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 \(C\) of constants, a set \(N\) of labeled nulls, and a set \(V\) 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 \(h : C \cup N \cup V \rightarrow C \cup N \cup V\) that is the identity on \(C\) and
maps \( N \) to \( C \cup N \). With a slight abuse of notation, homomorphisms are applied also to (sets/conjunctions of) atoms. A **conjunctive query** (CQ) \( q \) has the form \( \exists Y \phi(X, Y) \), where \( \phi(X, Y) \) is a conjunction of atoms without nulls. The answer to \( q \) over an instance \( I \), denoted \( q(I) \), is the set of all tuples \( t \) over \( C \) for which there is a homomorphism \( h \) such that \( h(\phi(X, Y)) \subseteq I \) and \( h(X) = t \). A **Boolean CQ** (BCQ) \( q \) is a CQ \( \exists Y \phi(Y) \), 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 \( h(\phi(Y)) \subseteq I \).

**Dependencies** A **tuple-generating dependency** (TGD) \( \sigma \) is a first-order formula \( \forall X \forall Y (\phi(X, Y) \rightarrow \exists Z p(X, Z)) \), where \( X, Y, \) and \( Z \) are pairwise disjoint sets of variables, \( \phi(X, Y) \) is a conjunction of atoms, and \( p(X, Z) \) is an atom, all without nulls; \( \phi(X, Y) \) is the **body** of \( \sigma \), denoted \( \text{body}(\sigma) \), while \( p(X, Z) \) is the **head** of \( \sigma \), denoted \( \text{head}(\sigma) \). 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 \( \sigma \), written \( I \models \sigma \), if the following holds: whenever there exists a homomorphism \( h \) such that \( h(\phi(X, Y)) \subseteq I \), then there exists \( h'(g) \supseteq h(X) \), where \( h(X) \) is the restriction of \( h \) on \( X \), such that \( h'(g(X, Z)) \subseteq I \). A **negative constraint** (NC) \( \nu \) is a first-order formula \( \forall X (\phi(X) \rightarrow \bot) \), where \( X \subseteq V \), \( \phi(X) \) is a conjunction of atoms without nulls, called the **body** of \( \nu \) and denoted \( \text{body}(\nu) \), and \( \bot \) denotes the truth constant \( \text{false} \). An instance \( I \) satisfies an NC \( \nu \), written \( I \models \nu \), if there is no homomorphism \( h \) such that \( h(\phi(X)) \subseteq I \). We will use \( q_{\sigma} \) to denote the BCQ \( \exists X \phi(X) \). Given a set \( \Sigma \) of TGDs and NCs, \( I \) satisfies \( \Sigma \), written \( I \models \Sigma \), if \( I \) satisfies each TGD and NC of \( \Sigma \). For brevity, we omit the universal quantifiers in front of TGDs and NCs, and use the comma (instead of \( \land \)) for conjunctioning. For a class \( \mathcal{C} \) of TGDs, \( \mathcal{C}_\bot \) denotes the combination of \( \mathcal{C} \) with arbitrary NCs. \( \mathcal{N} \) denotes the language using only NCs. Finite sets of TGDs and NCs are called **programs**, and TGDs are also called **existential rules**.

The Datalog\(^*\) 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 \( (D, \Sigma) \), where \( D \) is a database and \( \Sigma \) is a program. For a program \( \Sigma \), \( \Sigma_T \) and \( \Sigma_{NC} \) denote the subsets of \( \Sigma \) containing the TGDs and NCs of \( \Sigma \), respectively. The set of **models** of KB = \( (D, \Sigma) \), denoted \( \text{mods}(KB) \), is the set of instances \( \{ I \mid I \supseteq D \land I \models \Sigma \} \). We say that KB is **consistent** if \( \text{mods}(KB) \neq \emptyset \), otherwise KB is **inconsistent**. The answer to a CQ \( q \) relative to KB is the set of tuples \( \text{ans}(q, KB) = \bigcap \{ q(I) \mid I \in \text{mods}(KB) \} \). The answer to a BCQ \( q \) is true, denoted \( KB \models q \), if \( \text{ans}(q, KB) \neq \emptyset \). 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 \( t \) of constants, decide whether \( t \in \text{ans}(q, KB) \). 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** \( \mathcal{A}C0 \) 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. \( D^P = \text{NP} \land \text{co-NP} \) (resp., \( D^{\text{Exp}} = \text{NEXP} \land \text{co-NEXP} \)) 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). \( \Sigma^P_2 \) is the class of problems decidable in nondeterministic polynomial time with an NP oracle, and \( \Pi^P_2 \) is the complement of \( \Sigma^P_2 \). \( \Theta^P_2 \) 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. \( \Delta^P_2 \) (resp., \( \Delta^P_4 \)) is the class of problems decidable in deterministic polynomial time with an NP (resp., \( \Sigma^P_2 \) oracle). \( p^{\text{NP}} \) 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: \( \mathcal{A}C^0 \subseteq \mathcal{P} \subseteq \mathcal{N} \), co-NP \( \subseteq D^P \subseteq \Theta^P_2 \subseteq \Delta^P_2 \subseteq \Sigma^P_2 \), \( \Pi^P_2 \subseteq \Delta^P_2 \subseteq \text{PSPACE} \subseteq \text{EXP} \subseteq \text{NEXP} \), co-NEXP \( \subseteq D^{\text{Exp}} \subseteq P^{\text{NEXP}} \subseteq 2\text{EXP} \).

<table>
<thead>
<tr>
<th>Data</th>
<th>fp-comb.</th>
<th>ba-comb.</th>
<th>Comb.</th>
</tr>
</thead>
<tbody>
<tr>
<td>L, LF, AF</td>
<td>in ( \mathcal{A}C^0 )</td>
<td>NP</td>
<td>NP</td>
</tr>
<tr>
<td>S, SF</td>
<td>in ( \mathcal{A}C^0 )</td>
<td>NP</td>
<td>NP</td>
</tr>
<tr>
<td>A</td>
<td>in ( \mathcal{A}C^0 )</td>
<td>NP</td>
<td>NEXP</td>
</tr>
<tr>
<td>G</td>
<td>P</td>
<td>NP</td>
<td>EXP</td>
</tr>
<tr>
<td>F, GF</td>
<td>P</td>
<td>NP</td>
<td>NP</td>
</tr>
<tr>
<td>WS, WA</td>
<td>P</td>
<td>2EXP</td>
<td>2EXP</td>
</tr>
</tbody>
</table>

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 $(D', \Sigma)$ 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 $\Sigma_T$, entails some NC; more formally, a culprit is a subset $C$ of $D$ s.t. $(C, \Sigma_T) \models q$, for some $\nu \in \Sigma_{NC}$. A culprit for an NC $\nu$ is an “explanation” (Ceylan et al. 2019) of $q$. By deleting from $D$ a hitting set (Chomicki and Markinkowski 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 $\preceq$ over a set $S$ of databases (i.e., $\preceq$ is a reflexive and transitive binary relation on $S$), for two elements $D'$ and $D'' \in S$, we write $D' \preceq D''$ to denote that $D' \preceq D''$ and $D'' \preceq D'$. A database $D \in S$ is $\preceq$-maximal in $S$ iff there is no $D' \in S$ such that $D \preceq D'$.

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

$\text{Rep}_{\preceq}(KB)$ denotes the set of all $\preceq$-repairs of $KB$.

For a knowledge base $KB = (D, \Sigma)$, the closure of $KB$, denoted $\text{Cl}(KB)$, is the set of all facts built from constants in $D$ and $\Sigma$, entailed by $D$ and the TGDs of $\Sigma$.

Definition 2. Let $KB$ be a knowledge base, let $q$ be a BCQ, and let $\preceq$ be a preorder over the consistent selections of $KB$.

- $KB$ entails $q$ under the $\preceq$-ABox repair semantics ($\preceq$-AR), denoted $KB \models_{\preceq-AR} q$, if $(D', \Sigma) \models q$ for all $D' \in \text{Rep}_{\preceq}(KB)$.

- $KB$ entails $q$ under the $\preceq$-intersection of repairs semantics ($\preceq$-IAR), denoted $KB \models_{\preceq-IAR} q$, if $(D_1, \Sigma) \models q$, where $D_1 = \bigcap\{D' \mid D' \in \text{Rep}_{\preceq}(KB)\}$.

- $KB$ entails $q$ under the $\preceq$-intersection of closed repairs semantics ($\preceq$-ICR), denoted $KB \models_{\preceq-ICR} q$, if $(D_C, \Sigma) \models q$, where $D_C = \bigcap\{\text{Cl}(D', \Sigma) \mid D' \in \text{Rep}_{\preceq}(KB)\}$.

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

Problem: $\preceq-RC(L)$.

Input: A knowledge base $(D, \Sigma)$ with $\Sigma \in L$, and a database $D'$.

Question: Is $D'$ a $\preceq$-repair of $(D, \Sigma)$?

Problem: $\preceq-S(L)$, with $S \in \{AR, IAR, ICR\}$.

Input: A knowledge base $(D, \Sigma)$ with $\Sigma \in L$, and a BCQ $q$.

Question: Does $(D, \Sigma) \models_{\preceq-S} q$ hold?

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

Weights ($\leq_w$). The database $D$ comes along with a weight function $w: D \to \mathbb{N}$ 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 $P = (P_1, \ldots, P_n)$ of the database $D$, that is, $P$ is a partition of $D$ into the priority levels $P_i$, where $P_1$ contains the most reliable facts, and $P_n$ 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 $1 \leq i \leq n$, $|D_1 \cap P_i| \leq |D_2 \cap P_i|$, or there is some $1 \leq i \leq n$ such that $|D_1 \cap P_i| < |D_2 \cap P_i|$ and for every $1 \leq j < i$, $|D_1 \cap P_j| = |D_2 \cap P_j|$.

Prioritized Set Inclusion ($\subseteq_P$). For every $D_1, D_2 \subseteq D$, we write $D_1 \subseteq_P D_2$ iff for every $1 \leq i \leq n$, $D_1 \cap P_i \subseteq D_2 \cap P_i$, or there is some $1 \leq i \leq n$ such that $D_1 \cap P_i \subset D_2 \cap P_i$ and for every $1 \leq j < i$, $D_1 \cap P_j = D_2 \cap P_j$.

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 $\leq_w$ (resp., $\subseteq_w$), 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 $\{\leq_w, \leq_P\}$ and $\{\subseteq_w, \subseteq_P\}$, with criteria in the same class having the same complexity. Thus, moving from $\leq_w$ to the more general criteria $\leq_P$ and $\subseteq_w$ does not incur an increase of complexity (this holds also when moving from $\leq_w$ to $\subseteq_P$). Likewise, the complexity does not increase when moving from $\subseteq_w$ to $\leq_P$.

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 $\leq_P$. The $IAR$ and $ICR$ semantics have the same complexity across all maximality criteria, which is a behavior shown by $\leq_w$ as well (Lukasiewicz, Malizia, and Vaicenavičius 2019), while this does not hold for $\subseteq_w$ (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 $\leq_w$ (resp., $\leq_P$) and $\leq_w$ (resp., $\leq_P$). The complexity results for $\leq_w$ and $\leq_P$ can be found in (Lukasiewicz et al. 2022) and (Lukasiewicz, Malizia, and Vaicenavičius 2019), respectively. When we move from $\leq_w$ to the more general $\leq_P$ 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 ($L_\perp, S_\perp, A_\perp$, and their sublanguages) in the data and
Table 2: Complexity of \( \approx w - RC(\mathcal{L}) \) for \( \approx \in \{ \leq w, \leq p, \leq \} \). All entries are completeness results.

<table>
<thead>
<tr>
<th>( \mathcal{L} )</th>
<th>Data</th>
<th>ba-comb.</th>
<th>Comb.</th>
</tr>
</thead>
<tbody>
<tr>
<td>( L \downarrow, LF \downarrow, AF \downarrow )</td>
<td>co-NP</td>
<td>( \Pi_2^p )</td>
<td>PSPACE</td>
</tr>
<tr>
<td>( S \downarrow, SF \downarrow )</td>
<td>co-NP</td>
<td>( \Pi_2^p )</td>
<td>EXP</td>
</tr>
<tr>
<td>( A \downarrow )</td>
<td>co-NP</td>
<td>( \text{dExp} )</td>
<td>( \text{dExp} )</td>
</tr>
<tr>
<td>( G \downarrow )</td>
<td>co-NP</td>
<td>EXP</td>
<td>2EXP</td>
</tr>
<tr>
<td>( F \downarrow, GF \downarrow )</td>
<td>co-NP</td>
<td>( \Pi_2^p )</td>
<td>EXP</td>
</tr>
<tr>
<td>( WS \downarrow, WA \downarrow )</td>
<td>co-NP</td>
<td>2EXP</td>
<td>2EXP</td>
</tr>
</tbody>
</table>

Table 3: Complexity of \( \approx w - RC(\mathcal{L}) \) for \( \approx \in \{ \leq p, \leq \} \). All non-“in” entries are completeness results.

<table>
<thead>
<tr>
<th>( \mathcal{L} )</th>
<th>Data</th>
<th>ba-comb.</th>
<th>Comb.</th>
</tr>
</thead>
<tbody>
<tr>
<td>( L \downarrow, LF \downarrow, AF \downarrow )</td>
<td>in P</td>
<td>( \text{dExp} )</td>
<td>PSPACE</td>
</tr>
<tr>
<td>( S \downarrow, SF \downarrow )</td>
<td>in P</td>
<td>( \text{dExp} )</td>
<td>EXP</td>
</tr>
<tr>
<td>( A \downarrow )</td>
<td>in P</td>
<td>( \text{dExp} )</td>
<td>( \text{dExp} )</td>
</tr>
<tr>
<td>( G \downarrow )</td>
<td>in P</td>
<td>EXP</td>
<td>2EXP</td>
</tr>
<tr>
<td>( F \downarrow, GF \downarrow )</td>
<td>in P</td>
<td>( \text{dExp} )</td>
<td>EXP</td>
</tr>
<tr>
<td>( WS \downarrow, WA \downarrow )</td>
<td>in P</td>
<td>2EXP</td>
<td>2EXP</td>
</tr>
</tbody>
</table>

Theorem 3. If BCQ(\( \mathcal{L} \)) is in the complexity class \( \mathcal{C} \) in the data / ba-combined / combined complexity, then \( \leq w - RC(\mathcal{L}) \) is in co-(NP\( \mathcal{C} \)) in the data / ba-combined / combined complexity.

Proof. Let \((D, \Sigma)\) 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 \((D, \Sigma)\), we guess a subset \( D'' \) of \( D \) and check that (1) \((D'', \Sigma)\) is inconsistent, or (2) \((D'', \Sigma)\) is consistent and \( w(D') < w(D'') \). Both conditions can be verified in polynomial time with an oracle in \( \mathcal{C} \). □

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

Proof. Let \( KB = (D, \Sigma) \) be a knowledge base with \( \Sigma \in \mathcal{L} \), let \( P = (P_1, \ldots, P_n) \) be a prioritization, and let \( D' \) be a database. To decide whether \( D' \) is a \( \leq p \)-repair of \( KB \), we check that (1) \((D', \Sigma)\) is consistent and (2) for every \( 1 \leq i \leq n \) and fact \( f \) in \( P_i \), \((D' \setminus (P_1 \cup \cdots \cup P_i)) \cup \{ f \}, \Sigma \) is inconsistent. Condition (1) can be verified with a linear number of co-C checks. Condition (2) can be verified with a polynomial number of C checks. □

To obtain the \( d^p \) (resp., \( d^\text{Exp} \)) 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 \( \leq w - RC(A_{\downarrow}) \) in the ba-combined and combined complexity requires a dedicated analysis.

Theorem 5. \( \leq w - RC(A_{\downarrow}) \) is in \( d^\text{Exp} \) in the ba-combined and combined complexity.

Proof. Let \((D, \Sigma)\) 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(\( A_{\downarrow} \)) is in NEXP. We need to check that (1) \((D', \Sigma)\) is consistent and (2) there is no \( D'' \subseteq D \) such that \((D'', \Sigma)\) 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 \((D'', \Sigma)\) is inconsistent or \( w(D') \geq w(D'') \). Notice that checking inconsistency of a single \((D'', \Sigma)\) 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.

The upper bounds in Table 2 (resp., Table 3) for \( \leq p \) and \( \leq \) (resp., \( \leq \)) follow from those of \( \leq w \) (resp., \( \leq p \)) 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. \( \text{(Lopatenko and Bertossi 2016)} \leq - RC(\text{NC}) \) is co-NP-hard in the data complexity.

The theorem below provides lower bounds for \( \leq \) and \( \leq - RC(\mathcal{L}) \) 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(ℒ) is reducible in polynomial time to \(\preceq -RC(ℒ)\) and \(\succeq -RC(ℒ)\) in the ba-combined and combined complexity.

Proof. From a knowledge base \((D, \Sigma)\) with \(\Sigma \in ℒ\), and a BCQ \(q = \exists \Phi(\Psi)\), we derive an instance of \(\preceq -RC(ℒ)\) (resp., \(\succeq -RC(ℒ)\)) as follows: the knowledge base is \((D, \Sigma')\), where \(\Sigma' = \Sigma \cup \{ \Phi(\Psi) \rightarrow \top \}\), and the candidate \(\preceq -\) (resp., \(\succeq -\)) repair is \(D\). Notice that \(\Sigma' \in ℒ\) and the reduction takes polynomial time. It can be easily verified that \((D, \Sigma) \not\models q\) iff \(D\) is a \(\preceq -\) (resp., \(\succeq -\)) repair of \((D, \Sigma')\).

The theorem above does not provide tight lower bounds for both \(\preceq -\) and \(\succeq -\) in the following cases: A_\perp in the ba-combined and combined complexity, as well as LF_\perp, AF_\perp, and SF_\perp in the ba-combined complexity. We provide specific hardness results for such cases in the following.

Theorem 8. \(\succeq -RC(ℒ)\) is \(\Pi_2\)-hard in the ba-combined complexity.

Proof. We provide a reduction from the following \(\Pi_2\)-complete problem: decide the validity of a quantified Boolean formula \(\forall \exists X \exists Y \phi(X, Y)\), where \(\phi\) 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 \((D, \Sigma)\), with \(\Sigma \in NC\), and a database \(D'\), where all predicates have bounded arity.

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

\[
\text{Val}(x_i, f), \quad \text{Val}(x_i, t), \quad \text{Dummy}(x_i),
\]

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:

\[
\begin{align*}
\text{SimLit}(f, f), & \quad \text{OppLit}(f, t), \\
\text{SimLit}(t, t), & \quad \text{OppLit}(t, f),
\end{align*}
\]

where \(f\) and \(t\) are constants with the same meaning as above. The predicate \(\text{SimLit}(\cdot, \cdot)\) 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 \(\text{OppLit}(\cdot, \cdot)\) 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:

\[
\begin{align*}
\text{ClSat}(f, t, t), & \quad \text{ClSat}(t, t, t), \quad \text{ClSat}(t, f, f), \\
\text{ClSat}(f, t, f), & \quad \text{ClSat}(t, t, f), \quad \text{ClSat}(t, f, t),
\end{align*}
\]

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

In the following, we use \(D_\delta\) to denote the set of all \(\text{SimLit}, \text{OppLit},\) and \(\text{ClSat}\) facts in \(D\).

Finally, a fact \(\text{NonSat}()\) 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 \(\text{Val}\) facts:

\[
\text{Assign}X = \bigwedge_{x_i \in X} \text{Val}(x_i, T_i).
\]

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

\[
\text{Copy} = \bigwedge_{x_i \in X} \text{SimLit}(T_i, T_{j,k}),
\]

where in each atom \(\text{SimLit}(T_i, T_{j,k})\), \(T_{j,k}\) is a variable for the Boolean value of the literal \(\ell_{j,k} = x_i\) in \(\phi(X, Y)\). Observe that, in order for \(\text{Copy}\) 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.

\[
\text{Consist} = \bigwedge_{\forall \ell_{j,k}, \ell'_{j',k'} \text{s.t. } \ell_{j,k} \sim \ell'_{j',k'}} \text{SimLit}(T_{j,k}, T'_{j',k'}) \wedge \bigwedge_{\forall \ell_{j,k}, \ell'_{j',k'} \text{s.t. } \ell_{j,k} \not\sim \ell'_{j',k'}} \text{OppLit}(T_{j,k}, T'_{j',k'}),
\]

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 \(\phi(X, Y)\):

\[
\text{Satisfied} = \bigwedge_{j=1}^{m} \text{ClSat}(T_{j,1}, T_{j,2}, T_{j,3}).
\]

Then, we add the following NC to \(\Sigma\):

\[
\text{AssignX}, \text{Copy}, \text{Consist}, \text{Satisfied}, \text{NonSat}() \rightarrow \bot.
\]

We also add the following NCs to \(\Sigma\):

\[
\begin{align*}
\text{Val}(X, f), & \quad \text{Val}(X, t) \rightarrow \bot, \\
\text{Val}(X, V), & \quad \text{Dummy}(Y) \rightarrow \bot, \quad \text{NonSat}(), \quad \text{Dummy}(Y) \rightarrow \bot.
\end{align*}
\]

The candidate \(\preceq -\) repair. \(D' = D_\delta \cup \{ \text{Dummy}(x_i) \mid x_i \in X\}\). Notice that \(D'\) is a consistent selection of \((D, \Sigma)\).

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

(\(\Rightarrow\)) Assume that \(\forall X \exists Y \phi(X, Y)\) is valid. We show that for every consistent selection \(D''\) of \((D, \Sigma)\), we have
$|D''| \leq |D'|$, and thus $D'$ is a $\preceq$-repair of $(D, \Sigma)$. If $D''$ is a consistent selection of $(D, \Sigma)$, 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 $\tau$ be a truth assignment to the variables in $X$ such that $\phi(X/\tau, Y)$ is unsatisfiable. Let $D'' = D_{st} \cup \{Val(x_i, f) | x_i \in X, \tau(x_i) = false\} \cup \{Val(x_i, t) | x_i \in X, \tau(x_i) = true\} \cup \{NonSat()\}$. It is easy to see that $D''$ is consistent and $|D''| > |D'|$, and thus $D'$ is not a $\preceq$-repair.

Theorem 9. $\preceq$-RC(NC) is $D^\mathbb{P}$-hard in the data-combined complexity.

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

Below we build a knowledge base $(D, \Sigma)$, with $\Sigma \in NC$, 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 $D$:

- SimLit($f, f$), OppLit($f, t$), SimLit($t, f$), OppLit($t, t$), ClSat($f, t, t$), ClSat($t, t, t$), ClSat($t, f, f$), ClSat($f, t, f$), ClSat($f, t, f$), ClSat($f, t, f$).

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

The program has only the following NCs:

- $\text{Consist}^\phi, \text{Satisfied}^\phi, \text{Aux}() \rightarrow \perp$, $\text{Consist}^\phi, \text{Satisfied}^\phi \rightarrow \perp$,

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

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

Below, we show that $\phi$ is satisfiable and $\psi$ is not satisfiable iff $D'$ is a $\preceq$-repair of $(D, \Sigma)$.

$(\Rightarrow)$ Assume that $\phi$ is satisfiable and $\psi$ is not satisfiable. Since $\phi$ is satisfiable, $D'$ is a consistent selection of $(D, \Sigma)$. To show that $D'$ is a $\preceq$-repair of $(D, \Sigma)$, it suffices to show that $D' \cup \{\text{Aux}()\}$ (which is $D$) is not consistent. Indeed, since $\phi$ is satisfiable, $D' \cup \{\text{Aux}()\}$ is not consistent.

$(\Leftarrow)$ Assume that $\phi$ is not satisfiable or $\psi$ is satisfiable. If $\psi$ is satisfiable, then $D'$ is not a consistent selection of $(D, \Sigma)$, and thus not a $\preceq$-repair. If $\psi$ is not satisfiable and $\phi$ is not satisfiable, then $D' \cup \{\text{Aux}()\}$ is a consistent selection of $(D, \Sigma)$, and thus $D'$ is not a $\preceq$-repair of $(D, \Sigma)$.

The missing tight lower bounds concern $\preceq$-RC($A_1$) and $\preceq$-RC($A_\perp$) in the ba-combined and combined complexity.

Theorem 10. $\preceq$-RC($A_1$) and $\preceq$-RC($A_\perp$) are $D^\mathbb{Exp}$-hard in the ba-combined

Proof sketch. We provide a reduction from the following $D^\mathbb{Exp}$-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 $\text{BCQ}(\Sigma_{TP_i, \{w_i\}}, D_{w_i}, D_{TP_i}) \models \text{Tiling}'()$. We also add an additional fact $\text{Aux}()$ to the database. The candidate repair is the database minus $\text{Aux}()$. Two NCs are added to the program, one to ensure that the candidate repair is consistent iff $\text{Tiling}'()^2$ is not entailed (i.e., $TP_2$ has no solution with $w_2$) and another one to ensure that $\text{Aux}()$ cannot be taken if $\text{Tiling}'()$ is entailed (i.e., $TP_1$ has solution with $w_1$).

The lower bounds for $\preceq$ apply to $\leq_\mathbb{P}$ and $\leq_w$, while those for $\preceq$ apply to $\leq_\mathbb{P}$.

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 $\preceq \in \{\leq_w, \leq_\mathbb{P}\}$ in the following cases: $\preceq$-AR($\mathcal{L}$) and $\preceq$-IAR($\mathcal{L}$) in the data, ba-combined, and combined complexity, as well as $\preceq$-ICR($\mathcal{L}$) only in the data and ba-combined complexity.

Theorem 11. If BCQ($\mathcal{L}$) is in the complexity class $\mathcal{C}$ 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 $\mathcal{P}$ with an oracle for NP$^C$ 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 $w[D]$ of a consistent selection of $KB$. This can be done in polynomial time using an oracle in NP$^C$ 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
database $D' \subseteq D$, and then check whether $(D', \Sigma)$ is consistent and $w(D') \geq k$. Since we are assuming that BCQ($\mathcal{L}$) is in $\mathcal{C}$, checking consistency of $(D', \Sigma)$ requires polynomial time with an oracle in $\mathcal{C}$. Verifying $w(D') \geq k$ takes polynomial time as well.

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

Then, the $\text{NP}^\mathcal{C}$ oracle is asked whether $q$ is entailed under the $\leq w$-$\text{AR}$, $\leq w$-$\text{IAR}$, and $\leq w$-$\text{ICR}$ 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.

$\leq w$-$\text{AR}$: The oracle guesses a database $D' \subseteq D$ and then checks whether $D'$ is a $\leq w$-repair and $(D', \Sigma) \not\models q$.

$\leq w$-$\text{IAR}$: The oracle guesses a database $D* \subseteq D$ along with one database $D_\alpha \subseteq D$ for each $\alpha \in D \setminus D^*$, and then it checks that (1) $(D^*, \Sigma) \not\models q$, and (2) each $D_\alpha$ is a $\leq w$-repair with $\alpha \notin D_\alpha$.

$\leq w$-$\text{ICR}$: The oracle guesses a subset $D^*$ of $\text{Cl}(KB)$ (the size of $\text{Cl}(KB)$ 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 $\alpha \in \text{Cl}(KB) \setminus D^*$, and then it checks that (1) $(D^*, \Sigma) \not\models q$, and (2) each $D_\alpha$ is a $\leq w$-repair such that $\alpha \notin \text{Cl}(D_\alpha)$.

**Theorem 12.** If BCQ($\mathcal{L}$) is in the complexity class $\mathcal{C}$ in the data / ba-combined / combined complexity (resp., data / ba-combined complexity), then $\leq p$-$\text{AR}(\mathcal{L})$ and $\leq p$-$\text{IAR}(\mathcal{L})$ (resp., $\leq p$-$\text{ICR}(\mathcal{L})$) is in $\text{co-(NP}^\mathcal{C})$ 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 $P = (P_1, \ldots, P_n)$ be a prioritization of $D$, and let $q$ be a BCQ. We recall that deciding whether a database $D' \subseteq D$ is a $\leq p$-repair can be done in polynomial time with an oracle in $\mathcal{C}$ (see Theorem 4). We can decide query non-entailment under the $\leq p$-$\text{AR}$, $\leq p$-$\text{IAR}$, and $\leq p$-$\text{ICR}$ 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 $\leq p$-repaars), which yields the $\text{co-(NP}^\mathcal{C})$ upper bound for query entailment under the aforementioned semantics.

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

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

The same argument in the proof of Theorem 7.1 in (Łukasiewicz et al. 2022) applies to $\leq w$-$\text{ICR}(\mathcal{L})$ and $\leq p$-$\text{ICR}(\mathcal{L})$ in the combined complexity, noticing that the upper bounds for $\leq w$-$\text{AR}(\mathcal{L})$ and $\leq p$-$\text{AR}(\mathcal{L})$ 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 $\leq p$, in the $\text{fp}$-combined complexity.

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

**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 $NP^D$ 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.

\[ \leq_{w\cdot AR} \text{AR}: \text{We can decide non-entailment under the } \leq_{w\cdot AR} \text{ semantics by guessing } D' \subseteq D \text{ and verifying that } D' \text{ is a } \leq_{w\cdot AR} \text{-repair and } (D', \Sigma) \not\models q, \text{ which is in } NP^G. \]

\[ \leq_{w\cdot IAR} \text{ (resp., } \leq_{w\cdot ICR}) \text{: We can calculate the intersection of all } \leq_{w\cdot IAR} \text{-repairs } D_1 \text{ (resp., the intersection of all closed } \leq_{w\cdot IAR} \text{-repairs } D_C) \text{ as } D \text{ minus all } \alpha \in D \text{ (resp., as } Cl(KB) \text{ minus all } \alpha \in Cl(KB)) \text{ for which there exists a } \leq_{w\cdot IAR} \text{-repair (resp., a closed } \leq_{w\cdot IAR} \text{-repair) that does not contain } \alpha, \text{ which can be done in polynomial time with polynomially many parallel calls to an oracle for } NP^D. \text{ Then, we check } (D_1, \Sigma) \models q \text{ (resp., } (D_C, \Sigma) \models q \text{) in } C). \]

**Theorem 15.** If $BCQ(L)$ is in $D$ in the data complexity and in $C$ in the $fp$-combined complexity, then $\leq_{p\cdot AR}(L)$ (resp., $\leq_{p\cdot IAR}(L)$ and $\leq_{p\cdot ICR}(L)$) is in $co-(NP^G)$ (resp., can be answered by a computation in $P$ with a constant number of rounds of polynomially many parallel calls to an $NP^D$ oracle followed by a computation in $C$) 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 $\leq_p$-repairs rather than $\leq_{w\cdot}$. $\square$

The upper bounds for the $\leq_p$-semantics follow from those of the $\leq_{w\cdot}$-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, Bourgault, and Gousoude 2014b).

**Theorem 16.** For each $S \in \{AR, IAR, ICR\}$, $\leq_{p\cdot S}(NC)$ is $\Delta^p_3$-hard in the data complexity.

We now show $\Delta^p_3$-hardness for all languages and all semantics in the $ba\cdot$-combined complexity. The lower bound is tight only for $L_\perp$, $S_\perp$, $F_\perp$, and their specializations.

**Theorem 17.** For each $S \in \{AR, IAR, ICR\}$, $\leq_{p\cdot S}(NC)$ is $\Delta^p_3$-hard in the $ba\cdot$-combined complexity.

**Proof.** We provide a reduction from the following $\Delta^p_3$-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 $\tau_{max}$ to $X$ such that $\forall Y \psi(X/\tau_{max}, Y)$ is valid, satisfies $\tau_{max}(x_n) = true$ (where such a $\tau_{max}$ is known to exist).

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

Below we build a knowledge base $(D, \Sigma)$, 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 $D$:

\[ Val(x_i, f), \quad Val(x_i, t), \]

as well as the following facts:

\[ SimLit(f, t), \quad OppLit(f, t), \]

\[ SimLit(t, t), \quad OppLit(t, f), \]

\[ ClSat(f, t, t), \quad ClSat(t, t, t), \quad ClSat(t, f, f), \]

\[ ClSat(f, t, f), \quad ClSat(t, f, t), \]

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:

\[ AssignX, \quad Copy, \quad Consist, \quad Satisfied \rightarrow \bot, \]

\[ Val(X, f), \quad Val(X, t) \rightarrow \bot, \]

\[ Val(X, f), \quad ValTrue(X) \rightarrow \bot, \]

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

**The prioritization.** $P = (P_1, P_2, \ldots, P_{n+1})$, where

\[ P_1 = D \setminus \{ValTrue(x_i) \mid 1 \leq i \leq n\}, \]

\[ P_i = \{ValTrue(x_{i-1})\} \text{ for } 2 \leq i \leq n + 1. \]

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

We recall that for the formula $-\phi(X, Y)$, the lexicographically maximum truth assignment $\tau_{max}$ 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 $(D, \Sigma) \models_{\leq_p} q$, for each $S \in \{AR, IAR, ICR\}$. This is proven by showing that the following set $R$ is the only $\leq_p$ repair of $(D, \Sigma)$:

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

In particular, we show that, for every other consistent selection $R'$ of $(D, \Sigma)$, we have $R' <_p R$. Obviously, $(R, \Sigma)$ satisfies the last two NCs above. Moreover, since $\tau_{max}$ 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 $(D, \Sigma)$. Consider now any other consistent selection $R'$ of $(D, \Sigma)$. Notice that $|R \cap P_1| = |D_{st}| + n$. Thus, if $|R' \cap P_1| < |D_{st}| + n$, then $R' <_p R$. Otherwise, $|R' \cap P_1| = |D_{st}| + n$ ($R' \cap P_1$ cannot be higher than $|D_{st}| + n$ in order for $(R', \Sigma)$ to be consistent).

Notice that $R'$ contains exactly one $Val(x_i, \cdot)$ for each $1 \leq i \leq n$. Let $R'' = R' \cup \{ValTrue(x_i) \mid Val(x_i, t) \in \}$. We can replace $\psi(X, Y)$ with $-\phi(X, Y)$, where $\phi(X, Y) \equiv -\psi(X, Y)$, and $\phi(X, Y)$ is a 3CNF formula that can be constructed in polynomial time.
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 IAR/ARI/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 \( \{\leq_p, \leq_w\} \) and \( \{\leq_p, \leq_p, \leq_w\} \), 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 \( \{\}, \{\leq_p\}, \{\leq_p, \leq_w\}, \{\leq_w, \leq_p\} \), 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


