Preference-based Inconsistency-Tolerant Query Answering under Existential Rules

Marco Calautti[1], **Sergio Greco**[2], **Cristian Molinaro**[2], **Irina Trubitsyna**[2]
[1]DISI, University of Trento, Italy
[2]DIMES, University of Calabria, Italy
marco.calautti@unitn.it, {greco,cmolinaro,trubitsyna}@dimes.unical.it

Abstract

Query answering over inconsistent knowledge bases is a problem that has attracted a great deal of interest over the years. Different inconsistency-tolerant semantics have been proposed, and most of them are based on the notion of *repair*, that is, a "maximal" consistent subset of the database. In general, there can be several repairs, so it is often natural and desirable to express preferences among them. In this paper, we propose a framework for querying inconsistent knowledge bases under user preferences for existential rule languages. We provide generalizations of popular inconsistency-tolerant semantics taking preferences into account and study the data and combined complexity of different relevant problems.

1 Introduction

The problem of querying inconsistent knowledge bases has been investigated for many years. Different inconsistency-tolerant semantics have been proposed in the literature, that is, approaches to provide meaningful query answers despite the knowledge base being inconsistent.

Several popular semantics rely on the notion of *repair*, which is a "maximal" consistent subset of the database. Since inconsistency can be resolved in different ways, in general there are multiple repairs, which are then used in various ways to determine "valid" query answers. For instance, the *ABox repair (AR)* semantics (Arenas, Bertossi, and Chomicki 1999; Lembo et al. 2010) considers a query answer valid if it can be inferred from each of the repairs of the knowledge base. The *intersection of repairs (IAR)* semantics (Lembo et al. 2010) considers an answer valid if it can be inferred from the intersection of the repairs. The *intersection of closed repairs (ICR)* semantics (Bienvenu 2012) considers an answer valid if it can be inferred from the intersection of the closures of the repairs. We illustrate these semantics in our running example below.

Example 1. Consider a knowledge base (D, Σ) describing orders in a restaurant, where D is the union of the following four databases:

$$\begin{aligned}
D_{\text{base}} &= \{\mathsf{meat}(\mathsf{beef}), \mathsf{order}(\mathsf{o})\},\\
D_{\text{food}} &= \{\mathsf{main}(\mathsf{o}, \mathsf{beef}), \mathsf{side}(\mathsf{o}, \mathsf{cheese})\},\\
D_{\text{drinks}} &= \{\mathsf{drink}(\mathsf{o}, \mathsf{red}), \mathsf{drink}(\mathsf{o}, \mathsf{beer})\},\\
D_{\text{desserts}} &= \{\mathsf{dessert}(\mathsf{o}, \mathsf{cake}), \mathsf{dessert}(\mathsf{o}, \mathsf{pie})\},
\end{aligned}$$

stating that beef is a meat dish and o is an order; furthermore, order o has beef for the main course, cheese as side dish, red wine and beer as drinks, and cake and pie as desserts.

The ontology Σ contains the following dependencies:

$$\begin{aligned}
&\sigma_1 : \mathsf{main}(X, Y), \mathsf{meat}(Y) \rightarrow \mathsf{hasMeat}(X),\\
&\sigma_2 : \mathsf{drink}(X, \mathsf{red}) \rightarrow \mathsf{hasWine}(X),\\
&\sigma_3 : \mathsf{drink}(X, \mathsf{beer}) \rightarrow \mathsf{hasBeer}(X),\\
&\sigma_4 : \mathsf{drink}(X, Y) \rightarrow \mathsf{hasDrink}(X),\\
&\sigma_5 : \mathsf{main}(X, Y) \rightarrow \exists Z\, \mathsf{side}(X, Z),
\end{aligned}$$

$$\begin{aligned}
&\nu_1 : \mathsf{hasBeer}(X), \mathsf{hasWine}(X) && \rightarrow \bot,\\
&\nu_2 : \mathsf{hasBeer}(X), \mathsf{dessert}(X, Y) && \rightarrow \bot,\\
&\nu_3 : \mathsf{hasBeer}(X), \mathsf{side}(X, \mathsf{cheese}) && \rightarrow \bot,\\
&\nu_4 : \mathsf{hasWine}(X), \mathsf{dessert}(X, \mathsf{cake}), \mathsf{dessert}(X, \mathsf{pie}) && \rightarrow \bot.
\end{aligned}$$

The existential rules σ_1–σ_4 specify when an order includes a meat dish, wine, beer, and a drink, respectively. Then, σ_5 says that a main course always comes with a side dish.

The negative constraints ν_1–ν_3 say that beer cannot be included in an order together with wine, or a dessert, or cheese. Finally, the negative constraint ν_4 says that an order cannot include wine, cake, and pie all together.

The knowledge base is clearly inconsistent and admits four repairs:

$$\begin{aligned}
R_1 &= D_{\text{base}} \cup D_{\text{food}} \cup \{\mathsf{drink}(\mathsf{o}, \mathsf{red})\} \cup \{\mathsf{dessert}(\mathsf{o}, \mathsf{pie})\},\\
R_2 &= D_{\text{base}} \cup D_{\text{food}} \cup \{\mathsf{drink}(\mathsf{o}, \mathsf{red})\} \cup \{\mathsf{dessert}(\mathsf{o}, \mathsf{cake})\},\\
R_3 &= D_{\text{base}} \cup \{\mathsf{main}(\mathsf{o}, \mathsf{beef})\} \cup \{\mathsf{drink}(\mathsf{o}, \mathsf{beer})\},\\
R_4 &= D_{\text{base}} \cup D_{\text{food}} \cup D_{\text{desserts}}.
\end{aligned}$$

The query $Q = \exists X, Y\, \mathsf{dessert}(X, Y)$, asking whether some dessert has been ordered, is not entailed under the AR semantics, as repair R_3 does not include any dessert. Thus, Q is not entailed also under the IAR and ICR semantics. □

The AR, IAR, and ICR semantics have been extensively studied for both description logics (DLs) and existential rule languages. In each semantics, all repairs are equally important, and there is no way to prefer one over another. However, in many applications it is natural and desirable to express preferences, e.g., when one data source is more reliable than another, or when information is time-stamped and more recent facts are preferred over earlier ones. To cope with such scenarios, different preference-based approaches have been proposed—e.g., Bienvenu, Bourgaux, and Goasdoué (2014a) defined variants of the AR and IAR semantics where classical repairs are replaced by various types of preferred repairs, while Staworko, Chomicki, and Marcinkowski (2012) defined variants of the AR semantics where different kinds of preferred repairs are determined on the basis of a priority relation between database facts.

However, preferences do not always hold in general but may depend on several underlying factors. Most often users have different preferences under different circumstances. For instance, this is the case for advanced personalized web applications, where customers who want to make a purchase or reservation have different preferences depending on a variety of factors. As an example, location, time, and weather conditions may influence the place one wants to visit. The importance of expressing conditions under which preferences hold has been recognized in several works (Ceylan, Lukasiewicz, and Peñaloza 2015; Ceylan et al. 2017; Lukasiewicz, Martinez, and Simari 2013b; Lukasiewicz et al. 2014; Stefanidis, Pitoura, and Vassiliadis 2011; Agrawal, Rantzau, and Terzi 2006), which addressed the problem of querying *consistent* data in the presence of "context-dependent" preferences. This feature comes particularly in handy in ontological settings, where part of the knowledge is not known in advance, but it can well affect which preferences should be applied (we will provide some examples later on). In such settings, it is also particularly relevant to express preferences on information that is not directly available in the database, but it can be derived from the data via knowledge provided by an ontology.

To deal with the scenarios discussed above, we propose a framework for querying inconsistent knowledge bases under user preferences. Inspired by the work of Brewka, Niemelä, and Truszczynski (2003), where preference rules are used to determine "optimal" answer sets of logic programs, we enrich knowledge bases with preference rules, so as to narrow down the set of repairs to a set of *preferred* ones. We then define preferred counterparts of the AR, IAR, and ICR semantics by looking only at preferred repairs. By restricting the set of repairs to be considered for query answering, more information can be derived from the inconsistent knowledge base. Of course, this should not be done in an arbitrary way, but according to the users' preferences. We illustrate the basic idea of our approach in the following example.

Example 2. Consider again the scenario of Example 1. Suppose we would like to express preferences among repairs in such a way that the presence of some items in an order determines what other items are preferred. In our framework, one could specify this kind of preferences by means of the following set Π of *preference rules*:

$$\begin{aligned}&\rho_1: \mathsf{hasWine}(X) \succ \mathsf{hasBeer}(X) \leftarrow \mathsf{hasMeat}(X),\\ &\rho_2: \mathsf{dessert}(X,\mathsf{cake}) \succ \mathsf{dessert}(X,\mathsf{cherries}) \leftarrow \mathsf{main}(X,Y),\\ &\rho_3: \mathsf{dessert}(X,\mathsf{pie}) \succ \mathsf{dessert}(X,\mathsf{cake}) \leftarrow\\ &\qquad\qquad\qquad\qquad \exists Y\, \mathsf{hasDrink}(X), \mathsf{side}(X,Y).\end{aligned}$$

The first preference rule says that when an order includes meat, wine is preferred over beer.

The second preference rule states that when an order includes a main course, cake is preferred over cherries.

The third preference rule says that when an order includes a drink and some side dish, pie is preferred over cake.

We will see later that R_1 and R_4 (cf. Example 1) are the repairs that best satisfy the above preference rules, and thus they are *preferred*. Only preferred repairs are considered in the definition of *preferred* inconsistency-tolerant semantics. Thus, for instance, query Q of Example 1 is entailed under our *preferred* AR, IAR, and ICR semantics. $\square$

In the previous example, it is worth noting that the applicability of preference rules depends also on knowledge that is not directly available in the database, but it is rather derived from the database via the ontology—e.g., hasMeat in ρ_1 and hasDrink in ρ_3. Furthermore, derived knowledge can be used to express preferences, e.g., this is the case for ρ_1, where the preference is expressed over information derived from the database via the ontology (namely, hasWine and hasBeer).

In this paper, we introduce a novel framework for querying inconsistent knowledge bases under existential rules in the presence of user preferences. We provide a thorough analysis of the data and combined complexity of the following relevant problems: preferred repair checking, and preferred AR, IAR, and ICR entailment. Our results consider a wide range of existential rule languages.

The rest of the paper is organized as follows. Preliminaries are reported in Section 2. Our framework is presented in Section 3. A complexity analysis is reported in Section 4. Related work is discussed in Section 5. Concluding remarks and directions for future work are discussed in Section 6.

2 Preliminaries

In this section, we briefly recall some basics on existential rules from the context of Datalog$^{\pm}$ (Calì, Gottlob, and Lukasiewicz 2012). We also recall inconsistency-tolerant semantics for querying inconsistent knowledge bases.

General. We assume a set $\mathbf{C}$ of *constants*, a set $\mathbf{N}$ of *labeled nulls*, and a set $\mathbf{V}$ of *variables*. A *term* t is a constant, null, or variable. We also assume a set of *predicates*, each associated with an *arity*, i.e., a non-negative integer. An *atom* has the form $p(\mathbf{t})$, where p is an n-ary predicate, and $\mathbf{t}$ is a tuple of n terms. An atom containing only constants is also called a *fact*. Conjunctions of atoms are often identified with the sets of their atoms. An *instance* I is a (possibly infinite) set of atoms $p(\mathbf{t})$, where $\mathbf{t}$ is a tuple of constants and nulls. A *database* D is a finite instance that contains only constants. A *homomorphism* is a substitution $h\colon \mathbf{C}\cup\mathbf{N}\cup\mathbf{V}\to\mathbf{C}\cup\mathbf{N}\cup\mathbf{V}$ that is the identity on $\mathbf{C}$ and maps $\mathbf{N}$ to $\mathbf{C}\cup\mathbf{N}$. With a slight abuse of notation, homomorphisms are applied also to (sets/conjunctions of) atoms. A *conjunctive query* (CQ) Q has the form $\exists\mathbf{Y}\varphi(\mathbf{X},\mathbf{Y})$, where $\mathbf{X},\mathbf{Y}$ are tuples of variables, and $\varphi(\mathbf{X},\mathbf{Y})$ is a conjunction of atoms over the variables in $\mathbf{X},\mathbf{Y}$ 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 $h(\varphi(\mathbf{X},\mathbf{Y}))\subseteq I$ and $h(\mathbf{X})=\mathbf{t}$. A *Boolean CQ* (BCQ) Q is a CQ $\exists\mathbf{Y}\varphi(\mathbf{Y})$, i.e., all variables are existentially quantified; Q is *true* over I, denoted $I\models Q$, if $Q(I)\neq\emptyset$, i.e., there is a homomorphism h with $h(\varphi(\mathbf{Y}))\subseteq I$.

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 tuples of variables, $\varphi(\mathbf{X}, \mathbf{Y})$ is a conjunction of atoms over $\mathbf{X}, \mathbf{Y}$, and $p(\mathbf{X}, \mathbf{Z})$ is an atom, all without nulls; $\varphi(\mathbf{X}, \mathbf{Y})$ is the *body* of σ, denoted $body(\sigma)$, while $p(\mathbf{X}, \mathbf{Z})$ is the *head* of σ, denoted $head(\sigma)$. For clarity, we consider single-atom-head TGDs; however, our results extend to TGDs with a conjunction of atoms in the head. An instance I satisfies σ, written $I \models \sigma$, if the following holds: whenever there exists a homomorphism h such that $h(\varphi(\mathbf{X}, \mathbf{Y})) \subseteq I$, then there exists $h' \supseteq h|_{\mathbf{X}}$, where $h|_{\mathbf{X}}$ is the restriction of h on $\mathbf{X}$, such that $h'(p(\mathbf{X}, \mathbf{Z})) \in I$. A *negative constraint (NC)* ν is a first-order formula $\forall \mathbf{X}\, \varphi(\mathbf{X}) \rightarrow \bot$, with $\mathbf{X}$ a tuple of variables, $\varphi(\mathbf{X})$ is a conjunction of atoms over $\mathbf{X}$, without nulls, called the *body* of ν and denoted $body(\nu)$, and $\bot$ denotes the truth constant *false*. An instance I satisfies ν, written $I \models \nu$, if there is no homomorphism h such that $h(\varphi(\mathbf{X})) \subseteq I$.

Given a set Σ of TGDs and NCs, I satisfies Σ, written $I \models \Sigma$, if I satisfies each TGD and NC of Σ. For brevity, we omit the universal quantifiers in front of TGDs and NCs, and use the comma for conjoining atoms. Given a class of TGDs $\mathcal{L}$, we denote by $\mathcal{L}_\bot$ the formalism obtained by combining $\mathcal{L}$ with arbitrary NCs. Finite sets of TGDs and NCs are also called *programs*, and TGDs are also called *existential rules*.

Knowledge Bases. A *knowledge base* is a pair (D, Σ), where D is a database and Σ is a program. The subsets of a program Σ containing its TGDs and NCs are Σ_T and Σ_{NC}, respectively. The set of *models* of $KB = (D, \Sigma)$, denoted $mods(KB)$, is the set of instances $\{I \mid I \supseteq D \wedge I \models \Sigma\}$. We say that KB is *consistent* if $mods(KB) \neq \emptyset$, otherwise KB is *inconsistent*. The *answer* to a CQ Q relative to KB is the set of tuples $ans(Q, KB) = \bigcap\{Q(I) \mid I \in mods(KB)\}$. The answer to a BCQ Q is *true*, denoted $KB \models Q$, if $ans(Q, KB) \neq \emptyset$. The decision version of the *CQ answering* problem is: given a knowledge base KB, a CQ Q, and a tuple of constants $\mathbf{t}$, decide whether $\mathbf{t} \in ans(Q, KB)$. Since CQ answering can be reduced in LOGSPACE to BCQ answering, we focus on BCQs. Following Vardi (1982), the *data complexity* considers only the database as part of the input (the rest is fixed), while the *combined complexity* considers everything as part of the input.

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

We refer to (Lukasiewicz et al. 2015a) for an overview of the complexity of BCQ entailment for the above languages.

Inconsistency-Tolerant Semantics for Query Answering. We formally define the AR, IAR, and ICR semantics.

Let $KB = (D, \Sigma)$ be a knowledge base. A *repair* of KB is an inclusion-maximal subset R of D such that (R, Σ) is consistent. We use $Rep(KB)$ to denote the set of all repairs of KB. The *closure* $Cn(KB)$ of KB is the set of all facts (thus containing constants only) entailed by D and the TGDs of Σ. Let Q be a BCQ.

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

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

3 Preference Rules

In this section, we introduce the syntax and semantics of *preference rules*, which allow users to express preferences among repairs. The aim is to use such rules to identify a set of *preferred repairs* among all possible ones, and use only the preferred repairs for AR, IAR, and ICR entailment.

Our framework consists of two main steps:

1. First, for each repair R and preference rule ρ, we define when R *satisfies* ρ.
2. Then, preferred repairs are those that satisfy an inclusion-maximal set of preference rules.

Before presenting formal definitions, we illustrate satisfaction of a preference rule by a repair in the following example.

Example 3. Consider the knowledge base of Example 1 and the preference rule ρ_3 of Example 2. An instantiation of ρ_3 for order o is the following preference rule:[1]

$$\rho_3' : \mathsf{dessert}(\mathsf{o}, \mathsf{pie}) \succ \mathsf{dessert}(\mathsf{o}, \mathsf{cake}) \leftarrow \exists Y \mathsf{hasDrink}(\mathsf{o}), \mathsf{side}(\mathsf{o}, Y).$$

The preference rule consists of two parts: a *preference* $\mathsf{dessert}(\mathsf{o}, \mathsf{pie}) \succ \mathsf{dessert}(\mathsf{o}, \mathsf{cake})$ stating that $\mathsf{dessert}(\mathsf{o}, \mathsf{pie})$ is preferred over $\mathsf{dessert}(\mathsf{o}, \mathsf{cake})$, and a *precondition* $\exists Y \mathsf{hasDrink}(\mathsf{o}), \mathsf{side}(\mathsf{o}, Y)$ stating when the preference should be taken into account. Thus, the whole preference rule can be read as follows: if the precondition holds in a repair (in this case we also say that the preference rule is "applicable" in the repair), then the preference has to be "fulfilled" by the repair. When this implication holds, we say that the repair *satisfies* the preference rule.

Specifically, for a repair R, different cases may occur:

1. (R, Σ) does not entail the precondition, and thus the preference rule is *not* applicable in R. This case occurs for repair R_4.

[1]We will formally define what instantiations are in the following, using the notion of *ground instances* of preference rules.

2. (R, Σ) entails the precondition, and thus the preference rule is applicable in R. This case occurs for repairs R_1, R_2, and R_3. Then, we need to check whether the preference $\mathsf{dessert}(\mathsf{o}, \mathsf{pie}) \succ \mathsf{dessert}(\mathsf{o}, \mathsf{cake})$ is fulfilled by R, and different (sub)cases may occur:
   (a) (R, Σ) does not entail $\mathsf{dessert}(\mathsf{o}, \mathsf{pie})$, and (R, Σ) does not entail $\mathsf{dessert}(\mathsf{o}, \mathsf{cake})$. This case can be seen as a scenario where the *preference is irrelevant* to R—roughly speaking, none of the options hold. This case occurs for repair R_3.
   (b) (R, Σ) entails $\mathsf{dessert}(\mathsf{o}, \mathsf{pie})$. In this case, regardless of whether (R, Σ) entails $\mathsf{dessert}(\mathsf{o}, \mathsf{cake})$, the preference is fulfilled. This case occurs for repair R_1.
   (c) (R, Σ) does not entail $\mathsf{dessert}(\mathsf{o}, \mathsf{pie})$, and (R, Σ) entails $\mathsf{dessert}(\mathsf{o}, \mathsf{cake})$. In this case, the preference is *not* fulfilled, or it is "violated". This case occurs for repair R_2.

In Case 1, since ρ'_3 is not applicable in R, the implication expressed by the preference rule trivially holds, and we say that *R satisfies ρ'_3*.

In Case 2(b), ρ'_3 is applicable in R and its preference is fulfilled by R. In such a case, we say that *R satisfies ρ'_3*.

In Case 2(c), ρ'_3 is applicable in R and its preference is *not* fulfilled by R. In this case, we say that *R does not satisfy ρ'_3*.

Let us now discuss the case of irrelevance, namely Case 2(a). Irrelevance can be treated in different ways. On the one hand, it can be considered to be on a different level w.r.t. (non-)satisfaction. On the other hand, one can argue that not violating the preference is better than violating it. Under this view, Case 2(a) should be treated like Case 2(b). We adopt this latter view, and thus we say that *R satisfies ρ'_3* in Case 2(a). An analogous choice has been adopted also by Brewka, Niemelä, and Truszczynski (2003). The rationale behind this choice is that a repair where the preference is irrelevant should not be penalized. To provide a concrete simple example, consider a knowledge base having three repairs $\{\mathsf{dessert}(\mathsf{o}, \mathsf{pie})\}$, $\{\mathsf{dessert}(\mathsf{o}, \mathsf{cake})\}$, and $\{\mathsf{drink}(\mathsf{o}, \mathsf{red})\}$, with a preference rule $\mathsf{dessert}(\mathsf{o}, \mathsf{pie}) \succ \mathsf{dessert}(\mathsf{o}, \mathsf{cake}) \leftarrow$, where the precondition is empty and thus is always entailed. While the first repair should be clearly preferred over the second one, there is no reason to get rid of the third one. Indeed, in our framework, the first and third repairs satisfy the preference rule, and thus both are preferred ones. □

In the following, we formally define the syntax and semantics of our preference rules.

Syntax. The syntax of preference rules is as follows.

Definition 4. A *preference rule* ρ is an expression of the form

$$p(\mathbf{X}) \succ q(\mathbf{Y}) \leftarrow \exists \mathbf{Z}\, \varphi(\mathbf{Z}, \mathbf{W}),$$

where $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{Z}$, and $\mathbf{W}$ are tuples of variables such that $\mathbf{X} \cap \mathbf{Z} = \mathbf{Y} \cap \mathbf{Z} = \emptyset$, $p(\mathbf{X})$ and $q(\mathbf{Y})$ are atoms (whose variables are exactly $\mathbf{X}$ and $\mathbf{Y}$, respectively), and $\varphi(\mathbf{Z}, \mathbf{W})$ is a (possibly empty) conjunction of atoms (whose variables are exactly $\mathbf{Z} \cup \mathbf{W}$), all without nulls. □

In the previous definition, the right-hand (resp., left-hand) side of $\leftarrow$ is called the *body* (resp., *head*) of ρ and is denoted as $body(\rho)$ (resp., $head(\rho)$). Intuitively, the head expresses a preference among two atoms, while the body expresses a precondition for such a preference to be applied.

If every variable of ρ is existentially quantified, then ρ is *ground*—thus, a ground preference rule can contain only constants and existentially quantified variables, that is, it has the form

$$p(\mathbf{a}) \succ q(\mathbf{b}) \leftarrow \exists \mathbf{Z}\, \varphi(\mathbf{Z}, \mathbf{c}),$$

with $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$ tuples of constants and $\mathbf{Z}$ a tuple of variables.

A *preference program* is a finite set of preference rules.

The syntax (as well as the semantics) of our preference rules is akin to that of the preference rules proposed by Brewka, Niemelä, and Truszczynski (2003). However, as opposed to the approach of Brewka, Niemelä, and Truszczynski (2003), we additionally allow existentially quantified variables in the body of preference rules. This choice is motivated by the presence of existential rules in the knowledge base. We illustrate this aspect in the following example.

Example 5. Consider the preference rule ρ'_3 of Example 3, which is applicable in repairs R_1–R_3.

It is easy to see that (R_1, Σ) and (R_2, Σ) entail the body of ρ'_3, as both R_1 and R_2 contain a drink and a side dish for order o.

Consider now repair R_3. Notice that R_3 does not contain $\mathsf{side}(\mathsf{o}, \mathsf{cheese})$ (which was in the database), because R_3 contains $\mathsf{drink}(\mathsf{o}, \mathsf{beer})$ and the negative constraint ν_3 states that an order cannot include both beer and cheese. However, R_3 contains $\mathsf{main}(\mathsf{o}, \mathsf{beef})$, and the existential rule σ_5 states that every order including a main course must include a side dish as well. Thus, (R_3, Σ) entails that order o comes with some "unknown" side dish (which has to be different from cheese, because of ν_3). Thus, ρ'_3 is applicable in R_3. Roughly speaking, the existential quantification in σ_5 (there exists a side dish for an order including a main course) can be used to satisfy the precondition of ρ'_3 (if there exists a side dish for an order including a drink, then pie is preferred over cake). □

A prioritized knowledge base combines a knowledge base with a preference program.

Definition 6. A *prioritized knowledge base $\mathcal{K}$* is a triple (D, Σ, Π), where D is a database, Σ is a program, and Π is a preference program. □

We say that $\mathcal{K}$ is *consistent* (resp., *inconsistent*) iff the knowledge base (D, Σ) is consistent (resp., inconsistent). The set of all repairs of $\mathcal{K}$, denoted $Rep(\mathcal{K})$, is the set of all repairs of (D, Σ). An example of prioritized knowledge base $\mathcal{K} = (D, \Sigma, \Pi)$ is the one presented in Examples 1–2.

Semantics. Consider a prioritized knowledge base $\mathcal{K} = (D, \Sigma, \Pi)$. We use $\mathbf{C}_{\mathcal{K}}$ to denote the set of all constants appearing in $\mathcal{K}$ (that is, appearing in D, Σ, or Π). A *ground instance* of a preference rule $\rho \in \Pi$ is a ground preference rule derived from ρ by replacing every non-existential variable with a constant in $\mathbf{C}_{\mathcal{K}}$, with multiple occurrences of the same variable being replaced with the same constant. For instance, for the preference rule $p(X, Y) \succ q(X, Y) \leftarrow b(X)$, the occurrences of X in the body and in the head must be replaced with the same constant; likewise, the two occurrences of Y in the head must be replaced with the same constant, as the intended meaning of the preference rule is to prefer a p-fact for a pair of values over a q-fact for the same pair of values. We use $grnd(\rho)$ to denote the set of all ground instances of ρ, and define $grnd(\Pi) = \bigcup_{\rho \in \Pi} grnd(\rho)$.

Notice that ground instances are derived by considering only constants in $\mathbf{C}_{\mathcal{K}}$, rather than $\mathbf{C}$ (or $\mathbf{C} \cup \mathbf{N}$). Thus, constants that are not in $\mathbf{C}_{\mathcal{K}}$ and nulls do not appear in ground instances. As discussed in Example 3 (and formally defined later), to check whether a repair satisfies a ground preference rule, we need to check entailment of the body and of the head atoms. Since atoms containing at least one constant not in $\mathbf{C}_{\mathcal{K}}$ or containing at least one null are never entailed, we consider the grounding w.r.t. $\mathbf{C}_{\mathcal{K}}$ as a meaningful one.

To define preferred repairs, we first need to define when a repair satisfies a ground preference rule.

As discussed in Example 3, R satisfies ρ iff whenever R (together with the ontology) entails the body of ρ, the head of ρ is fulfilled by R. We formally define these notions below.

For a prioritized knowledge base $\mathcal{K} = (D, \Sigma, \Pi)$, by consistent subset R of D we mean a set $R \subseteq D$ such that (R, Σ) is consistent.

Definition 7. Let $\mathcal{K} = (D, \Sigma, \Pi)$ be a prioritized knowledge base, R a consistent subset of D, and ρ a ground preference rule in $grnd(\Pi)$ of the form $p(\mathbf{a}) \succ q(\mathbf{b}) \leftarrow \exists \mathbf{Z}\, \varphi(\mathbf{Z}, \mathbf{c})$.

We say that R *fulfills* $p(\mathbf{a}) \succ q(\mathbf{b})$ (w.r.t. $\mathcal{K}$) iff

$$(R, \Sigma) \not\models q(\mathbf{b}) \text{ or } (R, \Sigma) \models p(\mathbf{a}).$$

We say that R *satisfies* ρ (w.r.t. $\mathcal{K}$) iff $(R, \Sigma) \models \exists \mathbf{Z}\, \varphi(\mathbf{Z}, \mathbf{c})$ implies that R fullfils $p(\mathbf{a}) \succ q(\mathbf{b})$ w.r.t. $\mathcal{K}$. □

Notice that we do not transitively close preferences. In contrast, we look for an explicit preference rule saying that $p(\mathbf{a})$ is preferred over $q(\mathbf{b})$. A transitive closure would require (iteratively) adding a ground preference rule $A \succ C \leftarrow body_1, body_2$ for each pair of ground preference rules $A \succ B \leftarrow body_1$ and $B \succ C \leftarrow body_2$, which can yield an exponential blow-up in the number of preference rules. Nonetheless, if needed, transitivity can still be stated by explicitly including a transitive closure in Π.

Example 8. Consider the prioritized knowledge base $\mathcal{K} = (D, \Sigma, \Pi)$ of Examples 1–2 and the (ground) preference rule ρ_3' of Example 3.

Repair R_1 satisfies ρ_3', since

1. $(R_1, \Sigma) \models \exists Y \mathsf{hasDrink}(\mathsf{o}), \mathsf{side}(\mathsf{o}, Y)$, and
2. R_1 fulfills $\mathsf{dessert}(\mathsf{o}, \mathsf{pie}) \succ \mathsf{dessert}(\mathsf{o}, \mathsf{cake})$, since $(R_1, \Sigma) \models \mathsf{dessert}(\mathsf{o}, \mathsf{pie})$.

In contrast, repair R_2 does not satisfy ρ_3', since

1. $(R_2, \Sigma) \models \exists Y \mathsf{hasDrink}(\mathsf{o}), \mathsf{side}(\mathsf{o}, Y)$, and
2. R_2 does not fulfill $\mathsf{dessert}(\mathsf{o}, \mathsf{pie}) \succ \mathsf{dessert}(\mathsf{o}, \mathsf{cake})$, as $(R_2, \Sigma) \models \mathsf{dessert}(\mathsf{o}, \mathsf{cake})$, and $(R_2, \Sigma) \not\models \mathsf{dessert}(\mathsf{o}, \mathsf{pie})$. □

For a repair R of a prioritized knowledge base $\mathcal{K} = (D, \Sigma, \Pi)$, we use $\mathcal{S}(R, \mathcal{K})$ to denote the set of all preference rules in $grnd(\Pi)$ that are satisfied by R (w.r.t. $\mathcal{K}$). We define preferred repairs as follows.

Definition 9. Let $\mathcal{K}$ be a prioritized knowledge base. A repair R of $\mathcal{K}$ is *preferred* iff there is no repair R' of $\mathcal{K}$ s.t. $\mathcal{S}(R, \mathcal{K}) \subsetneq \mathcal{S}(R', \mathcal{K})$. The set of all preferred repairs of $\mathcal{K}$ is denoted as $PRep(\mathcal{K})$. □

Thus, preferred repairs satisfy an inclusion-maximal set of ground preference rules. Other criteria are possible, such as defining preferred repairs as those that satisfy a cardinality-maximal set of preference rules, or satisfying at least k preference rules, for some $k > 0$. All such approaches are interesting, in much the same way as cardinality and set-inclusion maximality are both interesting when defining (standard) repairs. We plan to investigate other criteria in future work.

Example 10. Consider the prioritized knowledge base $\mathcal{K} = (D, \Sigma, \Pi)$ and repairs R_1–R_4 of Examples 1–2.

First of all, note that only ground preference rules in $grnd(\Pi)$ whose body contains order o can be differently satisfied by R_1–R_4, as the body of any other ground preference rule in $grnd(\Pi)$ is never entailed by (R_i, Σ), for each $i \in \{1, 2, 3, 4\}$. Thus, such preference rules are satisfied by all repairs. Hence, we can focus on the following three ground preference rules:

$$\begin{aligned}
&\rho_1' : \mathsf{hasWine}(\mathsf{o}) \succ \mathsf{hasBeer}(\mathsf{o}) \leftarrow \mathsf{hasMeat}(\mathsf{o}),\\
&\rho_2' : \mathsf{dessert}(\mathsf{o}, \mathsf{cake}) \succ \mathsf{dessert}(\mathsf{o}, \mathsf{cherries}) \leftarrow \mathsf{main}(\mathsf{o}, \mathsf{beef}),\\
&\rho_3' : \mathsf{dessert}(\mathsf{o}, \mathsf{pie}) \succ \mathsf{dessert}(\mathsf{o}, \mathsf{cake}) \leftarrow\\
&\qquad\qquad \exists Y \mathsf{hasDrink}(\mathsf{o}), \mathsf{side}(\mathsf{o}, Y).
\end{aligned}$$

Notice that ρ_1' is applicable in all repairs. Repairs R_1 and R_2 satisfy ρ_1' because they both contain red wine; R_4 satisfies ρ_1' because it does not include any wine or beer; R_3 does not satisfy ρ_1' because it includes beer and no wine.

The preference rule ρ_2' is applicable in all repairs as well, and it is satisfied by all of them, essentially because none of them contain $\mathsf{dessert}(\mathsf{o}, \mathsf{cherries})$.

Finally, as discussed in Example 3, R_1, R_3, and R_4 satisfy ρ_3', while R_2 does not.

Thus, R_1 and R_4 satisfy all preference rules, while R_2 and R_3 do not, and thus R_1 and R_4 are preferred. □

Finally, we can easily generalize the inconsistency-tolerant semantics to the case of prioritized knowledge bases, where only the preferred repairs are considered for query answering.

Definition 11. Consider a prioritized knowledge base $\mathcal{K} = (D, \Sigma, \Pi)$ and a BCQ Q.

- $\mathcal{K}$ entails Q under the *Preferred ABox repair* (PAR) *semantics* if $(R, \Sigma) \models Q$ for every $R \in PRep(\mathcal{K})$.
- $\mathcal{K}$ entails Q under the *intersection of preferred repairs* (IPAR) *semantics* if $(D_I, \Sigma) \models Q$, where $D_I = \bigcap\{R \mid R \in PRep(\mathcal{K})\}$.
- $\mathcal{K}$ entails Q under the *intersection of closed preferred repairs* (ICPR) *semantics* if $(D_C, \Sigma) \models Q$, where $D_C = \bigcap\{Cn((R, \Sigma)) \mid R \in PRep(\mathcal{K})\}$. □

Example 12. Consider again the prioritized knowledge base $\mathcal{K}$ and the query Q from Examples 1–2. The preferred repairs of $\mathcal{K}$ are $PRep(\mathcal{K}) = \{R_1, R_4\}$ (cf. Example 10). Then, $\mathcal{K}$ entails Q under the PAR, IPAR, and ICPR semantics, while Q is not entailed under any of the standard inconsistency-tolerant semantics. □

$\mathcal{L}_\bot$	PRC		PAR		IPAR		ICPR	
	Data	Combined	Data	Combined	Data	Combined	Data	Combined
$\mathsf{L}_\bot$, $\mathsf{LF}_\bot$, $\mathsf{AF}_\bot$	coNP	PSPACE	Π^p_2	PSPACE	Π^p_2	PSPACE	Π^p_2	PSPACE
$\mathsf{SF}_\bot$	coNP	EXP	Π^p_2	EXP	Π^p_2	EXP	Π^p_2	EXP
$\mathsf{F}_\bot$, $\mathsf{GF}_\bot$	coNP	EXP	Π^p_2	EXP	Π^p_2	EXP	Π^p_2	EXP
$\mathsf{WA}_\bot$	coNP	2EXP	Π^p_2	2EXP	Π^p_2	2EXP	Π^p_2	2EXP
$\mathsf{S}_\bot$	coNP	EXP	Π^p_2	EXP	Π^p_2	EXP	Π^p_2	EXP- 2EXP
$\mathsf{A}_\bot$	coNP	P^{NEXP}	Π^p_2	P^{NEXP}	Π^p_2	P^{NEXP}	Π^p_2	P^{NEXP}- EXP^{NEXP}
$\mathsf{G}_\bot$, $\mathsf{WS}_\bot$	coNP	2EXP	Π^p_2	2EXP	Π^p_2	2EXP	Π^p_2	2EXP- 3EXP

Table 1: Data and combined complexity of preferred repair checking and preferred AR, IAR, and ICR entailment.

4 Complexity Results

In this section, we study the data and combined complexity of the "preferred" variant of different classical problems in the context of querying inconsistent knowledge bases.

The first problem asks whether a set of facts is a preferred repair of a prioritized knowledge base.

Problem: PREFERRED REPAIR CHECKING
Input: A prioritized knowledge base $\mathcal{K}$ and a database D'.
Output: Is D' a preferred repair of $\mathcal{K}$?

The remaining family of problems asks whether a query is entailed by a prioritized knowledge base under the PAR (resp., IPAR, ICPR) semantics.

Problem: S-ENTAIL, with $S \in \{\text{PAR, IPAR, ICPR}\}$.
Input: A prioritized knowledge base $\mathcal{K}$ and a BCQ Q.
Output: Does $\mathcal{K}$ entail Q under the S semantics?

Our results are reported in Table 1. A single complexity class in a cell refers to a completeness result, while two classes C_1-C_2 refer to C_1-hardness and C_2-membership.

In the data complexity, the use of preference rules incurs an increase of complexity w.r.t. the standard setting with no preferences. In fact, the data complexity goes from membership in P to coNP-completeness for repair checking, when moving from the standard setting to the preferred one. For the AR and ICR semantics, the data complexity goes from coNP-completeness of the standard setting to Π^p_2-completeness of our preferred framework. For the IAR semantics, the data complexity increases from membership in AC^0 or coNP-completeness (depending on the language) to Π^p_2-completeness. In the combined complexity, moving from the classical inconsistency-tolerant semantics to the preferred ones does not yield an increase of complexity, except for the languages $\mathsf{S}_\bot$, $\mathsf{A}_\bot$, $\mathsf{G}_\bot$, and $\mathsf{WS}_\bot$ under the ICPR semantics. Indeed, in such cases we do not have completeness results. Furthermore, except for $\mathsf{S}_\bot$, $\mathsf{A}_\bot$, $\mathsf{G}_\bot$, and $\mathsf{WS}_\bot$, each language exhibits the same complexity under all semantics.

In the following, we use $\mathcal{L}_\bot$ to refer to any of the languages considered in this paper and use $\mathcal{L}$ to refer to the language without negative constraints.

Preferred Repair Checking. We start with the data complexity of preferred repair checking. The following theorem shows that preferred repair checking is coNP-complete in the data complexity for all languages we consider. The hardness result is proved via a reduction from 2+2-CNF UNSATISFIABILITY.

Theorem 13. PREFERRED REPAIR CHECKING *is* coNP-complete *in the data complexity for all languages we consider.*

We now turn our attention to the combined complexity. We first focus on the upper bounds and provide a generic procedure that solves the complement of PREFERRED REPAIR CHECKING using oracles for the following two key problems.

The first problem is (classical) REPAIR CHECKING, that is, given a knowledge base (D, Σ) and a database D', decide whether D' is a repair of (D, Σ).

The second problem, named SATISFACTION, is defined as follows: given a prioritized knowledge base $\mathcal{K}$, a repair R of $\mathcal{K}$, and a ground preference rule ρ, decide whether R satisfies ρ w.r.t. $\mathcal{K}$.

The following proposition provides membership results for these two problems in the combined complexity.

Proposition 14. *For any language $\mathcal{L}_\bot$ considered here, if BCQ entailment for $\mathcal{L}$ is in* C *in the combined complexity, then* REPAIR CHECKING *(resp.* SATISFACTION*) for $\mathcal{L}_\bot$ is*

- *in* C *if* C *is deterministic and* C $\supseteq$ PSPACE*,*
- *in* P^{C} *if* C $=$ NEXP*,*

in the combined complexity.

The previous proposition is leveraged to derive membership results for PREFERRED REPAIR CHECKING in the combined complexity as follows.

Theorem 15. *For any language $\mathcal{L}_\bot$ considered here, if* REPAIR CHECKING *and* SATISFACTION *for $\mathcal{L}_\bot$ are in* C *in the combined complexity, then* PREFERRED REPAIR CHECKING *for $\mathcal{L}_\bot$ is in* C *in the combined complexity if* C *is deterministic and* C $\supseteq$ PSPACE*.*

The upper bounds of PREFERRED REPAIR CHECKING in the combined complexity of Table 1 follow from Proposition 14, Theorem 15, and the complexity results of (Lukasiewicz et al. 2015a) on the combined complexity of BCQ entailment.

Regarding the lower bounds, we can reduce BCQ entailment to the complement of PREFERRED REPAIR CHECKING, obtaining tight lower bounds for all languages we consider but $\mathsf{A}_\bot$, for which we will need a dedicated reduction.

Theorem 16. *For any language $\mathcal{L}_\perp$ considered here, BCQ entailment for $\mathcal{L}$ is reducible in polynomial time to the complement of* PREFERRED REPAIR CHECKING *for $\mathcal{L}_\perp$ in the combined complexity.*

Theorem 16 provides only a coNEXP-hardness for $\mathsf{A}_\perp$, not matching the $\mathrm{P}^{\mathrm{NExp}}$ upper bound. Below we show that the problem is actually $\mathrm{P}^{\mathrm{NExp}}$-hard in the combined complexity via a reduction from the $\mathrm{P}^{\mathrm{NExp}}$-complete extended tiling problem defined in (Eiter, Lukasiewicz, and Predoiu 2016).

Theorem 17. PREFERRED REPAIR CHECKING *for $\mathsf{A}_\perp$ is $\mathrm{P}^{\mathrm{NExp}}$-hard in the combined complexity.*

PAR Entailment. We now move to the PAR-ENTAIL problem, whose data complexity is show in the theorem below. The hardness result is shown via a reduction from the problem of deciding validity of a quantified Boolean formula.

Theorem 18. PAR-ENTAIL *is* Π_2^{p}-complete *in the data complexity for all languages we consider.*

We now focus on the combined complexity. To show the upper bounds, we provide a polynomial time nondeterministic procedure that exploits PREFERRED REPAIR CHECKING as an oracle to solve the complement of PAR-ENTAIL.

Theorem 19. *For any language $\mathcal{L}_\perp$ considered here, if* PREFERRED REPAIR CHECKING *for $\mathcal{L}_\perp$ is in* C *in the combined complexity, then* PAR-ENTAIL *for $\mathcal{L}_\perp$ is in* $\mathrm{coNP}^{\mathrm{C}}$ *in the combined complexity.*

Knowing that coC $=$ C for any deterministic class C, and $\mathrm{NP}^{\mathrm{C}} = \mathrm{C}$ for any deterministic class C $\supseteq$ PSPACE, we can combine the above general theorem and the combined complexity of PREFERRED REPAIR CHECKING to provide upper bounds matching the lower bounds inherited from the AR semantics (which is generalized by the PAR semantics).

IPAR Entailment. We now turn our attention to the IPAR-ENTAIL problem, whose data complexity is as follows.

Theorem 20. IPAR-ENTAIL *is* Π_2^{p}-complete *in the data complexity for all languages we consider.*

As for the combined complexity, with a similar approach to the one used for the PAR semantics, the following theorem provides upper bounds matching the lower bounds inherited from the combined complexity of the standard IAR semantics (Lukasiewicz, Malizia, and Molinaro 2018).

Theorem 21. *For any language $\mathcal{L}_\perp$ considered here, if* PREFERRED REPAIR CHECKING *for $\mathcal{L}_\perp$ is in* C *in the combined complexity, then* IPAR-ENTAIL *for $\mathcal{L}_\perp$ is in* $\mathrm{coNP}^{\mathrm{C}}$ *in the combined complexity.*

ICPR Entailment. The last problem we analyze is ICPR-ENTAIL, whose data complexity is as follows.

Theorem 22. ICPR-ENTAIL *is* Π_2^{p}-complete *in the data complexity for all languages we consider.*

As for the combined complexity, we start with a theorem providing upper bounds via a generic procedure. The procedure exploits the fact that for a prioritized knowledge base $\mathcal{K} = (D, \Sigma, \Pi)$, the database $D_C = \bigcap\{Cn((R, \Sigma)) \mid R \in PRep(\mathcal{K})\}$ is the set of all facts $p(\mathbf{a})$ that are entailed by $\mathcal{K}$ under the PAR semantics. So, after constructing D_C by making exponentially many entailment checks under the PAR semantics, we can check whether $(D_C, \Sigma) \models Q$ via a call to an oracle for BCQ entailment (which is of lower complexity w.r.t. PAR-ENTAIL).

Proposition 23. *For any language $\mathcal{L}_\perp$ considered here, if* PAR-ENTAIL *for $\mathcal{L}_\perp$ is in* C *in the combined complexity, then* ICPR-ENTAIL *for $\mathcal{L}_\perp$ is in* $\mathrm{EXP}^{\mathrm{C}}$ *in the combined complexity.*

The above upper bound is not tight w.r.t. the lower bounds inherited from classical ICR entailment. To obtain tight upper bounds, we need to devise more refined procedures exploiting some key properties of the underlying languages.

For the languages $\mathsf{L}_\perp$ and $\mathsf{LF}_\perp$, we exploit the fact that under linear TGDs, checking whether a query Q is entailed by a knowledge base (D, Σ) requires to consider only $|Q|$ facts of the database. That is, $(D, \Sigma) \models Q$ iff there is $D_Q \subseteq D$ such that $|D_Q| \leq |Q|$ and $(D_Q, \Sigma) \models Q$ (Gottlob et al. 2014, Theorem 5). In the case of $\mathsf{AF}_\perp$, we rely on the fact that for BCQ entailment, only a constant number (w.r.t. the database) of facts needs to be kept in memory.

Theorem 24. ICPR-ENTAIL *for* $\mathsf{L}_\perp$, $\mathsf{LF}_\perp$, *and* $\mathsf{AF}_\perp$*is in* PSPACE *in the combined complexity.*

The above upper bounds match the lower bounds inherited from (Lukasiewicz, Malizia, and Molinaro 2018).

For $\mathsf{F}_\perp$, $\mathsf{GF}_\perp$, and $\mathsf{SF}_\perp$. we rely on the fact that the EXP procedure for BCQ entailment of a query Q in a knowledge base (D, Σ) runs in time $O(k_1 \cdot ||D||^{k_2})$, where $||D||$ is the size of D, and k_1, k_2 are terms that only depend on Σ and the query Q.

Theorem 25. ICPR-ENTAIL *for* $\mathsf{F}_\perp, \mathsf{GF}_\perp$, *and* $\mathsf{SF}_\perp$ *is in* EXP *in the combined complexity.*

The above upper bounds match the lower bounds inherited from (Lukasiewicz, Malizia, and Molinaro 2018).

Regarding the class of weakly-acyclic TGDs, i.e., $\mathsf{WA}_\perp$, a consequence of (Fagin et al. 2005, Proof of Theorem 3.9) is that the running time of the double exponential time procedure for BCQ entailment is again of the form $O(k_1 \cdot ||D||^{k_2})$, where k_1, k_2 only depend on Σ and the query Q. Hence, with an argument similar to that of Theorem 25, we obtain the following result.

Theorem 26. ICPR-ENTAIL *for* $\mathsf{WA}_\perp$ *is in* 2EXP *in the combined complexity.*

The above upper bound matches the lower bound inherited from (Lukasiewicz, Malizia, and Molinaro 2018).

Despite our efforts, for the remaining languages we were not able to derive upper bounds tighter than those provided by Proposition 23, which are $\mathrm{EXP}^{\mathrm{EXP}}$ for $\mathsf{S}_\perp$, yielding a 2EXP upper bound; $\mathrm{EXP}^{\mathrm{NExp}}$ for $\mathsf{A}_\perp$; $\mathrm{EXP}^{2\mathrm{EXP}}$ for $\mathsf{G}_\perp$ and $\mathsf{WS}_\perp$, which leads to a 3EXP upper bound. We plan to close these open problems in future work.

5 Related Work

The AR, IAR, and ICR semantics have been extensively investigated for DLs—see, e.g., (Lembo et al. 2010; Bienvenu 2012; Bienvenu and Rosati 2013; Bienvenu, Bourgaux, and Goasdoué 2014b; Lembo et al. 2015; Bienvenu and Bourgaux 2016)—and under existential rules—see, e.g., (Lukasiewicz, Martinez, and Simari 2012; Lukasiewicz, Martinez, and Simari 2013a; Lukasiewicz et al. 2015a; Eiter, Lukasiewicz, and Predoiu 2016; Lukasiewicz, Malizia, and Molinaro 2018; Lukasiewicz, Malizia, and Vaicenavičius 2019).

For DLs, Bienvenu, Bourgaux, and Goasdoué (2014b) defined variants of the AR and IAR semantics where classical repairs are replaced by various types of preferred repairs.

One criterion they consider is to define preferred repairs as cardinality-maximal ones. This approach is quite different from ours. In fact, the preference criterion is fixed a priori between repairs (maximum cardinality), and there is no preference expressed between individual facts (thus, all facts are equally important). However, in our framework, if no preference between facts is expressed, then all repairs are preferred (including those that are not cardinality-maximal but are only inclusion-maximal).

Another criterion is $\leq_w$, where weights are assigned to facts, and preferred repairs are those with the highest overall weight. One natural way of mapping $\leq_w$ into our framework is to introduce a preference rule $a \succ b \leftarrow$ whenever the weight of a is greater than the weight of b. However, the preferred repairs returned by the two approaches are in general different. As an example, consider the database D_1 containing facts a, b, and c, whose weights are 3, 2, and 2, respectively. Suppose the following set of constraints is defined: $\Sigma_1 = \{a, b \rightarrow \bot, a, c \rightarrow \bot\}$. Then, the repairs are $\{a\}$ and $\{b, c\}$, whose weights are 3 and 4, respectively. Thus, $\{b, c\}$ is the only preferred repair w.r.t. $\leq_w$. The preferences induced by the aforementioned weights would be expressed in our framework via preference rules $a \succ b \leftarrow$ and $a \succ c \leftarrow$, which give $\{a\}$ as the only preferred repair.

Bienvenu, Bourgaux, and Goasdoué (2014b) defined two additional criteria based on priority levels, denoted as $\subseteq_P$ and $\leq_P$. Under such criteria, a *prioritization* is provided, that is, the database is partitioned into priority levels $P_1, \ldots P_n$, with facts in P_1 considered the most reliable, and those in P_n the least reliable. Then, either the set-inclusion or the cardinality criterion is applied to each level. Such approaches are somehow closer to ours. One natural way of expressing a prioritization in our framework is to introduce a preference rule $a \succ b \leftarrow$ for each pair of facts a and b such that $a \in P_i$ and $b \in P_{i+1}$ for some i. This approach expresses preferences between adjacent levels only. However, the two approaches do not provide the same set of preferred repairs. As an example, consider the database $D_2 = \{a, b, c, d\}$ and the constraints $\Sigma_2 = \{a, b \rightarrow \bot,\ a, c \rightarrow \bot,\ d, b \rightarrow \bot,\ d, c \rightarrow \bot\}$. The repairs are $R_1 = \{a, d\}$ and $R_2 = \{b, c\}$. Consider the prioritization $P_1 = \{a\}$, $P_2 = \{b\}$, $P_3 = \{c\}$, and $P_4 = \{d\}$. Then, R_1 is the only preferred repair according to both $\subseteq_P$ and $\leq_P$. The aforementioned translation of the prioritization into preference rules yields $a \succ b \leftarrow$, $b \succ c \leftarrow$, and $c \succ d \leftarrow$. Then, in our framework both R_1 and R_2 are preferred.

An alternative translation is the one where preferences are expressed between every pair of levels, not only adjacent ones. More precisely, a prioritization might be translated into our framework by introducing a preference rule $a \succ b \leftarrow$ for each pair of facts a and b such that $a \in P_i$ and $b \in P_j$ with $i < j$. Also in this case the preferred repairs do not coincide. As an example, consider $D_3 = \{a, b, c, d, e\}$ and $\Sigma_3 = \{c, d \rightarrow \bot, e, d \rightarrow \bot\}$. There are two repairs $R_1 = \{a, b, c, e\}$ and $R_2 = \{a, b, d\}$. Consider now the prioritization $P_1 = \{a\}$, $P_2 = \{b, c\}$, $P_3 = \{d\}$, and $P_4 = \{e\}$. Repair R_1 is the only preferred one w.r.t. both $\subseteq_P$ and $\leq_P$. By translating the prioritization into preference rules we get $\Pi_3 = \{a \succ b \leftarrow, a \succ c \leftarrow, a \succ d \leftarrow, a \succ e \leftarrow, b \succ d \leftarrow, b \succ e \leftarrow, c \succ d \leftarrow, c \succ e \leftarrow, d \succ e \leftarrow\}$. It can be easily verified that in our framework both R_1 and R_2 are preferred repairs.

Even if we restrict our framework to preference rules with empty body, none of the criteria discussed above allow us to express arbitrary preferences between any pair of facts, as we do, such as cyclic preference relations or preferences between facts that are not in the database but are entailed from the database via the ontology. Also, we allow preconditions to be stated. Another difference is that Bienvenu, Bourgaux, and Goasdoué (2014b) consider DLs (in particular, they focus on DL-Lite$_\mathcal{R}$), while we consider existential rule languages. Finally, besides the AR and IAR semantics, we additionally consider the ICR semantics.

Staworko, Chomicki, and Marcinkowski (2012) introduced a framework where the AR semantics (a.k.a. *consistent query answering*) is generalized to take into account a priority relation $\succ$ expressing preferences among facts in the database. Such preferences are then used to define three notions of preferred repairs: *global-*, *Pareto-*, and *completion-optimal* repairs. Different computational problems of this framework have been recently investigated. In particular, for functional dependencies, Fagin, Kimelfeld, and Kolaitis (2015) showed dichotomies for the preferred repair checking problem, Livshits and Kimelfeld (2017) addressed the problem of counting and enumerating preferred repairs, while Kimelfeld, Livshits, and Peterfreund (2017) studied the problem of deciding whether there exists exactly one preferred repair.

Clearly, preferences among facts can be expressed in our framework via preference rules with empty body. In the presence of preferences of this form, one interesting question is whether the two frameworks give the same preferred repairs. In general, this is not the case. As an example, consider the database $D = \{a, b, c\}$, the constraints $a, b \rightarrow \bot$ and $a, c \rightarrow \bot$, and the priority $a \succ b$. The only repairs are $\{a\}$ and $\{b, c\}$, and both are global-, Pareto-, and completion-optimal. However, considering the preference rule $a \succ b \leftarrow$ in our framework, $\{a\}$ is the only preferred repair.

Staworko, Chomicki, and Marcinkowski (2012) defined desirable properties for families of preferred repairs, namely *Non-emptiness* (i.e., the set of preferred repairs is always non-empty), *Monotonicity* (i.e., adding preferences can only narrow the set of preferred repairs), *Non-discrimination* (i.e., the set of preferred repairs coincides with the set of repairs when no preference is expressed), *Categoricity* (i.e., when the preference relation is total there is a single preferred repair), and *Conservativeness* (i.e., preferred repairs are a subset of all repairs). Different preference-based frameworks in the literature satisfy different subsets of such properties—we refer to (Staworko, Chomicki, and Marcinkowski 2012) for a thorough discussion. In our framework, it is straightforward to see that Non-emptiness, Non-discrimination, and Conservativeness are satisfied. On the other hand, Monotonicity and Categoricity are not satisfied. As an example, consider the knowledge base (D_2, Σ_2), where D_2 and Σ_2 are the ones previously introduced. Recall that the repairs are $\{a, d\}$ and $\{b, c\}$. Consider now the preference rule $a \succ b \leftarrow$. In our framework, the only preferred repair is $\{a, d\}$. By introducing the additional preference rule $c \succ d \leftarrow$, both repairs are preferred, thus violating monotonicity. Satisfaction of Monotonicity highly depends on how preferences are used to determine preferred repairs. In the example above, $a \succ b \leftarrow$ is satisfied by the first repair but not by the second one, which gives a reason to prefer the former over the latter in our framework. When there is no reason (i.e., no other preference) to prefer the second repair over the first one, the first repair is the only preferred one. However, the addition of $c \succ d \leftarrow$ provides a reason to prefer the second repair over the first one, which makes them incomparable in our framework, and both end up being preferred. Thus, in our approach, Monotonicity is not as meaningful/relevant as in (Staworko, Chomicki, and Marcinkowski 2012), where preferences are interpreted in a different way. Categoricity also depends on how preferences between facts are used to determine preferred repairs. Consider the knowledge base (D_3, Σ_3) and the set of preference rules Π_3 previously introduced. By adding $b \succ c \leftarrow$ to Π_3, the preference relation is total. However, both repairs of (D_3, Σ_3) are preferred.

Staworko, Chomicki, and Marcinkowski (2012) focus on *acyclic* preference relations where a preference can only involve two facts violating the same integrity constraint. In contrast, we do not impose any restriction on preferences, which can thus be cyclic and involve facts that do not violate any integrity constraint or violating different integrity constraints. Also, our preference rules can express preferences between facts derived via the ontology. Furthermore, they deal with denial constraints, while we consider existential rules and negative constraints. Also, we allow users to express preconditions for preferences to hold, which cannot be done in their framework. Finally, besides the AR semantics, we additionally consider the IAR and ICR semantics.

Another framework related to ours are *active integrity constraints* (AICs) (Caroprese, Greco, and Zumpano 2009). An AIC specifies an integrity constraint along with the updates that are allowed to restore consistency when the constraint is violated. *Founded* repairs are those repairs obtained by applying only allowed updates. Consistent query answering is then (re)defined by looking only at founded repairs. The framework thus expresses a sort of hard preference by partitioning conflicting facts into allowed and non-allowed ones—to the point that in some cases there does not exist any founded repair (while in our framework the set of preferred repairs is always non-empty). Our framework allows us to express preferences in a different way, as we can express preferences among facts that do not violate integrity constraints or violating different integrity constraints, and such preferences do not forbid any fact a priori. Also, AICs can express denial constraints, while we consider ontologies consisting of existential rules and negative constraints.

Querying *consistent* knowledge bases in the presence of user preferences has been investigated for both DLs (Ceylan et al. 2017; Ceylan, Lukasiewicz, and Peñaloza 2015) and existential rule languages (Fazzinga et al. 2018; Lukasiewicz et al. 2015b; Lukasiewicz et al. 2014; Lukasiewicz, Martinez, and Simari 2013b; Lukasiewicz et al. 2013). The problem has been investigated also for relational data by Stefanidis, Pitoura, and Vassiliadis (2011). Specifically, preferences are annotated with context states specifying under which conditions a preference holds. Each query is also associated with a set of context states. The problem is then to select the best matching tuples taking into account preferences, as well as the context states of both the data and the query. One important aspect that our work and the one by Stefanidis, Pitoura, and Vassiliadis (2011) share is the capability of expressing conditions under which preferences hold. The main difference between this paper and the aforementioned ones is that we deal with inconsistency.

More specifically, they lift preferences to *query answers* for different purposes, e.g., to rank and compute top-k query answers. In our case, query answers are not ranked. Preferences are rather lifted to *repairs* in order to identify preferred ones.

To the best of our knowledge, this is the first paper addressing the problem of querying inconsistent knowledge bases *under existential rules in the presence of user preferences*.

We conclude by mentioning that there has been work in logic programming with a similar spirit, in that logic programs are combined with preferences so as to determine a set of *preferred answer sets*—e.g., see (Sakama and Inoue 2000; Brewka, Niemelä, and Truszczynski 2003; Greco, Trubitsyna, and Zumpano 2007)— while we use preferences to determine a set of preferred repairs. Despite such a difference in the underlying settings, at a higher level of abstraction both scenarios tackle the problem of expressing preferences among sets of facts (answer sets or repairs).

6 Conclusion

We have proposed a framework for querying inconsistent knowledge bases under existential rules in the presence of user preferences. We have analyzed the data and combined complexity of different relevant problems for a wide range of existential rule languages.

An interesting direction for future work is to extend our analysis to the *fixed-program combined complexity*, that is, when the ontology is assumed to be fixed, and to the *bounded-arity combined complexity*, that is, when it is assumed that the maximum arity of the predicates is bounded by an integer constant. Another direction for future work is to apply preferences rules to different kinds of repairs, e.g., cardinality-maximal ones (recall that we relied on the standard inclusion-maximal notion of repair). Also, it would be interesting to apply the framework of Staworko, Chomicki, and Marcinkowski (2012) to existential rule languages.

Acknowledgments

We thank the anonymous referees for many constructive comments and suggestions that helped improve the paper.

References

Agrawal, R.; Rantzau, R.; and Terzi, E. 2006. Context-sensitive ranking. In *Proc. SIGMOD*, 383–394.

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

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

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

Bienvenu, M.; 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. In *Proc. AAAI*, 996–1002.

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

Brewka, G.; Niemelä, I.; and Truszczynski, M. 2003. Answer set optimization. In *Proc. IJCAI*, 867–872.

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.

Caroprese, L.; Greco, S.; and Zumpano, E. 2009. Active integrity constraints for database consistency maintenance. *IEEE Trans. Knowl. Data Eng.* 21(7):1042–1058.

Ceylan, İ. İ.; Lukasiewicz, T.; Peñaloza, R.; and Tifrea-Marciuska, O. 2017. Query answering in ontologies under preference rankings. In *Proc. IJCAI*, 943–949.

Ceylan, İ. İ.; Lukasiewicz, T.; and Peñaloza, R. 2015. Answering EL queries in the presence of preferences. In *Proc. DL*.

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.

Fagin, R.; Kimelfeld, B.; and Kolaitis, P. G. 2015. Dichotomies in the complexity of preferred repairs. In *Proc. PODS*, 3–15.

Fazzinga, B.; Lukasiewicz, T.; Martinez, M. V.; Simari, G. I.; and Tifrea-Marciuska, O. 2018. Ontological query answering under many-valued group preferences in datalog+/-. *Int. J. Approx. Reason.* 93:354–371.

Gottlob, G.; Kikot, S.; Kontchakov, R.; Podolskii, V.; Schwentick, T.; and Zakharyaschev, M. 2014. The price of query rewriting in ontology-based data access. *Artificial Intelligence* 213:42 – 59.

Greco, S.; Trubitsyna, I.; and Zumpano, E. 2007. On the semantics of logic programs with preferences. *J. Artif. Intell. Res.* 30:501–523.

Kimelfeld, B.; Livshits, E.; and Peterfreund, L. 2017. Detecting ambiguity in prioritized database repairing. In *Proc. ICDT*, 17:1–17:20.

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

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

Livshits, E., and Kimelfeld, B. 2017. Counting and enumerating (preferred) database repairs. In *Proc. PODS*, 289–301.

Lukasiewicz, T.; Martinez, M. V.; Simari, G. I.; and Tifrea-Marciuska, O. 2013. Group preferences for query answering in datalog+/- ontologies. In *Proc. SUM*, 360–373.

Lukasiewicz, T.; Martinez, M. V.; Simari, G. I.; and Tifrea-Marciuska, O. 2014. Ontology-based query answering with group preferences. *ACM Trans. Internet Techn.* 14(4):25:1–25:24.

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

Lukasiewicz, T.; Martinez, M. V.; Simari, G. I.; and Tifrea-Marciuska, O. 2015b. Preference-based query answering in probabilistic datalog+/- ontologies. *J. Data Semantics* 4(2):81–101.

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

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

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

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

Lukasiewicz, T.; Martinez, M. V.; and Simari, G. I. 2013b. Preference-based query answering in datalog+/- ontologies. In *Proc. IJCAI*, 1017–1023.

Sakama, C., and Inoue, K. 2000. Prioritized logic programming and its application to commonsense reasoning. *Artif. Intell.* 123(1-2):185–222.

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.

Stefanidis, K.; Pitoura, E.; and Vassiliadis, P. 2011. Managing contextual preferences. *Inf. Syst.* 36(8):1158–1180.

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