_\perp$, $\mathsf{SF}_\perp$, $\mathsf{F}_\perp$, and $\mathsf{GF}_\perp$), EXP (for $\mathsf{G}_\perp$ and $\mathsf{WG}_\perp$), P^{NEXP} (for $\mathsf{A}_\perp$), and 2EXP (for $\mathsf{WS}_\perp$ and $\mathsf{WA}_\perp$). If we compare these results with those for the subset-maximal AR semantics, we observe that there is an increase in the complexity only for the languages whose classical reasoning in the *ba*-combined complexity is in NP

	Data	*fp*-comb.	*ba*-comb.	Comb.
$\mathsf{L}_\perp$, $\mathsf{LF}_\perp$, $\mathsf{AF}_\perp$	$\Theta_2^{\text{P}+*}$	Θ_2^{P}	Θ_3^{P}	PSPACE
$\mathsf{G}_\perp$	$\Theta_2^{\text{P}+*}$	Θ_2^{P}	EXP	2EXP
$\mathsf{WG}_\perp$	EXP	EXP	EXP	2EXP
$\mathsf{S}_\perp$, $\mathsf{SF}_\perp$	$\Theta_2^{\text{P}+*}$	Θ_2^{P}	Θ_3^{P}	EXP
$\mathsf{F}_\perp$, $\mathsf{GF}_\perp$	$\Theta_2^{\text{P}+}$	Θ_2^{P}	Θ_3^{P}	EXP
$\mathsf{A}_\perp$	$\Theta_2^{\text{P}+}$	Θ_2^{P}	P^{NEXP}	P^{NEXP}
$\mathsf{WS}_\perp$, $\mathsf{WA}_\perp$	$\Theta_2^{\text{P}+*}$	Θ_2^{P}	2EXP	2EXP

Table 3: Complexity of $\leq$-IAR and $\leq$-ICR BCQ answering. All entries are completeness results. $^+$Different proof of membership for IAR in (Bienvenu, Bourgaux, and Goasdoué 2014). *Different proof of hardness for IAR for $\mathsf{L}_\perp$, $\mathsf{G}_\perp$, $\mathsf{S}_\perp$, and $\mathsf{WS}_\perp$ in (Bienvenu, Bourgaux, and Goasdoué 2014).

(see, Lukasiewicz et al. 2015). Intuitively, in order to compute the size of the largest consistent selections, we need to perform a binary search asking an oracle to guess a selection of convenient size and check that it is consistent. This requires a Σ_2^{p} oracle, from which it follows that the overall procedure requires a computation in Θ_3^{p}. The combined complexity of $\leq$-AR-BCQ answering is among PSPACE (for $\mathsf{L}_\perp$, $\mathsf{LF}_\perp$, and $\mathsf{AF}_\perp$), EXP (for $\mathsf{S}_\perp$, $\mathsf{SF}_\perp$, $\mathsf{F}_\perp$, and $\mathsf{GF}_\perp$), P^{NEXP} (for $\mathsf{A}_\perp$), and 2EXP (for $\mathsf{G}_\perp$, $\mathsf{WS}_\perp$, $\mathsf{WA}_\perp$, and $\mathsf{WG}_\perp$). In these cases, we do not observe any increase in the complexity of the tasks, compared with the complexity AR semantics with subset-maximal repairs (see, Lukasiewicz et al. 2015).

The complexity of $\leq$-IAR- and $\leq$-ICR-BCQ answering (see Table 3) slightly drops to Θ_2^{p} in the *fp*-combined complexity for all languages, but $\mathsf{WG}_\perp$. As we can see, the complexity of IAR- and ICR-BCQ answering is the same when cardinality-maximal repairs are considered, because either the complexity of classical reasoning or the complexity of computing the size of the biggest repairs dominate the complexity of the task. For this reason, since there is no extra computational cost for using ICR instead of IAR, in this setting ICR has an advantage over IAR given that ICR is finer approximation of AR than IAR.

Derivation of Complexity Results

In this section, we first derive the membership results and then the hardness results of Tables 2 and 3.

Membership Results

The following theorem shows that if BCQ answering from knowledge bases over some Datalog$^\pm$ language L is in $\mathbf{C}$ in the data (resp., *ba*-combined and combined) complexity, then $\leq$-AR-BCQ and $\leq$-IAR-BCQ answering from knowledge bases over L can be done in polynomial time with logarithmically many calls to an oracle for $\text{NP}^{\mathbf{C}}$ in the data (resp., *ba*-combined and combined) complexity. This result holds for $\leq$-ICR-BCQ answering as well, but only in the data and *ba*-combined complexity. For the combined complexity case of $\leq$-ICR-BCQ answering, we will need a different proof.

Theorem 4. *Let L be a Datalog$^{\pm}$ language. If BCQ answering from knowledge bases over L is in $\mathbf{C}$ in the data / ba-combined / combined complexity (resp., data / ba-combined complexity), then $\leq$-AR and $\leq$-IAR (resp., $\leq$-ICR) BCQ answering from knowledge bases over L is in* P *with an oracle for* NP$^{\mathbf{C}}[O(\log n)]$ *in the data / ba-combined / combined complexity (resp., data / ba-combined complexity).*

Proof. Let $KB = (D, \Sigma)$ be a knowledge base over L, and let q be a query. First, we compute the size max of the largest consistent selections of KB. This can be done by a machine in polynomial time calling, logarithmically many times, an oracle in NP$^{\mathbf{C}}$. Indeed, we can perform a binary search in the range $[0, |D|]$ by asking the oracle whether there is a consistent selection of size at least k. The oracle has to guess a subset of size k of the database, and then check whether the guess is consistent. Since we are assuming that classical reasoning in fragment L is in $\mathbf{C}$, the oracle can guess a selection in NP and then check its consistency in $\mathbf{C}$.

Then, we exploit the NP$^{\mathbf{C}}$ oracle to decide whether the BCQ q is entailed under the considered semantics. In particular, we ask the oracle whether the query is *not* entailed. The way in which the oracle finds the answer depends on the specific semantics, AR, IAR, or ICR, considered.

- AR**:** The oracle guesses a consistent selection D' of size max, and checks that $(D', \Sigma) \not\models q$.
- IAR**:** The oracle guesses a database $D^\star \subseteq D$ and checks that: (1) $(D^\star, \Sigma) \not\models q$, and (2) there exist repairs D'_α with $\alpha \notin D'_\alpha$, one for each $\alpha \in D \setminus D^\star$ (witnessing that the intersection of the repairs is a subset of $D^\star$). Point (2) is checked via additional guesses of the various D'_α.
- ICR**:** The oracle verifies the existence of a subset D' of $Cn(KB)$ (the size of $Cn(KB)$ is polynomial in the input, because the program has, in the worst case, bounded arity) such that: (i) for each atom $\alpha \in (Cn(KB) \setminus D')$, there is a repair D'_α such that $\alpha \notin Cn(D'_\alpha)$; (ii) $(D', \Sigma) \not\models q$. The oracle can do this by first guessing such a set D' along with the witnesses D'_α, and second checking that the various D'_α are consistent, their size is max, and $\alpha \notin Cn(D'_\alpha)$, and that $(D', \Sigma) \not\models q$. The existence of the various repairs D'_α with the properties above reported guarantees that $D_I \subseteq D'$, and $(D', \Sigma) \not\models q$ implies $(D_I, \Sigma) \not\models q$ by monotonicity. □

As mentioned above, to characterize the combined complexity of $\leq$-ICR-BCQ answering, we need a different proof. Indeed, in a combined complexity scenario, the size of $Cn(KB)$ is exponential in the size of the input, and the $D' \subseteq Cn(KB)$ needed to be guessed in the ICR case of the proof above could be too big for an NP machine.

The next result proves all the upper bounds for $\leq$-ICR BCQ answering in Table 3 for the combined complexity.

Theorem 5. *$\leq$-ICR BCQ answering from knowledge bases over Datalog$^{\pm}$ languages L in the combined complexity is in the complexity classes shown in Table 3.*

We next analyze the *fp*-combined complexity and show that if BCQ answering from knowledge bases over some Datalog$^{\pm}$ language L is in $\mathbf{D}$ (resp., $\mathbf{C}$) in the data (resp., *fp*-combined) complexity, then $\leq$-AR, $\leq$-IAR, and $\leq$-ICR, BCQ answering from knowledge bases over L can be done in polynomial time with logarithmically many calls to an oracle for NP$^{\mathbf{D}}$ followed by a computation in co-NP$^{\mathbf{C}}$ (for the $\leq$-AR) or a computation in $\mathbf{C}$ (for the $\leq$-IAR and $\leq$-ICR). This is shown in a similar way as Theorem 4.

Theorem 6. *If BCQ answering from knowledge bases over a Datalog$^{\pm}$ language L is in $\mathbf{D}$ in the data complexity and in $\mathbf{C}$ in the fp-combined complexity, then $\leq$-AR (resp., $\leq$-IAR and $\leq$-ICR) BCQ answering from knowledge bases over L is possible by a computation in* P *with an oracle for* NP$^{\mathbf{D}}[O(\log n)]$*, followed by a computation in* co-NP$^{\mathbf{C}}$ *(resp., $\mathbf{C}$), in the fp-combined complexity.*

As a corollary of the theorems above, we obtain all the upper bounds for $\leq$-AR BCQ answering in Table 2, and all the upper bounds for $\leq$-IAR and $\leq$-ICR BCQ answering in Table 3. In particular, the Θ^{p}_2 membership of $\leq$-AR and $\leq$-IAR BCQ answering in the data complexity for the languages whose BCQ reasoning is feasible in polynomial time in the data complexity was already known (Bienvenu, Bourgaux, and Goasdoué 2014). The other results are new.

Corollary 7. *$\leq$-AR (resp., $\leq$-IAR and $\leq$-ICR) BCQ answering from knowledge bases over Datalog$^{\pm}$ languages L in Table 2 (resp., Table 3) in the data, fp-combined, ba-combined, and combined complexity belongs to the complexity classes shown in Table 2 (resp., Table 3).*

Hardness Results

The hardness results not explicitly proven follows from the hardness of classical reasoning over consistent ontologies.

We now show that $\leq$-AR, $\leq$-IAR, and $\leq$-ICR BCQ answering are Θ^{p}_2-hard in the data complexity for all the Datalog$^{\pm}$ languages considered.

Theorem 8. *For every $C \in \{AR, IAR, ICR\}$, $\leq$-C BCQ answering from knowledge bases over* $\mathsf{LF}_\bot$*,* $\mathsf{AF}_\bot$*, and* $\mathsf{SF}_\bot$ *(and hence also for* $\mathsf{L}_\bot$*,* $\mathsf{A}_\bot$*,* $\mathsf{S}_\bot$*,* $\mathsf{G}_\bot$*,* $\mathsf{GF}_\bot$*,* $\mathsf{F}_\bot$*,* $\mathsf{WS}_\bot$*, and* $\mathsf{WA}_\bot$*) is* Θ^{p}_2-hard *in the data complexity.*

Proof. We use a reduction from the Θ^{p}_2-complete problem InAllMaxIS (Lopatenko and Bertossi 2007; 2016): given a graph G and a vertex w, decide whether w belongs to all the independent sets of G of maximum size.

Let (G, w) be an instance of InAllMaxIS, where $G = (V, E)$ and $w \in V$, and $n = |V|$. From (G, w) we build the knowledge base $KB_{MaxIS} = (D_{MaxIS}, \Sigma_{MaxIS})$ and the query q_{MaxIS} as follows.

For each vertex $v \in V$, there is a fact $In(v)$ in D_{MaxIS}. For each edge $(v_1, v_2) \in E$, there are n facts $Edge(v_1, v_2, i)$ in D_{MaxIS}, with $1 \leq i \leq n$. D_{MaxIS} contains also the fact $distinguished(w)$. The only dependency in Σ_{MaxIS} is the NC $In(X) \wedge In(Y) \wedge Edge(X, Y, Z) \rightarrow \bot$. Finally, $q_{MaxIS} = \exists X(In(X) \wedge distinguished(X))$. Observe that Σ_{MaxIS} is vacuously linear, sticky, acyclic, and full.

We can show that (G, w) is a 'yes'-instance of InAllMaxIS iff KB_{MaxIS} entails q_{MaxIS} under any $\leq$-C, for $C \in \{AR, IAR, ICR\}$. Intuitively, repairs contain all facts $Edge(v1, v2, i)$, $distinguished(w)$ and a set of facts $In(v)$ corresponding to a maximum independent set of G. □

For $\leq$-AR BCQ answering, we show its Π^{p}_2-hard in the *fp*-combined complexity, for all the Datalog$^{\pm}$ languages considered. This follows from a reduction in (Lukasiewicz et al. 2015), which also applies in the case of maximum cardinality. Note that also a simplification of the reduction used here to prove Theorem 10 can be used to show this result.

Theorem 9. *$\leq$-AR BCQ answering from knowledge bases over $\mathsf{LF}_\bot$, $\mathsf{AF}_\bot$, and $\mathsf{SF}_\bot$ (and hence also for $\mathsf{L}_\bot$, $\mathsf{A}_\bot$, $\mathsf{S}_\bot$, $\mathsf{G}_\bot$, $\mathsf{GF}_\bot$, $\mathsf{F}_\bot$, $\mathsf{WS}_\bot$, and $\mathsf{WA}_\bot$) is* Π^{p}_2-hard *in the fp-combined complexity.*

We now show that, for any $C \in \{AR, IAR, ICR\}$, $\leq$-C BCQ answering is Θ^{p}_3-hard in the *ba*-combined complexity, already for the simplest languages here considered.

Theorem 10. *For every $C \in \{AR, IAR, ICR\}$, $\leq$-C BCQ answering from knowledge bases over $\mathsf{LF}_\bot$, $\mathsf{AF}_\bot$, and $\mathsf{SF}_\bot$ (and hence also for $\mathsf{L}_\bot$, $\mathsf{S}_\bot$, $\mathsf{GF}_\bot$, and $\mathsf{F}_\bot$) is* Θ^{p}_3-hard *in the ba-combined complexity.*

Proof sketch. We prove the statement via a reduction from the Θ^{p}_3-complete problem COMP-VALID$_2$, which is a generalization of the problem COMP-SAT (Lukasiewicz and Malizia 2016): given two sets A and B of quantified Boolean formulas characterized by 2 alternating quantifiers, decide whether the number of valid formulas in A is greater than the number of valid formulas in B. The Θ^{p}_3-hardness of COMP-VALID$_2$ holds even if the following restrictions are imposed over the instances (Lukasiewicz and Malizia 2017):

- $|A| = |B|$;
- all formulas of A and B are of the kind $(\forall X)(\exists Y)\phi(X, Y)$, where $\phi(X, Y)$ is a non-quantified 3CNF formula;
- all formulas of A and B have the same number of clauses in the non-quantified part;
- all formulas of A and B have the same sets of universally quantified variables and existentially quantified variables;
- sets $A = \{\Phi_1, \ldots, \Phi_v\}$ and $B = \{\Psi_1, \ldots, \Psi_v\}$ are such that Φ_{u+1} (resp., Ψ_{u+1}) being valid implies Φ_u (resp., Ψ_u) being valid as well, for any u (intuitively, all valid formulas have the lowest indices in sets A and B).

From the last assumption it follows that (A, B) is a 'yes'-instance of COMP-VALID$_2$ iff there is an index u such that $\Phi_u \in A$ is valid and $\Psi_u \in B$ is not valid.

Given the restrictions listed above, we can also assume, without simplifying the computational complexity of the problem, that all formulas of A and B are $NQBF_{2,\forall}$ ones (Greco et al. 2009; 2011; Schaefer 2001). $NQBF_{2,\forall}$ formulas are quantified Boolean formulas of the kind $\Phi = (\forall X)(\exists Y)\phi(X, Y)$, in which $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_r\}$ are two disjoint sets of Boolean variables, and $\phi(X, Y) = c_{i(1)} \wedge c_{\bar{i}(1)} \wedge \cdots \wedge c_{i(n)} \wedge c_{\bar{i}(n)} \wedge c_1 \wedge \cdots \wedge c_m$ is a 3CNF formula, where each universally quantified variable $x_k \in X$ occurs only in the two clauses $c_{i(k)} = (x_k \vee \neg y_k)$ and $c_{\bar{i}(k)} = (\neg x_k \vee y_k)$—intuitively, each variable x_k enforces the truth value of a corresponding variable y_k, which thus plays its role in the formula $\phi(X, Y)$. Indeed, a formula $(\forall X)(\exists Y)c_1 \wedge \cdots \wedge c_m$ can be replaced by $(\forall X)(\exists Y_X)(\exists Y)c_{i(1)} \wedge c_{\bar{i}(1)} \wedge \cdots \wedge c_{i(n)} \wedge c_{\bar{i}(n)} \wedge c'_1 \wedge \cdots \wedge c'_m$, where $X = \{x_1, \ldots, x_n\}, Y_X = \{y^x_1, \ldots, y^x_n\}$ and each c'_i is obtained by replacing every $x_j \in X$ by $y^x_j \in Y_X$ in c_i. We will refer to the clauses $c_1, \ldots, c_m$ of an $NQBF_{2,\forall}$ formula as the *main* clauses of the formula.

Let $\mathcal{I} = (A, B)$ be an instance of COMP-VALID$_2$ satisfying all the restrictions above. We denote by X and Y the sets of the universally and existentially quantified variables, respectively. From $\mathcal{I}$, we build the knowledge base $KB_{CV}(\mathcal{I}) = (D_{CV}, \Sigma_{CV})$ and query $q_{CV}(\mathcal{I})$ as follows. In what follows, the first two arguments of each fact of D_{CV} specify the identifier of the set (i.e., whether the formula belongs to set A or B) and the numeric identifier of the formula in the set, respectively. In this way, we can discriminate facts referring to the different formulas in the two sets.

For each variable $x_i \in X$, in D_{CV} there are facts:

$$Val(s, u, x_i, t, k) \qquad Val(s, u, x_i, f, k)$$

where $s \in \{a, b\}$ is a constant representing sets A and B, respectively, $u \in \{1, \ldots, v\}$ is a numeric constant representing the index of the formula in the sets, x_i is a constant representing the respective variable, t and f are constants representing Boolean values *true* and *false*, respectively, and $k \in \{1, 2, 3\}$ is a numeric constant (allowing us to have three copies of the facts).

There are facts D_{CV} that will be used to impose the consistency of the assignments to the literals:

$$SimLit(s, u, t, t, k) \qquad OppLit(s, u, t, f, k)$$
$$SimLit(s, u, f, f, k) \qquad OppLit(s, u, f, t, k),$$

where $s \in \{a, b\}$, $u \in \{1, \ldots, v\}$, t, f, and $k \in \{1, 2, 3\}$, are constants with the same meaning as above.

There are facts in D_{CV} that will be used to select possible ways of satisfying clauses in the formulas:

$$ClSat(s, u, f, t, t, k) \qquad ClSat(s, u, t, t, t, k),$$
$$ClSat(s, u, f, t, f, k) \qquad ClSat(s, u, t, t, f, k)$$
$$ClSat(s, u, f, f, t, k) \qquad ClSat(s, u, t, f, t, k)$$
$$ClSat(s, u, t, f, f, k)$$

where $s \in \{a, b\}$, $u \in \{1, \ldots, v\}$, t, f, and $k \in \{1, 2, 3\}$, are constants with the same meaning as above.

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

Finally, D_{CV} also contains facts that will be used to signal the validity or the non-validity of the formulas:

$$NonValid(s, u, k) \qquad Valid(s, u),$$

where $s \in \{a, b\}$, $u \in \{1, \ldots, v\}$, and $k \in \{1, 2\}$, are constants with the same meaning as above. Note that we have two copies for facts *NonValid*, and one copy for facts *Valid*.

In the program Σ_{CV}, there is no TGD, and there are the following NCs. For notational convenience, in the NCs below, we will use the underscore '_' as a placeholder for a fresh variable not appearing anywhere else in the NC.

The first negative constraint is

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

Next, for each formula in sets A and B, there is an NC. This NC is used to check the satisfiability of a formula once an assignment for the variables in X has been provided.

For simplicity in the exposition, let us assume that we are considering the formula $\Phi_1 = (\forall X)(\exists Y)\phi_1(X,Y)$ belonging to set A. The following construction for the NC can be generalized to any formula of A or B. Since this NC is intricate, we look at its various constituent pieces.

A first piece of the NC checks that *at least one* of the copies of the structural facts is in the selection:

$$\mathit{Config}(a,1) \equiv \bigwedge_{p(a,1,\mathbf{c},\_)\in D^{St}_{CV}} p(a,1,\mathbf{c},\_).$$

By using a fresh variable as the last argument of all the predicates in Config, the presence of just one copy of each structural fact is enough to support the activation of Config.

A second piece of the NC "reads" the assignment on the variables in X encoded in the selection of the database:

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

Below we use the following notation: $l_{j,k}$ is the k^{th} literal in the j^{th} main clause, c_j, and $v_{j,k}$ is the variable of $l_{j,k}$.

A third piece of the NC aims at "copying" the assignment on the variables in X onto the associated variables in Y:

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

where $T_{j,k}$ is a variable for the Boolean value of the literal $l_{j,k}$ for which $v_{j,k} = y_i$ in $\phi_1(X,Y)$. Observe that, in order for Copy to work properly, each variable y_i must appear as a positive literal in one of the main clauses of $\phi_1(X,Y)$ at least once. This can be assumed without loss of generality, because if y_i always appears as a negative literal in all the main clauses of $\phi_1(X,Y)$, then we can replace all the occurrences of the negative literal $\neg y_i$ with the positive literal y_i without altering the satisfiability properties of $\phi_1(X,Y)$.

A fourth piece of the NC forces that the facts selected to simulate the assignment on the variables in Y are consistent. In the notation below, $\ell_{j,k} \sim \ell_{j',k'}$ means that literals $\ell_{j,k}$ and $\ell_{j',k'}$ are both positive or negative, while $\ell_{j,k} \not\sim \ell_{j',k'}$ means that one literal is positive and the other is negative.

$$\mathit{ConsistY}(a,1) \equiv \bigwedge_{\substack{\forall(\ell_{j,k},\ell_{j',k'})\\ \text{s.t. } v_{j,k}=v_{j',k'}\wedge\\ \ell_{j,k}\sim\ell_{j',k'}}} \mathit{SimLit}(a,1,T_{j,k},T_{j',k'},\_) \bigwedge_{\substack{\forall(\ell_{j,k},\ell_{j',k'})\\ \text{s.t. } v_{j,k}=v_{j',k'}\wedge\\ \ell_{j,k}\not\sim\ell_{j',k'}}} \mathit{OppLit}(a,1,T_{j,k},T_{j',k'},\_),$$

where $T_{j,k}$ is a variable with the same meaning as above.

The last piece of the NC checks $\phi_1(X,Y)$'s satisfiability:

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

these predicates are only for the main clauses of $\phi_1(X,Y)$.

To conclude, the NC associated with $\Phi_1 \in A$ is:

$$\mathit{Config}(a,1), \mathit{AssignX}(a,1), \mathit{Copy}(a,1), \mathit{ConsistY}(a,1), \mathit{Satisfied}(a,1), \mathit{NonValid}(a,1,\_) \rightarrow \bot.$$

The last NC is: $\mathit{NonValid}(S,U,\_)$, $\mathit{Valid}(S,U) \rightarrow \bot$.

The query is:

$$q_{CV}(\mathcal{I}) = (\exists U)(\,\mathit{Valid}(a,U), \mathit{NonValid}(b,U,\_)).$$

$KB_{CV}(\mathcal{I})$ has no TGDs, and bounded arity predicates.

It can be shown that $\mathcal{I}$ is a 'yes'-instance of COMP-VALID$_2$ iff $KB_{CV}(I)$ entails $q_{CV}(I)$ under $\leq$-C semantics, for any $C \in \{AR, IAR, ICR\}$. □

The following result shows that $\leq$-AR-BCQ $\leq$-IAR-BCQ, and $\leq$-ICR-BCQ answering for $\mathsf{A}_\perp$ are P$^{\text{NEXP}}$-hard in the ba-combined complexity, proving all P$^{\text{NEXP}}$-hardness results in Tables 2 and 3, including those for the more general combined complexity.

Theorem 11. *For any $C \in \{AR, IAR, ICR\}$, $\leq$-C BCQ answering for $\mathsf{A}_\perp$ are* P$^{\text{NEXP}}$*-hard in the ba-combined and combined complexity.*

Proof sketch. Intuitively, the reduction for the P$^{\text{NEXP}}$-hardness proof in (Eiter, Lukasiewicz, and Predoiu 2016) for $\subseteq$-AR-BCQ answering for $\mathsf{A}_\perp$ in the ba-combined complexity is already a reduction for $\leq$-AR-BCQ answering, as maximal-cardinality consistent database subsets there coincide with maximal-subset consistent database subsets. Furthermore, since the reduction uses a ground atomic query, this also shows the P$^{\text{NEXP}}$-hardness of $\leq$-ICR-BCQ answering for $\mathsf{A}_\perp$, as $\leq$-ICR-BCQ answering coincides with $\leq$-AR-BCQ answering for ground queries. Finally, the reduction for the P$^{\text{NEXP}}$-hardness proof in (Eiter, Lukasiewicz, and Predoiu 2016) for $\subseteq$-AR-BCQ answering for $\mathsf{A}_\perp$ in the ba-combined complexity is turned into a P$^{\text{NEXP}}$-hardness proof for $\leq$-IAR-BCQ answering in this case. There, one encodes initial tiling assignments $v_1(X_i), \ldots, v_n(X_n)$ and has a ground atomic query q, which we now also include in the database along with a fresh ground atom nq and the NC $v_1(X_i) \wedge \ldots \wedge v_n(X_n) \wedge q \wedge nq \rightarrow \bot$. This intuitively "forces" the atom q into the database. □

The Θ^{p}_2-hardness in the *fp*-combined complexity of $\leq$-IAR-BCQ and $\leq$-ICR-BCQ answering over all the languages considered follows from the Θ^{p}_2-hardness in the data complexity of $\leq$-IAR-BCQ answering.

Theorem 12. *$\leq$-IAR and $\leq$-ICR BCQ answering is Θ^{p}_2-hard in the fp-combined complexity for $\mathsf{L}_\perp$, $\mathsf{LF}_\perp$, $\mathsf{A}_\perp$, $\mathsf{AF}_\perp$, $\mathsf{S}_\perp$, $\mathsf{SF}_\perp$, $\mathsf{G}_\perp$, $\mathsf{GF}_\perp$, $\mathsf{F}_\perp$, $\mathsf{WS}_\perp$, and $\mathsf{WA}_\perp$ knowledge bases.*

Summary and Outlook

We have given a precise picture of the complexity of BCQ answering under different cardinality-maximal inconsistency-tolerant semantics, namely, the ABox repair, the intersection of repairs (IAR), and the intersection of closed repairs (ICR) semantics, for the most popular Datalog$^{\pm}$ languages and complexity measures. Note that these complexity results can now also be used to derive further complexity

results for other Datalog$^{\pm}$ languages. For example, for *shy* Datalog$^{\pm}$ (Leone et al. 2012), by the results of this paper, it is immediate that BCQ answering is complete for Θ_2^{p} and EXP in the data and combined complexity, respectively.

Future research lines include considering other classes of existential rules and defining other semantics for inconsistency-tolerant ontological query answering. In particular, it would be interesting to explore whether there are data-tractable and/or even first-order rewritable other such semantics. Furthermore, a more fine-grained way to analyze the complexity of query answering would be a non-uniform approach, looking at the complexity of a single ontology or a single ontology-mediated query (see, e.g., (Bienvenu et al. 2014; Koutris and Suciu 2014; Hernich et al. 2017)).

Acknowledgments. This work was supported by the Alan Turing Institute under the UK EPSRC grant EP/N510129/1, and by 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.

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

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. 2016. Explaining inconsistency-tolerant query answering over description logic knowledge bases. In *Proc. AAAI*, 900–906.

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.

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.

Greco, G.; Malizia, E.; Palopoli, L.; and Scarcello, F. 2009. On the complexity of compact coalitional games. In *Proc. IJCAI*, 147–152.

Greco, G.; Malizia, E.; Palopoli, L.; and Scarcello, F. 2011. On the complexity of core, kernel, and bargaining set. *Artif. Intell.* 175(12–13):1877–1910.

Hernich, A.; Lutz, C.; Papacchini, F.; and Wolter, F. 2017. Dichotomies in ontology-mediated querying with the guarded fragment. In *Proc. PODS*, 185–199.

Koutris, P., and Suciu, D. 2014. A dichotomy on the complexity of consistent query answering for atoms with simple keys. In *Proc. ICDT*, 165–176.

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.

Leone, N.; Manna, M.; Terracina, G.; and Veltri, P. 2012. Efficiently computable Datalog$^{\exists}$ programs. In *Proc. KR*.

Lopatenko, A., and Bertossi, L. E. 2007. Complexity of consistent query answering in databases under cardinality-based and incremental repair semantics. In *Proc. ICDT*, 179–193.

Lopatenko, A., and Bertossi, L. 2016. Complexity of consistent query answering in databases under cardinality-based and incremental repair semantics (extended version). Technical Report 1605.07159, arXiv.

Lukasiewicz, T., and Malizia, E. 2016. On the complexity of mCP-nets. In *Proc. AAAI*, 558–564.

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. 2018a. Complexity of approximate query answering under inconsistency in Datalog$^{\pm}$. In *Proc. IJCAI*, 1921–1927.

Lukasiewicz, T.; Malizia, E.; and Molinaro, C. 2018b. Complexity of approximate query answering under inconsistency in Datalog$^{\pm}$. In *Proc. SEBD*, 22:1–8.

Lukasiewicz, T.; Martinez, M. V.; and Simari, G. I. 2012. Inconsistency handling in Datalog+/– ontologies. In *Proc. ECAI*, 558–563.

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

Rosati, R. 2011. On the complexity of dealing with inconsistency in description logic ontologies. In *Proc. IJCAI*.

Schaefer, M. 2001. Graph Ramsey theory and the polynomial hierarchy. *J. Comput. Syst. Sci.* 62(2):290–322.

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