Proceedings of the 20th International Conference on Principles of Knowledge Representation and Reasoning
Main Track

Complexity of Inconsistency-Tolerant Query Answering in Datalog+/– Under Preferred Repairs

Thomas Lukasiewicz[1,2] , **Enrico Malizia**[3] , **Cristian Molinaro**[4]
[1]Institute of Logic and Computation, Vienna University of Technology, Austria
[2]Department of Computer Science, University of Oxford, UK
[3]DISI, University of Bologna, Italy
[4]DIMES, University of Calabria, Italy
thomas.lukasiewicz@tuwien.ac.at, enrico.malizia@unibo.it, cmolinaro@dimes.unical.it

Abstract

Inconsistency-tolerant semantics have been proposed to provide meaningful ontological query answers even in the presence of inconsistencies. Several such semantics rely on the notion of a *repair*, which is a "maximal" consistent subset of the database, where different maximality criteria might be adopted depending on the application at hand. Previous work in the context of Datalog$^\pm$ has considered only the subset and cardinality maximality criteria. We take here a step further and study inconsistency-tolerant semantics under maximality criteria based on weights and priority levels. We provide a thorough complexity analysis for a wide range of existential rule languages and for several complexity measures.

1 Introduction

In real-world applications, data possibly coming from different sources may exhibit inconsistency. Obtaining meaningful ontological query answers in these scenarios requires inconsistency-tolerant semantics. Popular ones are the *ABox repair (AR)*, first defined for relational databases (Arenas, Bertossi, and Chomicki 1999) and then generalized for description logics (DLs) (Lembo et al. 2010), the *intersection of repairs (IAR)* (Lembo et al. 2010), and the *intersection of closed repairs (ICR)* (Bienvenu 2012).

All the aforementioned semantics, as well as others (see, e.g., (Lembo et al. 2010)), are based on the notion of a *repair*, which is a "maximal" consistent subset of the knowledge base's facts. Subset maximality was adopted upon introduction of all the above semantics. However, other maximality criteria are relevant in practice. For instance, maximum cardinality is a stronger criterion ruling out subset-maximal repairs not containing the highest number of facts, which is suitable for settings where all database facts are considered equally reliable. When some facts are considered more reliable than others, both the criteria above can be refined by *priorities*, where the database is partitioned into groups of different priority levels. Then, the maximality of consistent subsets of facts is checked per priority level via the subset and cardinality maximality criteria. Further, when database facts are associated with weights (e.g., quantitatively measuring their reliability), a natural criterion is to select maximum-weight consistent subsets of the database. All the criteria above have been proven apt in many contexts, including ranked knowledge bases (Brewka 1989; Benferhat et al. 1993), explaining query answering (Ceylan et al. 2021), abduction reasoning (Eiter and Gottlob 1995), preferred subtheories for default reasoning (Brewka 1991), and prioritized circumscription (Lifschitz 1985).

With different possible criteria to define repairs, one relevant issue is to understand how the choice of a criterion affects the complexity of common reasoning tasks. While the aforementioned criteria have been studied in the context of querying inconsistent DL knowledge bases (Bienvenu, Bourgaux, and Goasdoué 2014a), they have received little attention under existential rule languages. Indeed, in the latter setting, the complexity of inconsistency-tolerant query answering has been studied only for subset-maximal (Lukasiewicz et al. 2022) and cardinality-maximal repairs (Lukasiewicz, Malizia, and Vaicenavičius 2019).

In this paper, we close this gap and study the complexity of the AR, IAR, and ICR semantics for maximality criteria based on weights and priority levels. We also study the complexity of another common reasoning task in inconsistency handling, namely, repair checking—that is, deciding whether a database is a repair (w.r.t. a maximality criterion). Besides analyzing the complexity of repair checking for the criteria based on weights and priority levels, we also consider the subset and cardinality criteria, for which this problem has not been thoroughly investigated yet. We provide a thorough complexity analysis for a wide range of existential rule languages and for several complexity measures.

2 Preliminaries

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, a null, or a 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 containing only 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 its variables are existentially quantified; for BCQs, the only possible answer is the empty tuple. A BCQ 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 $head(\sigma)$. We consider single-atom-head TGDs; however, our results extend to TGDs with a conjunction of atoms in the head. An instance I satisfies a TGD 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 an NC ν, written $I \models \nu$, if there is *no* homomorphism h such that $h(\varphi(\mathbf{X})) \subseteq I$. We will use q_ν to denote the BCQ $\exists \mathbf{X}\, \varphi(\mathbf{X})$. 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. For a class $\mathbb{C}$ of TGDs, $\mathbb{C}_\bot$ denotes the combination of $\mathbb{C}$ with arbitrary NCs. NC denotes the language using only NCs. Finite sets of TGDs and NCs are called *programs*, and TGDs are also called *existential rules*.

The Datalog$^\pm$ languages here considered guaranteeing 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), the "weak" generalizations weakly sticky (WS) (Calì, Gottlob, and Pieris 2012) and weakly acyclic TGDs (WA) (Fagin et al. 2005), their "full" (i.e., existential-free) restrictions linear full (LF), guarded full (GF), sticky full (SF), and acyclic full TGDs (AF), respectively, and full TGDs (F) in general. We refer to (Calautti et al. 2022; Lukasiewicz et al. 2022) for a more detailed overview.

Knowledge Bases A *knowledge base* is a pair (D, Σ), where D is a database and Σ is a program. For a program Σ, Σ_T and Σ_{NC} denote 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 $ans(q, KB) \neq \emptyset$. Another way to define ontological query answering is via the concept of the *Chase* (see, e.g., Calì, Gottlob, and Kifer 2013; Tsamoura et al. 2021). The decision version of *CQ answering* is: given a knowledge base KB, a CQ q, and a tuple $\mathbf{t}$ of constants, decide whether $\mathbf{t} \in ans(q, KB)$. Since CQ answering can be reduced in LOGSPACE to BCQ answering, we focus on BCQs. We denote by BCQ($\mathcal{L}$) the problem of BCQ answering when restricted over programs belonging to $\mathcal{L}$.

Following Vardi (1982), the *combined complexity* of BCQ answering considers the database, the program, 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 constant. The *fixed-program-combined* (or *fp-combined*) *complexity* considers the program fixed; in the *data complexity* the query is fixed as well. Table 1 recalls complexity results of BCQ answering for the languages in this paper (Lukasiewicz et al. 2022).

In the *repair checking* problem, i.e., deciding if a database is a repair of a knowledge base, the data and *fp*-combined complexity coincide, as there is no query in the input.

Computational Complexity AC^0 is the class of problems that can be decided by uniform families of Boolean circuits of polynomial size and constant depth. PSPACE (resp., P, EXP, 2EXP) is the class of problems decidable in deterministic polynomial space (resp., polynomial time, exponential time, double exponential time). NP and NEXP are the classes of problems decidable in nondeterministic polynomial and exponential time, respectively; co-NP and co-NEXP are their complement. $\text{D}^\text{P} = \text{NP} \wedge \text{co-NP}$ (resp., $\text{D}^\text{EXP} = \text{NEXP} \wedge \text{co-NEXP}$) is the class of problems that are the conjunction of a problem in NP (resp., NEXP) and a problem in co-NP (resp., co-NEXP). Σ_2^P is the class of problems decidable in nondeterministic polynomial time with an NP oracle, and Π_2^P is the complement of Σ_2^P. Θ_2^P is the class of problems decidable in deterministic polynomial time with logarithmically-many calls (or, equivalently, a constant number of rounds of polynomially-many parallel calls) to an NP oracle. Δ_2^P (resp., Δ_3^P) is the class of problems decidable in deterministic polynomial time with an NP (resp., Σ_2^P) oracle. P^NEXP is the class of problems that are decidable in deterministic polynomial time with a NEXP oracle. The above complexity classes and their inclusion relationships are: $\text{AC}^0 \subseteq \text{P} \subseteq \text{NP}, \text{co-NP} \subseteq \text{D}^\text{P} \subseteq \Theta_2^\text{P} \subseteq \Delta_2^\text{P} \subseteq \Sigma_2^\text{P}, \Pi_2^\text{P} \subseteq \Delta_3^\text{P} \subseteq \text{PSPACE} \subseteq \text{EXP} \subseteq \text{NEXP}, \text{co-NEXP} \subseteq \text{D}^\text{EXP} \subseteq \text{P}^\text{NEXP} \subseteq 2\text{EXP}$.

	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. 2022). All non-"in" entries are completeness results.

3 Inconsistency-Tolerant Semantics Under Preferred Repairs

We recall the AR, IAR, and ICR semantics, defined w.r.t. an arbitrary notion of preferred repair. We then introduce different maximality criteria for preferred repairs.

Given a knowledge base $KB = (D, \Sigma)$, a *selection* of KB is a database D' such that $D' \subseteq D$. A selection D' of KB is *consistent* iff (D', Σ) is consistent. Symmetrically, the concept of consistent selection is linked to that of culprit. Intuitively, a culprit is a subset of D that, together with Σ_T, entails some NC; more formally, a *culprit* is a subset C of D s.t. $(C, \Sigma_T) \models q_\nu$ for some $\nu \in \Sigma_{NC}$. A culprit for an NC ν is an "explanation" (Ceylan et al. 2019) of q_ν. By deleting from D a hitting set (Chomicki and Marcinkowski 2005; Gottlob and Malizia 2014; 2018) of facts S intersecting all culprits, we obtain a consistent selection $D' = D \setminus S$. The consistent selections of a knowledge base can be ordered according to some criteria to select the maximal (or "preferred") ones. Given a preorder $\preccurlyeq$ over a set $\mathcal{S}$ of databases (i.e., $\preccurlyeq$ is a reflexive and transitive binary relation on $\mathcal{S}$), for two elements D' and $D'' \in \mathcal{S}$, we write $D' \prec D''$ to denote 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.

$Rep_{\preccurlyeq}(KB)$ denotes the set of all $\preccurlyeq$-repairs of KB.

For a knowledge base $KB = (D, \Sigma)$, the *closure* of KB, denoted $Cl(KB)$, is the set of all facts 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 a preorder over the consistent selections of KB.

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

Two common tasks in inconsistency handling are *repair checking* (i.e., deciding whether a database is a $\preccurlyeq$-repair) and query entailment under inconsistency-tolerant semantics, which in our case are the $\preccurlyeq$-AR/IAR/ICR semantics.

Problem: $\preccurlyeq$-$RC(\mathcal{L})$.
Input: A knowledge base (D, Σ) with $\Sigma \in \mathcal{L}$, and a database D'.
Question: Is D' a $\preccurlyeq$-repair of (D, Σ)?

Problem: $\preccurlyeq$-$S(\mathcal{L})$, with $S \in \{AR, IAR, ICR\}$.
Input: A knowledge base (D, Σ) with $\Sigma \in \mathcal{L}$, and a BCQ q.
Question: Does $(D, \Sigma) \models_{\preccurlyeq\text{-}S} q$ hold?

Besides $\subseteq$ and $\leq$, we also consider the preorders introduced below. In the following, (D, Σ) is a knowledge base.

Weights ($\leq_w$) The database D comes along with a weight function $w\colon D \to \mathbb{N}$ assigning weights to its facts. For every $D' \subseteq D$, w assigns a weight to D' defined as $w(D') = \sum_{f \in D'} w(f)$ (with a slight abuse of notation, w applies to both facts and sets of facts). Also, w induces a preorder over the subsets of D as follows: for every $D_1, D_2 \subseteq D$, we write $D_1 \leq_w D_2$ iff $w(D_1) \leq w(D_2)$. We assume that weights are represented in binary (this plays a role for establishing upper bounds—see, e.g., the proof of Theorem 11).

The following two preorders are based on a *prioritization* $P = (\mathcal{P}_1, \dots, \mathcal{P}_n)$ of the database D, that is, P is a partition of D into the priority levels $\mathcal{P}_i$, where $\mathcal{P}_1$ contains the most reliable facts, and $\mathcal{P}_n$ contains the least reliable facts of D.

Prioritized Cardinality ($\leq_P$) For every $D_1, D_2 \subseteq D$, we write $D_1 \leq_P D_2$ iff for every $1 \leq i \leq n$, $|D_1 \cap \mathcal{P}_i| = |D_2 \cap \mathcal{P}_i|$, or there is some $1 \leq i \leq n$ such that $|D_1 \cap \mathcal{P}_i| < |D_2 \cap \mathcal{P}_i|$ and for every $1 \leq j < i$, $|D_1 \cap \mathcal{P}_j| = |D_2 \cap \mathcal{P}_j|$.

Prioritized Set Inclusion ($\subseteq_P$) For every $D_1, D_2 \subseteq D$, we write $D_1 \subseteq_P D_2$ iff for every $1 \leq i \leq n$, $D_1 \cap \mathcal{P}_i = D_2 \cap \mathcal{P}_i$, or there is some $1 \leq i \leq n$ such that $D_1 \cap \mathcal{P}_i \subsetneq D_2 \cap \mathcal{P}_i$ and for every $1 \leq j < i$, $D_1 \cap \mathcal{P}_j = D_2 \cap \mathcal{P}_j$.

Note that $\leq_w$ generalizes $\leq_P$, see, e.g., (Bienvenu, Bourgaux, and Goasdoué 2014a; Eiter and Gottlob 1995). Also, $\leq_P$ (resp., $\subseteq_P$) generalizes $\leq$ (resp., $\subseteq$), as the latter can be captured by the former with the prioritization $P = (D)$.

4 Overview of Complexity Results

Our complexity results are summarized in Tables 2 to 7. In particular, Tables 2 and 3 cover repair checking, while Tables 4 to 7 cover inconsistency-tolerant query entailment.

As for repair checking, our results show that we can partition the considered maximality criteria into two classes $\{\leq, \leq_P, \leq_w\}$ and $\{\subseteq, \subseteq_P\}$, with criteria in the same class having the same complexity. Thus, moving from $\leq$ to the more general criteria $\leq_P$ and $\leq_w$ does not incur an increase of complexity (this holds also when moving from $\leq_P$ to $\leq_w$). Likewise, the complexity does not increase when moving from $\subseteq$ to $\subseteq_P$. Comparing the two classes of criteria, $\leq_w$, $\leq_P$, and $\leq$ are always at least as expensive as $\subseteq_P$ and $\subseteq$, with higher complexity in most of the cases.

We now discuss the complexity results for inconsistency-tolerant query entailment. Our results show that $\leq_w$ and $\leq_P$ exhibit the same complexity, which is always at least as high as the one of $\subseteq_P$. The IAR and ICR semantics have the same complexity across all maximality criteria, which is a behavior shown by $\leq$ as well (Lukasiewicz, Malizia, and Vaicenavičius 2019), while this does not hold for $\subseteq$ (Lukasiewicz et al. 2022). As usual, the IAR and ICR semantics are at most as expensive as the AR semantics.

Another interesting comparison to make is between $\subseteq$ (resp., $\leq$) and $\subseteq_P$ (resp., $\leq_w, \leq_P$). The complexity results for $\subseteq$ and $\leq$ can be found in (Lukasiewicz et al. 2022) and (Lukasiewicz, Malizia, and Vaicenavičius 2019), respectively. When we move from $\subseteq$ to the more general $\subseteq_P$ criterion, the complexity does not increase for the AR and ICR semantics. In contrast, the complexity increases for the IAR semantics, but only for the FO-rewritable languages ($\mathsf{L}_\perp$, $\mathsf{S}_\perp$, $\mathsf{A}_\perp$, and their sublanguages) in the data and

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

Table 2: Complexity of $\preccurlyeq$-$RC(\mathcal{L})$ for $\preccurlyeq\ \in \{\leq_w, \leq_P, \leq\}$. All entries are completeness results.

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

Table 3: Complexity of $\preccurlyeq$-$RC(\mathcal{L})$ for $\preccurlyeq\ \in \{\subseteq_P, \subseteq\}$. All non-"in" entries are completeness results.

fp-combined complexity. In particular, the complexity goes from membership in AC^0 to co-NP-completeness in the data complexity, and from NP-completeness to Θ_2^{P}-completeness in the *fp*-combined complexity. When we move from $\leq$ to the more general $\leq_w$ and $\leq_P$ criteria, the complexity of all inconsistency-tolerant semantics increases in several cases.

5 Repair Checking

We first discuss membership, and then hardness results. In the theorem statements, $\mathcal{L}$ *is any language of this paper*.

5.1 Membership Results

Membership results rely on the fact that testing that a knowledge base (D, Σ) is consistent can be accomplished by checking that there is no $\nu \in \Sigma_{NC}$ such that $(D, \Sigma_T) \models q_\nu$.

The following two theorems provide upper bounds for $\leq_w$-$RC(\mathcal{L})$ and $\subseteq_P$-$RC(\mathcal{L})$, respectively, via general procedures that apply to all languages and complexity measures that we consider. The resulting upper bounds are always tight, except for $\leq_w$-$RC(\mathsf{A}_\bot)$ in the ba-combined and combined complexity—these cases will be dealt with separately.

Theorem 3. *If BCQ($\mathcal{L}$) is in the complexity class* $\mathbf{C}$ *in the data / ba-combined / combined complexity, then $\leq_w$-$RC(\mathcal{L})$ is in* co-($\mathrm{NP}^{\mathbf{C}}$) *in the data / ba-combined / combined complexity.*

Proof. Let (D, Σ) be a knowledge base with $\Sigma \in \mathcal{L}$, let w be D's weight function, and let D' be a database. To decide whether D' is *not* a $\leq_w$-repair of (D, Σ), we guess a subset D'' of D and check that (1) (D', Σ) is inconsistent, or (2) (D'', Σ) is consistent and $w(D') < w(D'')$. Both conditions can be verified in polynomial time with an oracle in $\mathbf{C}$. $\square$

Theorem 4. *If BCQ($\mathcal{L}$) is in the complexity class* $\mathbf{C}$ *in the data / ba-combined / combined complexity, then $\subseteq_P$-$RC(\mathcal{L})$ can be decided with a polynomial number of* $\mathbf{C}$ *checks and a linear number of* co-$\mathbf{C}$ *checks in the data / ba-combined / combined complexity.*

Proof. Let $KB = (D, \Sigma)$ be a knowledge base with $\Sigma \in \mathcal{L}$, let $P = (\mathcal{P}_1, \dots, \mathcal{P}_n)$ be a prioritization, and let D' be a database. To decide whether D' is a $\subseteq_P$-repair of KB, we check that (1) (D', Σ) is consistent and (2) for every $1 \leq i \leq n$ and fact f in $\mathcal{P}_i \setminus D'$, $((D' \cap (\mathcal{P}_1 \cup \cdots \cup \mathcal{P}_i)) \cup \{f\}, \Sigma)$ is inconsistent. Condition (1) can be verified with a linear number of co-$\mathbf{C}$ checks. Condition (2) can be verified with a polynomial number of $\mathbf{C}$ checks. $\square$

To obtain the D^{P} (resp., $\mathrm{D}^{\mathrm{EXP}}$) membership results in Table 3, observe in the previous theorem that multiple NP, co-NP, NEXP, and co-NEXP checks can be carried out with a single NP, co-NP, NEXP, and co-NEXP check, respectively.

The upper bound for $\leq_w$-$RC(\mathsf{A}_\bot)$ in the ba-combined and combined complexity requires a dedicated analysis.

Theorem 5. $\leq_w$-$RC(\mathsf{A}_\bot)$ *is in* $\mathrm{D}^{\mathrm{EXP}}$ *in the ba-combined and combined complexity.*

Proof. Let (D, Σ) be a knowledge base with $\Sigma \in \mathcal{L}$, let w be the weight function for D, and let D' be a database. First, notice that BCQ($\mathsf{A}_\bot$) is in NEXP. We need to check that (1) (D', Σ) is consistent and (2) there is no $D'' \subseteq D$ such that (D'', Σ) is consistent and $w(D') < w(D'')$. Condition (1) can be verified in co-NEXP: we can apply the same argument used for Condition (1) in the proof of Theorem 4 along with the observation reported after the theorem. As for Condition (2), we need to check that all the (exponentially many) subsets D'' of D are such that (D'', Σ) is inconsistent or $w(D') \geq w(D'')$. Notice that checking inconsistency of a single (D'', Σ) is in NEXP. Thus, we can guess exponentially many witnesses for the inconsistency of every single D''—notice that the overall size of the guess remains exponential. Then, we can go over each D'' and check that $w(D') \geq w(D'')$ holds or D'' is inconsistent (by verifying its witness, which takes exponential time, since checking inconsistency of a single D'' is in NEXP). The latter checking procedure goes through an exponential number of databases D'' and requires exponential time on each of them, hence remaining exponential overall. $\square$

The upper bounds in Table 2 (resp., Table 3) for $\leq_P$ and $\leq$ (resp., $\subseteq$) follow from those of $\leq_w$ (resp., $\subseteq_P$) discussed so far, since the latter generalizes the former.

5.2 Hardness Results

The following theorem provides tight lower bounds for $\leq$ for all languages in the data complexity.

Theorem 6. *(Lopatenko and Bertossi 2016)* $\leq$-$RC(\mathsf{NC})$ *is* co-NP-hard *in the data complexity.*

The theorem below provides lower bounds for $\leq$-$RC(\mathcal{L})$ and $\subseteq$-$RC(\mathcal{L})$ for all languages in the ba-combined and combined complexity. Such lower bounds are not always tight; we will devise tailored reductions for different cases.

Theorem 7. *The complement of BCQ($\mathcal{L}$) is reducible in polynomial time to $\leq$-$RC(\mathcal{L})$ and $\subseteq$-$RC(\mathcal{L})$ in the ba-combined and combined complexity.*

Proof. From a knowledge base (D, Σ) with $\Sigma \in \mathcal{L}$, and a BCQ $q = \exists \mathbf{Y} \phi(\mathbf{Y})$, we derive an instance of $\leq$-$RC(\mathcal{L})$ (resp., $\subseteq$-$RC(\mathcal{L})$) as follows: the knowledge base is (D, Σ'), where $\Sigma' = \Sigma \cup \{\phi(\mathbf{Y}) \rightarrow \bot\}$, and the candidate $\leq$- (resp., $\subseteq$-) repair is D. Notice that $\Sigma' \in \mathcal{L}$ and the reduction takes polynomial time. It can be easily verified that $(D, \Sigma) \not\models q$ iff D is a $\leq$- (resp., $\subseteq$-) repair of (D, Σ'). $\square$

The theorem above does not provide tight lower bounds for both $\leq$ and $\subseteq$ in the following cases: $\mathsf{A}_\bot$ in the *ba*-combined and combined complexity, as well as $\mathsf{LF}_\bot$, $\mathsf{AF}_\bot$, and $\mathsf{SF}_\bot$ in the *ba*-combined complexity. We provide specific hardness results for such cases in the following.

Theorem 8. $\leq$-$RC(\mathsf{NC})$ *is* Π_2^P-hard *in the ba-combined complexity.*

Proof. We provide a reduction from the following Π_2^P-complete problem: decide the validity of a quantified Boolean formula $\forall X \exists Y \phi(X, Y)$, where ϕ is in 3CNF. We will use $\ell_{j,k}$ to refer to the k^{th} literal of the j^{th} clause of $\phi(X, Y)$.

Below we build a knowledge base (D, Σ), with $\Sigma \in \mathsf{NC}$, and a database D', where all predicates have bounded arity.

The database. For each variable $x_i \in X$, the following facts are included into D:

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

where x_i is a constant representing the respective variable in X, and f and t are constants representing the Boolean values false and true, respectively.

Furthermore, the following facts are included in D to impose the consistency of the truth assignments to the literals:

$$\begin{array}{ll} SimLit(f, f), & OppLit(f, t), \\ SimLit(t, t), & OppLit(t, f), \end{array}$$

where f and t are constants with the same meaning as above. The predicate $SimLit(\cdot, \cdot)$ is used to impose that when a variable appears twice as a positive or a negative literal in two different places in the formula, the two literals must have the same truth value. On the other hand, the predicate $OppLit(\cdot, \cdot)$ is 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, the two literals must have different truth value.

The following facts are included in D to select possible ways of satisfying the clauses in a 3CNF formula:

$$\begin{array}{lll} ClSat(f, t, t), & ClSat(t, t, t), & ClSat(t, f, f), \\ ClSat(f, t, f), & ClSat(t, t, f), & \\ ClSat(f, f, t), & ClSat(t, f, t), & \end{array}$$

where f and t are constants with the same meaning as above. The predicate $ClSat(\cdot, \cdot, \cdot)$ states which truth assignments to the *literals* (and not to the variables) satisfy a clause.

In the following, we use D_{st} to denote the set of all $SimLit$, $OppLit$, and $ClSat$ facts in D.

Finally, a fact $NonSat()$ is added to D.

The program has no TGDs. Below, we define some conjunctions that are used to define the NCs.

A first piece "reads" onto the query variables T_i the assignment to the variables in X encoded in a set of Val facts:

$$AssignX \equiv \bigwedge_{x_i \in X} Val(x_i, T_i).$$

A second piece "copies" the assignment on each variable x_i onto an occurrence of x_i as a positive literal in $\phi(X, Y)$:

$$Copy \equiv \bigwedge_{x_i \in X} SimLit(T_i, T_{j,k}),$$

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

A third piece forces the ground values f and t assigned to the variables $T_{j,k}$, so that assignments to the literals are consistent. Below, $\ell_{j,k} \sim \ell_{j',k'}$ means that literals $\ell_{j,k}$ and $\ell_{j',k'}$ refer to the same variable, and are both positive or both negative, while $\ell_{j,k} \not\sim \ell_{j',k'}$ means that they refer to the same variable, but one literal is positive and the other is negative.

$$\begin{aligned} Consist \equiv & \bigwedge_{\substack{\forall(\ell_{j,k}, \ell_{j',k'}) \\ \text{s.t. } \ell_{j,k} \sim \ell_{j',k'}}} SimLit(T_{j,k}, T_{j',k'}) \\ & \bigwedge_{\substack{\forall(\ell_{j,k}, \ell_{j',k'}) \\ \text{s.t. } \ell_{j,k} \not\sim \ell_{j',k'}}} OppLit(T_{j,k}, T_{j',k'}), \end{aligned}$$

where $T_{j,k}$ and $T_{j',k'}$ are variables with the above meaning.

The last piece checks the satisfiability of $\phi(X, Y)$, where m is the number of clauses of $\phi(X, Y)$:

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

Then, we add the following NC to Σ:

$$AssignX, Copy, Consist, Satisfied, NonSat() \rightarrow \bot.$$

We also add the following NCs to Σ:

$$\begin{array}{l} Val(X, f), Val(X, t) \rightarrow \bot, \\ Val(X, V), Dummy(Y) \rightarrow \bot, \\ NonSat(), Dummy(Y) \rightarrow \bot. \end{array}$$

The candidate $\leq$-repair. $D' = D_{st} \cup \{Dummy(x_i) \mid x_i \in X\}$. Notice that D' is a consistent selection of (D, Σ).

Below we show that $\forall X \exists Y \phi(X, Y)$ is valid iff D' is a $\leq$-repair of (D, Σ).

($\Rightarrow$) Assume that $\forall X \exists Y \phi(X, Y)$ is valid. We show that for every consistent selection D'' of (D, Σ), we have

$|D''| \leq |D'|$, and thus D' is a $\leq$-repair of (D, Σ). If D'' is a consistent selection of (D, Σ), there are two cases: either (a) D'' contains at least one $Dummy$ fact, or (b) it does not.

Case (a). If a $Dummy$ fact is in D'', then D'' contains neither Val facts nor the $NonSat()$ fact; thus $|D''| \leq |D'|$.

Consider Case (b). Observe that the maximum number of Val facts that D'' can contain is $|X|$. Hence, there are two cases: either (i) D'' contains strictly less than $|X|$ Val facts, or (ii) it does not. Consider Case (i). If the number of Val facts in D'' is strictly less than $|X|$, then $|D''| \leq |D'|$—at the very most, D'' can include the whole D_{st} and $NonSat()$. Consider now Case (ii). The number of Val facts in D'' is exactly $|X|$, and thus D'' encodes a truth assignment to the variables in X. There are again two cases: either (1) D'' excludes some fact in D_{st}, or (2) it does not. For Case (1), since D'' does not contain some fact in D_{st}, it must be the case that $|D''| \leq |D'|$. In Case (2), since D'' contains all facts in D_{st} and $\forall X \exists Y \phi(X, Y)$ is valid, D'' cannot include $NonSat()$. Thus $|D''| = |D'|$, and hence, $|D''| \leq |D'|$.

($\Leftarrow$) Assume that $\forall X \exists Y \phi(X, Y)$ is not valid, and let τ be a truth assignment to the variables in X such that $\phi(X/\tau, Y)$ is unsatisfiable. Let $D'' = D_{st} \cup \{ Val(x_i, f) \mid x_i \in X, \tau(x_i) = \mathit{false} \} \cup \{ Val(x_i, t) \mid x_i \in X, \tau(x_i) = \mathit{true} \} \cup \{NonSat()\}$. It is easy to see that D'' is consistent and $|D''| > |D'|$, and thus D' is not a $\leq$-repair. $\square$

Theorem 9. *$\subseteq$-$RC(\mathsf{NC})$ is* D$^{\text{P}}$-hard *in the ba-combined complexity.*

Proof. We exhibit a reduction from the following D$^{\text{P}}$-complete problem: given a pair (ϕ, ψ) of 3CNF formulas, decide whether ϕ is satisfiable and ψ is not satisfiable.

Below we build a knowledge base (D, Σ), with $\Sigma \in \mathsf{NC}$, and a database D', where all predicates have bounded arity.

The database. The following facts, whose meaning is the same as in the proof of Theorem 8, are added to D:

$$\begin{array}{lll} SimLit(f,f), & OppLit(f,t), & \\ SimLit(t,t), & OppLit(t,f), & \\ ClSat(f,t,t), & ClSat(t,t,t), & ClSat(t,f,f), \\ ClSat(f,t,f), & ClSat(t,t,f), & \\ ClSat(f,f,t), & ClSat(t,f,t). & \end{array}$$

Furthermore, a fact $Aux()$ is added to D.

The program has only the following NCs:

$$Consist^{\phi}, Satisfied^{\phi}, Aux() \rightarrow \bot,$$
$$Consist^{\psi}, Satisfied^{\psi} \rightarrow \bot,$$

where $Consist^{\phi}$ and $Satisfied^{\phi}$ (resp., $Consist^{\psi}$ and $Satisfied^{\psi}$) are the conjunctions $Consist$ and $Satisfied$ introduced in the proof of Theorem 8 defined for ϕ (resp., ψ).

The candidate $\subseteq$-repair. $D' = D \setminus \{Aux()\}$.

Below, we show that ϕ is satisfiable and ψ is not satisfiable iff D' is a $\subseteq$-repair of (D, Σ).

($\Rightarrow$) Assume that ϕ is satisfiable and ψ is not satisfiable. Since ψ is not satisfiable, D' is a consistent selection of (D, Σ). To show that D' is a $\subseteq$-repair of (D, Σ), it suffices to show that $D' \cup \{Aux()\}$ (which is D) is not consistent. Indeed, since ϕ is satisfiable, $D' \cup \{Aux()\}$ is not consistent.

($\Leftarrow$) Assume that ϕ is not satisfiable or ψ is satisfiable. If ψ is satisfiable, then D' is not a consistent selection of (D, Σ), and thus not a $\subseteq$-repair. If ψ is not satisfiable and ϕ is not satisfiable, then $D' \cup \{Aux()\}$ is a consistent selection of (D, Σ), and thus D' is not a $\subseteq$-repair of (D, Σ). $\square$

The missing tight lower bounds concern $\leq$-$RC(\mathsf{A}_\bot)$ and $\subseteq$-$RC(\mathsf{A}_\bot)$ in the ba-combined and combined complexity.

Theorem 10. *$\leq$-$RC(\mathsf{A}_\bot)$ and $\subseteq$-$RC(\mathsf{A}_\bot)$ are* D$^{\text{EXP}}$*-hard in the ba-combined*

Proof sketch. We provide a reduction from the following D$^{\text{EXP}}$-complete problem: Given two (independent) instances TP_1 and TP_2 of the tiling problem for the exponential squares $2^{n_1} \times 2^{n_1}$ and $2^{n_2} \times 2^{n_2}$, respectively, and two initial tiling conditions w_1 and w_2, respectively, decide whether TP_1 has solution with w_1 *and* TP_2 has *no* solution with w_2.

For $i = 1, 2$, we use the encoding by Eiter, Lukasiewicz, and Predoiu (2016) to create programs $\Sigma_{TP_i, |w_i|}$, and databases D_{w_i} and D_{TP_i}, such that TP_i has a solution with w_i iff $\left(D_{TP_i} \cup D_{w_i}, \Sigma_{TP_i, |w_i|}\right) \models Tiling^i()$. We also add an additional fact $Aux()$ to the database. The candidate repair is the database minus $Aux()$. Two NCs are added to the program, one to ensure that the candidate repair is consistent iff $Tiling()^2$ is not entailed (i.e., TP_2 has no solution with w_2) and another one to ensure that $Aux()$ cannot be taken if $Tiling()^1$ is entailed (i.e., TP_1 has solution with w_1). $\square$

The lower bounds for $\leq$ apply to $\leq_P$ and $\leq_w$, while those for $\subseteq$ apply to $\subseteq_P$.

6 Inconsistency-Tolerant Query Entailment

We first discuss membership and then hardness results.

6.1 Membership Results

The following two theorems provide upper bounds for each $\preccurlyeq \in \{\leq_w, \subseteq_P\}$ in the following cases: $\preccurlyeq$-$AR(\mathcal{L})$ and $\preccurlyeq$-$IAR(\mathcal{L})$ in the data, ba-combined, and combined complexity, as well as $\preccurlyeq$-$ICR(\mathcal{L})$ only in the data and ba-combined complexity.

Theorem 11. *If BCQ($\mathcal{L}$) is in the complexity class* $\mathbf{C}$ *in the data / ba-combined / combined complexity (resp., data / ba-combined complexity), then $\leq_w$-$AR(\mathcal{L})$ and $\leq_w$-$IAR(\mathcal{L})$ (resp., $\leq_w$-$ICR(\mathcal{L})$) is in* P *with an oracle for* NP$^{\mathbf{C}}$ *in the data / ba-combined / combined complexity (resp., data / ba-combined complexity).*

Proof. Let $KB = (D, \Sigma)$ be a knowledge base with $\Sigma \in \mathcal{L}$, let w be the weight function for D, and let q be a BCQ. First, we compute the maximum weight max of a consistent selection of KB. This can be done in polynomial time using an oracle in NP$^{\mathbf{C}}$ as follows. We can perform a binary search in the range $[0, w(D)]$ by asking the oracle whether there is a consistent selection with weight at least k. Such a binary search takes at most a polynomial number of steps, as weights are encoded in binary. The oracle has to guess a

$\mathcal{L}$	**Data**	*fp*-**comb.**	*ba*-**comb.**	**Comb.**
$\mathsf{L}_\perp, \mathsf{LF}_\perp, \mathsf{AF}_\perp$	Δ_2^{P}	Π_2^{P}	Δ_3^{P}	PSPACE
$\mathsf{S}_\perp, \mathsf{SF}_\perp$	Δ_2^{P}	Π_2^{P}	Δ_3^{P}	EXP
$\mathsf{A}_\perp$	Δ_2^{P}	Π_2^{P}	$\mathrm{P}^{\mathrm{NEXP}}$	$\mathrm{P}^{\mathrm{NEXP}}$
$\mathsf{G}_\perp$	Δ_2^{P}	Π_2^{P}	EXP	2EXP
$\mathsf{F}_\perp, \mathsf{GF}_\perp$	Δ_2^{P}	Π_2^{P}	Δ_3^{P}	EXP
$\mathsf{WS}_\perp, \mathsf{WA}_\perp$	Δ_2^{P}	Π_2^{P}	2EXP	2EXP

Table 4: Complexity of $\preccurlyeq$-$AR(\mathcal{L})$ for $\preccurlyeq \in \{\leq_w, \leq_P\}$. All entries are completeness results.

$\mathcal{L}$	**Data**	*fp*-**comb.**	*ba*-**comb.**	**Comb.**
$\mathsf{L}_\perp, \mathsf{LF}_\perp, \mathsf{AF}_\perp$	Δ_2^{P}	Δ_2^{P}	Δ_3^{P}	PSPACE
$\mathsf{S}_\perp, \mathsf{SF}_\perp$	Δ_2^{P}	Δ_2^{P}	Δ_3^{P}	EXP
$\mathsf{A}_\perp$	Δ_2^{P}	Δ_2^{P}	$\mathrm{P}^{\mathrm{NEXP}}$	$\mathrm{P}^{\mathrm{NEXP}}$
$\mathsf{G}_\perp$	Δ_2^{P}	Δ_2^{P}	EXP	2EXP
$\mathsf{F}_\perp, \mathsf{GF}_\perp$	Δ_2^{P}	Δ_2^{P}	Δ_3^{P}	EXP
$\mathsf{WS}_\perp, \mathsf{WA}_\perp$	Δ_2^{P}	Δ_2^{P}	2EXP	2EXP

Table 5: Complexity of $\preccurlyeq$-$IAR(\mathcal{L})$ and $\preccurlyeq$-$ICR(\mathcal{L})$ for $\preccurlyeq \in \{\leq_w, \leq_P\}$. All entries are completeness results.

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

Table 6: Complexity of $\subseteq_P$-$AR(\mathcal{L})$. All entries are completeness results.

$\mathcal{L}$	**Data**	*fp*-**comb.**	*ba*-**comb.**	**Comb.**
$\mathsf{L}_\perp, \mathsf{LF}_\perp, \mathsf{AF}_\perp$	co-NP	Θ_2^{P}	Π_2^{P}	PSPACE
$\mathsf{S}_\perp, \mathsf{SF}_\perp$	co-NP	Θ_2^{P}	Π_2^{P}	EXP
$\mathsf{A}_\perp$	co-NP	Θ_2^{P}	$\mathrm{P}^{\mathrm{NEXP}}$	$\mathrm{P}^{\mathrm{NEXP}}$
$\mathsf{G}_\perp$	co-NP	Θ_2^{P}	EXP	2EXP
$\mathsf{F}_\perp, \mathsf{GF}_\perp$	co-NP	Θ_2^{P}	Π_2^{P}	EXP
$\mathsf{WS}_\perp, \mathsf{WA}_\perp$	co-NP	Θ_2^{P}	2EXP	2EXP

Table 7: Complexity of $\subseteq_P$-$IAR(\mathcal{L})$ and $\subseteq_P$-$ICR(\mathcal{L})$. All entries are completeness results.

database $D' \subseteq D$, and then check whether (D', Σ) is consistent and $w(D') \geq k$. Since we are assuming that BCQ($\mathcal{L}$) is in $\mathbf{C}$, checking consistency of (D', Σ) requires polynomial time with an oracle in $\mathbf{C}$. Verifying $w(D') \geq k$ takes polynomial time as well.

Notice that, by knowing the maximum weight max of a consistent selection of KB, we can now verify whether a database $D' \subseteq D$ is a $\leq_w$-repair by checking whether (D', Σ) is consistent and $w(D') = max$.

Then, the $\mathrm{NP}^{\mathbf{C}}$ oracle is asked whether q is entailed under the $\leq_w$-AR, $\leq_w$-IAR, and $\leq_w$-ICR semantics. In particular, we ask the oracle whether the query is *not* entailed. The way in which the oracle computes the answer depends on the specific semantics considered, as discussed below.

- $\leq_w$-AR**:** The oracle guesses a database $D' \subseteq D$ and then checks whether D' is a $\leq_w$-repair and $(D', \Sigma) \not\models q$.
- $\leq_w$-IAR**:** The oracle guesses a database $D^\star \subseteq D$ along with one database $D_\alpha \subseteq D$ for each $\alpha \in D \setminus D^\star$, and then it checks that (1) $(D^\star, \Sigma) \not\models q$, and (2) each D_α is a $\leq_w$-repair with $\alpha \notin D_\alpha$.
- $\leq_w$-ICR**:** The oracle guesses a subset $D^\star$ of $Cl(KB)$ (the size of $Cl(KB)$ is polynomial in the input, because the program has bounded arity in the worst case) along with one database $D_\alpha \subseteq D$ for each $\alpha \in (Cl(KB) \setminus D^\star)$, and then it checks that (1) $(D^\star, \Sigma) \not\models q$, and (2) each D_α is a $\leq_w$-repair such that $\alpha \notin Cl(D_\alpha)$. □

Theorem 12. *If BCQ($\mathcal{L}$) is in the complexity class* $\mathbf{C}$ *in the data / ba-combined / combined complexity (resp., data / ba-combined complexity), then* $\subseteq_P$-$AR(\mathcal{L})$ *and* $\subseteq_P$-$IAR(\mathcal{L})$ *(resp.,* $\subseteq_P$-$ICR(\mathcal{L})$*) is in* co-($\mathrm{NP}^{\mathbf{C}}$) *in the data / ba-combined / combined complexity (resp., data / ba-combined complexity).*

Proof. Let $KB = (D, \Sigma)$ be a knowledge base with $\Sigma \in \mathcal{L}$, let $P = (\mathcal{P}_1, \ldots, \mathcal{P}_n)$ be a prioritization of D, and let q be a BCQ. We recall that deciding whether a database $D' \subseteq D$ is a $\subseteq_P$-repair can be done in polynomial time with an oracle in $\mathbf{C}$ (see Theorem 4). We can decide query *non*-entailment under the $\subseteq_P$-AR, $\subseteq_P$-IAR, and $\subseteq_P$-ICR semantics as discussed at the end of the proof of Theorem 11 (with the only difference being that we need to check whether databases are $\subseteq_P$-repairs), which yields the co-($\mathrm{NP}^{\mathbf{C}}$) upper bound for query entailment under the aforementioned semantics. □

The previous theorem does not provide upper bounds for $\leq_w$-$ICR(\mathcal{L})$ and $\subseteq_P$-$ICR(\mathcal{L})$ in the combined complexity. They are shown by the following theorem.

Theorem 13. $\leq_w$-$ICR(\mathcal{L})$ *and* $\subseteq_P$-$ICR(\mathcal{L})$ *in the combined complexity are in the complexity classes shown in Tables 5 and 7, respectively.*

Proof. The same argument in the proof of Theorem 7.1 in (Lukasiewicz et al. 2022) applies to $\leq_w$-$ICR(\mathcal{L})$ and $\subseteq_P$-$ICR(\mathcal{L})$ in the combined complexity, noticing that the upper bounds for $\leq_w$-$AR(\mathcal{L})$ and $\subseteq_P$-$AR(\mathcal{L})$ are the same as those of the classical (i.e., with set-inclusion maximal repairs) AR semantics in the combined complexity. □

The two theorems below state upper bounds for all three semantics, for $\leq_w$ and $\subseteq_P$, in the *fp*-combined complexity.

Theorem 14. *If BCQ($\mathcal{L}$) is in* $\mathbf{D}$ *in the data complexity and in* $\mathbf{C}$ *in the fp-combined complexity, then* $\leq_w$-$AR(\mathcal{L})$ *(resp.,* $\leq_w$-$IAR(\mathcal{L})$ *and* $\leq_w$-$ICR(\mathcal{L})$*) in the fp-combined complexity can be answered by a computation in* P *with an oracle for* $\mathrm{NP}^{\mathbf{D}}$*, followed by a computation in* co-($\mathrm{NP}^{\mathbf{C}}$) *(resp.,* $\mathbf{C}$*).*

Proof. Let $KB = (D, \Sigma)$ be a knowledge base with $\Sigma \in \mathcal{L}$, let w be the weight function for D, and let q be a query. The maximum weight of a consistent selection of KB can be computed via binary search in polynomial time calling an oracle for $\text{NP}^{\mathbf{D}}$ in the *fp*-combined complexity (as discussed at the beginning of the proof of Theorem 11, even though we can refer to the data complexity of standard BCQ answering for consistency checking, since we are referring to the *fp*-combined complexity, and thus the program is fixed).

The rest of the procedure depends on the specific semantics, as discussed below.

$\leq_w$**-**AR**:** We can decide *non*-entailment under the $\leq_w$-AR semantics by guessing $D' \subseteq D$ and verifying that D' is a $\leq_w$-repair and $(D', \Sigma) \not\models q$, which is in $\text{NP}^{\mathbf{C}}$.

$\leq_w$**-**IAR **(resp.,** $\leq_w$**-**ICR**):** We can calculate the intersection of all $\leq_w$-repairs D_I (resp., the intersection of all closed $\leq_w$-repairs D_C) as D minus all $\alpha \in D$ (resp., as $Cl(KB)$ minus all $\alpha \in Cl(KB)$) for which there exists a $\leq_w$-repair (resp., a closed $\leq_w$-repair) that does not contain α, which can be done in polynomial time with polynomially many parallel calls to an oracle for $\text{NP}^{\mathbf{D}}$. Then, we check $(D_I, \Sigma) \models q$ (resp., $(D_C, \Sigma) \models q$) (in $\mathbf{C}$). $\square$

Theorem 15. *If BCQ($\mathcal{L}$) is in $\mathbf{D}$ in the data complexity and in $\mathbf{C}$ in the fp-combined complexity, then $\subseteq_P$-$AR(\mathcal{L})$ (resp., $\subseteq_P$-$IAR(\mathcal{L})$ and $\subseteq_P$-$ICR(\mathcal{L})$) is in* co-$(\text{NP}^{\mathbf{C}})$ *(resp., can be answered by a computation in* P *with a constant number of rounds of polynomially many parallel calls to an* $\text{NP}^{\mathbf{D}}$ *oracle followed by a computation in* $\mathbf{C}$*) in the fp-combined complexity.*

Proof. We can apply the same procedures in the second part of the proof of Theorem 14 (thus, without the initial computation of the maximum weight of a consistent selection), with the only difference being that we refer to $\subseteq_P$-repairs rather than $\leq_w$-repairs. $\square$

The upper bounds for the $\leq_P$-semantics follow from those of the $\leq_w$-semantics, which generalizes the former.

6.2 Hardness Results

Hardness results are discussed per criterion.

Let us start with the $\leq_P$ preorder. For all three semantics, lower bounds in the data complexity follow from (Bienvenu, Bourgaux, and Goasdoué 2014b).

Theorem 16. *For each $S \in \{AR, IAR, ICR\}$, $\leq_P$-$S(\mathsf{NC})$ is Δ_2^{P}-hard in the data complexity.*

We now show Δ_3^{P}-hardness for all languages and all semantics in the *ba*-combined complexity. The lower bound is tight only for $\mathsf{L}_\perp$, $\mathsf{S}_\perp$, $\mathsf{F}_\perp$, and their specializations.

Theorem 17. *For each $S \in \{AR, IAR, ICR\}$, $\leq_P$-$S(\mathsf{NC})$ is Δ_3^{P}-hard in the ba-combined complexity.*

Proof. We provide a reduction from the following Δ_3^{P}-complete problem (Krentel 1992): given a 3DNF formula $\psi(X, Y)$ over variables X and Y, with $x_1, \ldots, x_n$ being the lexicographical order of the variables in X, decide whether the lexicographically maximum truth assignment τ_{max} to X such that $\forall Y\ \psi(X/\tau_{max}, Y)$ is valid, satisfies $\tau_{max}(x_n) = true$ (where such a τ_{max} is known to exist).

We can replace $\psi(X, Y)$ with $\neg\phi(X, Y)$, where $\phi(X, Y) \equiv \neg\psi(X, Y)$, and $\phi(X, Y)$ is a 3CNF formula that can be constructed in polynomial time.

Below we build a knowledge base (D, Σ), a prioritization P, and a query q, where all predicates have bounded arity.

The database. For each variable $x_i \in X$, the following facts are included into D:

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

as well as the following facts

$$\begin{array}{lll} SimLit(f, f), & OppLit(f, t), & \\ SimLit(t, t), & OppLit(t, f), & \\ ClSat(f, t, t), & ClSat(t, t, t), & ClSat(t, f, f), \\ ClSat(f, t, f), & ClSat(t, t, f), & \\ ClSat(f, f, t), & ClSat(t, f, t), & \end{array}$$

whose meaning is the same discussed in the proof of Theorem 8. Additionally, for each variable $x_i \in X$, the fact $ValTrue(x_i)$ is included into D.

In the following, we use D_{st} to denote the set of all $SimLit$, $OppLit$, and $ClSat$ facts in D.

The program has no TGDs, while the NCs are:

$$\begin{array}{l} AssignX, Copy, Consist, Satisfied \to \perp, \\ Val(X, f), Val(X, t) \to \perp, \\ Val(X, f), ValTrue(X) \to \perp, \end{array}$$

where $AssignX$, $Copy$, $Consist$, and $Satisfied$ are defined like in the proof of Theorem 8.

The prioritization. $P = (\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_{n+1})$, where

$$\begin{array}{l} \mathcal{P}_1 = D \setminus \{\, ValTrue(x_i) \mid 1 \leq i \leq n\}, \\ \mathcal{P}_i = \{\, ValTrue(x_{i-1})\} \text{ for } 2 \leq i \leq n+1\,. \end{array}$$

The query is $q = Val(x_n, t)$.

We recall that for the formula $\neg\phi(X, Y)$, the lexicographically maximum truth assignment τ_{max} to X such that $\forall Y\ \neg\phi(X/\tau_{max}, Y)$ is valid is known to exist.

We show that $\tau_{max}(x_n) = true$ iff $(D, \Sigma) \models_{\leq_P\text{-}S} q$, for each $S \in \{AR, IAR, ICR\}$. This is proven by showing that the following set R is the only $\leq_P$ repair of (D, Σ):

$$R = D_{st} \cup \{\, Val(x_i, f) \mid \tau_{max}(x_i) = false, 1 \leq i \leq n\} \cup \bigcup_{\tau_{max}(x_i) = true, 1 \leq i \leq n} \{\, Val(x_i, t), ValTrue(x_i)\}.$$

In particular, we show that, for every other consistent selection R' of (D, Σ), we have $R' <_P R$. Obviously, (R, Σ) satisfies the last two NCs above. Moreover, since τ_{max} is a truth assignment such that $\forall Y\ \neg\phi(X/\tau_{max}, Y)$ is valid, the first NC above is not violated. Thus, R is a consistent selection of (D, Σ). Consider now any other consistent selection R' of (D, Σ). Notice that $|R \cap \mathcal{P}_1| = |D_{st}| + n$. Thus, if $|R' \cap \mathcal{P}_1| < |D_{st}| + n$, then $R' <_P R$. Otherwise, $|R' \cap \mathcal{P}_1| = |D_{st}| + n$ ($|R' \cap \mathcal{P}_1|$ cannot be higher than $|D_{st}| + n$ in order for (R', Σ) to be consistent).

Notice that R' contains exactly one $Val(x_i, \cdot)$ for each $1 \leq i \leq n$. Let $R'' = R' \cup \{\, ValTrue(x_i) \mid Val(x_i, t) \in$

$R'\}$. Clearly, R'' is a consistent selection (because R' is a consistent selection, and adding the $ValTrue(x_i)$ facts keeps R'' consistent). Also, $R' \leq_P R''$. Let $\tau_{R''}$ be the truth assignment to X defined as follows: for each $x_i \in X$, $\tau_{R''}(x_i) = true$ iff $Val(x_i, t) \in R''$, and $\tau_{R''}(x_i) = false$ iff $Val(x_i, f) \in R''$. Notice that, in order for R'' to be consistent, it has to be the case that $\forall Y\, \neg\phi(X/\tau_{R''}, Y)$ is valid. Since τ_{max} is the lexicographically maximum truth assignment such that $\forall Y\, \neg\phi(X/\tau_{max}, Y)$ is valid, it must be the case that $R'' <_P R$, and thus $R' <_P R$. $\square$

The lower bounds for $\leq_P$-$IAR(\mathcal{L})$ and $\leq_P$-$ICR(\mathcal{L})$ in the data complexity turn out to be tight in the *fp*-combined complexity as well. All the remaining lower bounds follow from those of the AR, IAR, and ICR semantics under cardinality-maximal repairs (Lukasiewicz, Malizia, and Vaicenavičius 2019).

All lower bounds for the $\leq_w$ criterion follow from the lower bounds for the $\leq_P$ criterion discussed above and the fact that the former generalizes the latter.

Let us consider now the $\subseteq_P$ criterion. The lower bounds for $\subseteq_P$-$IAR(\mathcal{L})$ in the data and *fp*-combined complexity are stated in the following theorem, which follows from the proofs in (Bienvenu, Bourgaux, and Goasdoué 2014b).

Theorem 18. $\subseteq_P$-$IAR(\mathsf{NC})$ *is* co-NP-hard *in the data complexity and* Θ_2^{P}-hard *in the fp-combined complexity.*

The remaining lower bounds for $\subseteq_P$-$IAR(\mathcal{L})$ and all lower bounds for $\subseteq_P$-$AR(\mathcal{L})$ and $\subseteq_P$-$ICR(\mathcal{L})$ follow from those of the inconsistency-tolerant semantics with subset-maximal repairs—see (Lukasiewicz et al. 2022).

7 Related Work

Bienvenu, Bourgaux, and Goasdoué (2014a) studied the data and combined complexity of the AR and IAR semantics for DL-Lite$_{\mathcal{R}}$ for all maximality criteria here considered. For existential rule languages, the AR, IAR, and ICR semantics complexity was studied by Lukasiewicz et al. (2022) for subset-maximal repairs and by Lukasiewicz, Malizia, and Vaicenavičius (2019) for cardinality-maximal repairs, for all languages and complexity measures here considered.

A line of research closely related to this paper concerns inconsistency-tolerant query answering defined w.r.t. "preferred" repairs, which are selected *among the subset-maximal ones* on the basis of user preferences. Specifically, Staworko, Chomicki, and Marcinkowski (2012) introduced a framework where the AR semantics is generalized to take into account a priority relation expressing preferences among conflicting facts in the database. The class of *denial constraints* is considered, which is a slight generalization of negative constraints allowing comparison atoms. Preferences among facts are then used to define three notions of preferred repairs: *global*-, *Pareto*-, and *completion-optimal* repairs. These notions of optimal repairs have been recently applied to DLs by Bienvenu and Bourgaux (2020; 2022), who studied the data complexity of query entailment under the AR, IAR, and brave semantics based on optimal repairs, existence of a unique optimal repair, and enumeration of all optimal repairs. The data and combined complexity of the AR, IAR, and ICR semantics for existential rules when preferences are expressed via so-called *preference rules* have been investigated by Calautti et al. (2022). The crucial difference between the body of work above and the current paper is that the former considers subset-maximal repairs only (where preferred ones are identified on the basis of user preferences), while we consider different notions of maximality to define repairs in the first place.

8 Summary and Outlook

We have considered natural ways to define the maximality of repairs which are particularly relevant in practice, going beyond the "classical" subset-maximality criterion. The criteria that we have considered naturally arise in many real applications, e.g., when some database facts are considered more reliable than others. We have provided a thorough complexity analysis of repair checking and AR/IAR/ICR query entailment. Our results provide new insights into how different notions of repair behave in terms of complexity of common reasoning tasks. In summary, we can draw the following conclusions. As for repair checking, maximality criteria can be partitioned into two classes $\{\subseteq, \subseteq_P\}$ and $\{\leq, \leq_P, \leq_w\}$, with criteria in the same class having the same complexity, and criteria in the second class being at least as expensive as those in the first class. As for inconsistency-tolerant query entailment, maximality criteria can be partitioned into the following ordered list of classes $\{\subseteq\}, \{\subseteq_P\}, \{\leq\}, \{\leq_w, \leq_P\}$, with criteria in the same class having the same complexity, and criteria in a class being at least as expensive as those in the preceding classes.

Recently, there has been an increasing interest on explainable AI, including explaining query answering under existential rules (Ceylan et al. 2019; Ceylan et al. 2020a; Ceylan et al. 2021) and DLs (Bienvenu, Bourgaux, and Goasdoué 2019; Ceylan et al. 2020b). In particular, (Bienvenu, Bourgaux, and Goasdoué 2019; Lukasiewicz, Malizia, and Molinaro 2020; 2022) addressed the problem of explaining why a query is entailed or not under inconsistency-tolerant semantics, where repairs are subset-maximal. An interesting direction for future work is to address the same problem for the different types of repairs considered in this paper, also in the presence of generalized repairs, i.e., when rules can be repaired as well (Lukasiewicz, Malizia, and Molinaro 2018), and when a richer formalism to express preferences over repairs can be employed (Calautti et al. 2022; Lukasiewicz and Malizia 2019; 2022).

Acknowledgments

Thomas Lukasiewicz was supported by the Alan Turing Institute under the UK EPSRC grant EP/N510129/1, the AXA Research Fund, and the EU TAILOR grant 952215.

References

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

Benferhat, S.; Cayrol, C.; Dubois, D.; Lang, J.; and Prade, H. 1993. Inconsistency management and prioritized syntax-based entailment. In *Proc. IJCAI*, 640–647.

Bienvenu, M., and Bourgaux, C. 2020. Querying and repairing inconsistent prioritized knowledge bases: Complexity analysis and links with abstract argumentation. In *Proc. KR*, 141–151.

Bienvenu, M., and Bourgaux, C. 2022. Querying inconsistent prioritized data with ORBITS: Algorithms, implementation, and experiments. In *Proc. KR*.

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

Bienvenu, M.; Bourgaux, C.; and Goasdoué, F. 2014b. Querying inconsistent description logic knowledge bases under preferred repair semantics. Technical report, https://www.lri.fr/~bibli/Rapports-internes/2014/RR1572.pdf.

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.

Brewka, G. 1989. Preferred subtheories: An extended logical framework for default reasoning. In *Proc. IJCAI*.

Brewka, G. 1991. *Nonmonotonic Reasoning - Logical Foundations of Commonsense*, volume 12 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press.

Calautti, M.; Greco, S.; Molinaro, C.; and Trubitsyna, I. 2022. Preference-based inconsistency-tolerant query answering under existential rules. *Artif. Intell.* 312:103772.

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.

Ceylan, İ. İ.; Lukasiewicz, T.; Malizia, E.; Molinaro, C.; and Vaicenavicius, A. 2020a. Explanations for negative query answers under existential rules. In *Proc. KR*, 223–232.

Ceylan, İ. İ.; Lukasiewicz, T.; Malizia, E.; and Vaicenavicius, A. 2020b. Explanations for ontology-mediated query answering in description logics. In *Proc. ECAI*, 672–679.

Ceylan, İ. İ.; Lukasiewicz, T.; Malizia, E.; Molinaro, C.; and Vaicenavicius, A. 2021. Preferred explanations for ontology-mediated queries under existential rules. In *Proc. AAAI*, 6262–6270.

Chomicki, J., and Marcinkowski, J. 2005. Minimal-change integrity maintenance using tuple deletions. *Inf. Comput.* 197(1–2):90–121.

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

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.

Krentel, M. W. 1992. Generalizations of Opt P to the polynomial hierarchy. *Theor. Comput. Sci.* 97(2):183–198.

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

Lifschitz, V. 1985. Computing circumscription. In *Proc. IJCAI*, 121–127.

Lopatenko, A., and Bertossi, L. E. 2016. Complexity of consistent query answering in databases under cardinality-based and incremental repair semantics (extended version). *CoRR* abs/1605.07159.

Lukasiewicz, T., and Malizia, E. 2019. Complexity results for preference aggregation over (m)CP-nets: Pareto and majority voting. *Artif. Intell.* 272:101–142.

Lukasiewicz, T., and Malizia, E. 2022. Complexity results for preference aggregation over (m)CP-nets: Max and rank voting. *Artif. Intell.* 303:art. no. 103636.

Lukasiewicz, T.; Malizia, E.; Martinez, M. V.; Molinaro, C.; Pieris, A.; and Simari, G. I. 2022. Inconsistency-tolerant query answering for existential rules. *Artif. Intell.* 307:103685.

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 Molinaro, C. 2020. Explanations for inconsistency-tolerant query answering under existential rules. In *Proc. AAAI*, 2909–2916.

Lukasiewicz, T.; Malizia, E.; and Molinaro, C. 2022. Explanations for negative query answers under inconsistency-tolerant semantics. In *Proc. IJCAI*, 2705–2711.

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.

Staworko, S.; Chomicki, J.; and Marcinkowski, J. 2012. Prioritized repairing and consistent query answering in relational databases. *Ann. Math. Artif. Intell.* 64(2-3):209–246.

Tsamoura, E.; Carral, D.; Malizia, E.; and Urbani, J. 2021. Materializing knowledge bases via trigger graphs. *Proc. VLDB Endow.* 14(6):943–956.

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