# Preference-based Inconsistency-Tolerant Query **Answering under Existential Rules**

(Discussion Paper)

Marco Calautti<sup>1</sup>, Sergio Greco<sup>2</sup>, Cristian Molinaro<sup>2</sup> and Irina Trubitsyna<sup>2</sup>

<sup>1</sup>DISI, University of Trento, Italy <sup>2</sup>DIMES, University of Calabria, Italy

#### 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, most of which 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 [1, 2] 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 [2] considers an answer valid if it can be inferred from the intersection of the repairs. The intersection of closed repairs (ICR) semantics [3] 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  $(D, \Sigma)$  describing orders in a restaurant, where D is the following database:

{meat(beef), order(o), main(o, beef), side(o, cheese),

drink(o, red), drink(o, beer), dessert(o, cake), dessert(o, pie)},

CEUR Workshop Proceedings (CEUR-WS.org)

SEBD 2021: The 29th Italian Symposium on Advanced Database Systems, September 5-9, 2021, Pizzo Calabro (VV), Italy marco.calautti@unitn (M. Calautti); greco@dimes.unical.it (S. Greco); cmolinaro@dimes.unical.it (C. Molinaro); trubitsyna@dimes.unical.it (I. Trubitsyna)

<sup>© 02021</sup> Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

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  $\Sigma$  contains the following dependencies:

 $\begin{array}{ll} \sigma_1\colon \mathrm{main}(X,Y), \mathrm{meat}(Y) \to \mathrm{has}\mathrm{Meat}(X), \\ \sigma_2\colon \mathrm{drink}(X,\mathrm{red}) \to \mathrm{has}\mathrm{Wine}(X), \\ \sigma_3\colon \mathrm{drink}(X,\mathrm{beer}) \to \mathrm{has}\mathrm{Beer}(X), \\ \sigma_4\colon \mathrm{drink}(X,Y) \to \mathrm{has}\mathrm{Drink}(X), \\ \sigma_5\colon \mathrm{main}(X,Y) \to \exists Z \operatorname{side}(X,Z), \\ \nu_1\colon \mathrm{has}\mathrm{Beer}(X), \mathrm{has}\mathrm{Wine}(X) & \to \bot, \\ \nu_2\colon \mathrm{has}\mathrm{Beer}(X), \mathrm{dessert}(X,Y) & \to \bot, \\ \nu_3\colon \mathrm{has}\mathrm{Beer}(X), \mathrm{side}(X,\mathrm{chese}) & \to \bot, \\ \nu_4\colon \mathrm{has}\mathrm{Wine}(X), \mathrm{dessert}(X,\mathrm{cake}), \mathrm{dessert}(X,\mathrm{pie}) & \to \bot. \end{array}$ 

The existential rules  $\sigma_1 - \sigma_4$  specify when an order includes a meat dish, wine, beer, and a drink, respectively. Then,  $\sigma_5$  says that a main course always comes with a side dish. The negative constraints  $\nu_1 - \nu_3$  say that beer cannot be included in an order together with wine, or a dessert, or cheese. Finally,  $\nu_4$  says that an order cannot include wine, cake, and pie all together.

The knowledge base is clearly inconsistent and admits the following four repairs:  $R_1 = D \setminus \{ \text{drink}(o, \text{beer}), \text{dessert}(o, \text{cake}) \}, R_2 = D \setminus \{ \text{drink}(o, \text{beer}), \text{dessert}(o, \text{pie}) \}, R_3 = D \setminus \{ \text{side}(o, \text{cheese}), \text{drink}(o, \text{red}), \text{dessert}(o, \text{pie}), \text{dessert}(o, \text{cake}) \}, and R_4 = D \setminus \{ \text{drink}(o, \text{red}), \text{drink}(o, \text{beer}) \}$ . The query  $Q = \exists X, Y \text{ dessert}(X, Y)$ , asking if 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.  $\Box$ 

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 deal with the scenarios discussed above, we propose a framework for querying inconsistent knowledge bases under user preferences. 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.<sup>1</sup>

**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  $\Pi$  of preference rules:

 $\rho_1$ : hasWine(X)  $\succ$  hasBeer(X)  $\leftarrow$  hasMeat(X),

- $\rho_2$ : dessert(X, cake) > dessert(X, cherries)  $\leftarrow main(X, Y)$ ,
- $\rho_3$ : dessert(X, pie) > dessert(X, cake) \leftarrow \exists Y hasDrink(X), side(X, Y).

<sup>&</sup>lt;sup>1</sup>An extended version of this paper has been accepted to KR 2020 [4].

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.

## 2. Preliminaries

We briefly recall some basics on existential rules from the Datalog<sup> $\pm$ </sup> family [5] and inconsistency-tolerant semantics for querying inconsistent knowledge bases.

**General.** We assume a set  $\mathbb{C}$  of *constants*, a set  $\mathbb{N}$  of *labeled nulls*, and a set  $\mathbb{V}$  of *variables*. A *term t* is a constant, null, or variable. We also assume a set of *predicates*, each associated with an *arity*, i.e., a non-negative integer. An *atom* has the form p(t), where p is an n-ary predicate, and 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(t), with t a tuple of constants and nulls. A *database* D is a finite instance with only constants. A *homomorphism* is a substitution h from terms to terms that is the identity on  $\mathbb{C}$  and maps  $\mathbb{N}$  to  $\mathbb{C} \cup \mathbb{N}$ . With abuse of notation, homomorphisms are applied also to (sets/conjunctions of) atoms. A *(Boolean) conjunctive query* (BCQ) Q has the form  $\exists \mathbf{X}\varphi(\mathbf{X})$ , where  $\mathbf{X}$  is a tuple of variables, and  $\varphi(\mathbf{X})$  is a conjunction of atoms over the variables in  $\mathbf{X}$  without nulls. An instance I satisfies Q, denoted  $I \models Q$ , if there is a homomorphism h with  $h(\varphi(\mathbf{X})) \subseteq I$ .<sup>2</sup>

**Dependencies.** A *tuple-generating dependency* (*TGD*)  $\sigma$  is a first-order formula of the form  $\forall \mathbf{X} \forall \mathbf{Y} \varphi(\mathbf{X}, \mathbf{Y}) \to \exists \mathbf{Z} p(\mathbf{X}, \mathbf{Z})$ , 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  $\sigma$ , denoted *body*( $\sigma$ ), while  $p(\mathbf{X}, \mathbf{Z})$  is the *head* of  $\sigma$ , 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  $\sigma$ , 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*)  $\nu$  is a first-order formula  $\forall \mathbf{X} \varphi(\mathbf{X}) \to \bot$ , with  $\mathbf{X}$  a tuple of variables,  $\varphi(\mathbf{X})$  is a conjunction of atoms over  $\mathbf{X}$ , without nulls, called the *body* of  $\nu$  and denoted *body*( $\nu$ ), and  $\bot$  denotes the truth constant *false*. An instance *I* satisfies  $\nu$ , written  $I \models \nu$ , if *I* does not satisfy the BCQ  $\exists \mathbf{X} \varphi(\mathbf{X})$ . Negative constraints that are not satisfied can be represented via hypergraphs [6].

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 for conjoining atoms. Given a class of TGDs  $\mathcal{L}$ , we denote by  $\mathcal{L}_{\perp}$  the formalism obtained

<sup>&</sup>lt;sup>2</sup>We focus on Boolean queries for clarity, but all our results immediately extend to non-Boolean conjunctive queries.

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  $(D, \Sigma)$ , where D is a database and  $\Sigma$  is a program. The set of models of  $KB = (D, \Sigma)$ , denoted mods(KB), is the set of instances  $\{I \mid I \supseteq D \land I \models \Sigma\}$ . We say that KB is consistent if  $mods(KB) \neq \emptyset$ , otherwise KB is inconsistent. We say that KB entails a BCQ Q, denoted  $KB \models Q$ , if  $M \models Q$ , for each  $M \in mods(KB)$ . The data complexity considers only the database as part of the input, while the combined complexity considers everything as part of the input.

The Datalog<sup> $\pm$ </sup> languages that we consider to guarantee decidability are among the most frequently analysed in the literature, namely, linear (L) [5], guarded (G) [7], sticky (S) [8], and acyclic TGDs (A), along with the "weak" (proper) generalizations weakly sticky (WS) [8] and weakly acyclic TGDs (WA) [9], 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 refer to [10] for an overview of the complexity of BCQ entailment for the above languages.

Inconsistency-Tolerant Semantics. We now recall 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  $(R, \Sigma)$  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  $\Sigma$ . KB entails a BCQ Q under the

- AR semantics if  $(R, \Sigma) \models Q$ , for each  $R \in Rep(KB)$ .
- *IAR semantics* if  $(D_I, \Sigma) \models Q$ , where  $D_I = \bigcap \{R \mid R \in Rep(KB)\}$ .
- ICR semantics if  $(D_C, \Sigma) \models Q$ , where  $D_C = \bigcap \{Cn((R, \Sigma)) \mid R \in Rep(KB)\}$ .

We refer to [10] and [11] for an overview of the complexity of AR- and IAR-/ICR-query answering, respectively, for different TGD 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.

Syntax. The syntax of preference rules is as follows.

**Definition 3.** A preference rule  $\rho$  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 over the variables  $\mathbf{X}$  and  $\mathbf{Y}$ , respectively, and  $\varphi(\mathbf{Z}, \mathbf{W})$  is a (possibly empty) conjunction of atoms over the variables  $\mathbf{Z} \cup \mathbf{W}$ , all without nulls.

In the previous definition, the right-hand (resp., left-hand) side of  $\leftarrow$  is called the *body* (resp., *head*) of  $\rho$  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. A preference rule  $\rho$  is *ground* if all its variables are existentially quantified, 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. A prioritized knowledge base combines a knowledge base with a preference program.

**Definition 4.** A prioritized knowledge base  $\mathcal{K}$  is a triple  $(D, \Sigma, \Pi)$ , where D is a database,  $\Sigma$  is a program, and  $\Pi$  is a preference program.

We say that  $\mathcal{K}$  is *consistent* (resp., *inconsistent*) iff the knowledge base  $(D, \Sigma)$  is consistent (resp., inconsistent). The set of all repairs of  $\mathcal{K}$ , denoted  $Rep(\mathcal{K})$ , is the set of all repairs of  $(D, \Sigma)$ . 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  $D, \Sigma$ , or  $\Pi$ ). A ground instance of a preference rule  $\rho \in \Pi$  is a ground preference rule derived from  $\rho$  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. We use  $grnd(\rho)$  to denote the set of all ground instances of  $\rho$ , and define  $grnd(\Pi) = \bigcup_{\rho \in \Pi} grnd(\rho)$ . To define preferred repairs, we first need to define when a repair satisfies a ground preference rule.

Intuitively, a repair R satisfies a ground preference rule  $\rho$  iff whenever R (together with the ontology) entails the body of  $\rho$ , the head of  $\rho$  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  $(R, \Sigma)$  is consistent.

**Definition 5.** Let  $\mathcal{K} = (D, \Sigma, \Pi)$  be a prioritized knowledge base, R a consistent subset of D, and  $\rho$  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  $(R, \Sigma) \models q(\mathbf{b})$  implies  $(R, \Sigma) \models p(\mathbf{a})$ . We say that Rsatisfies  $\rho$  (w.r.t.  $\mathcal{K}$ ) iff  $(R, \Sigma) \models \exists \mathbf{Z} \varphi(\mathbf{Z}, \mathbf{c})$  implies R fulfills  $p(\mathbf{a}) \succ q(\mathbf{b})$  w.r.t.  $\mathcal{K}$ .  $\Box$ 

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  $\Pi$ .

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 6.** A repair R of a prioritized knowledge base  $\mathcal{K}$  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  $PRep(\mathcal{K})$ .  $\Box$ 

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. 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 7.** 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  $\circ$  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  $(R_i, \Sigma)$ , for each  $i \in \{1, 2, 3, 4\}$ . Thus, such preference rules are satisfied by all repairs. Hence, we can focus on these three ground preference rules:

- $\rho'_1$ : hasWine(o)  $\succ$  hasBeer(o)  $\leftarrow$  hasMeat(o),
- $\rho_2' \colon \mathsf{dessert}(\mathsf{o},\mathsf{cake}) \succ \mathsf{dessert}(\mathsf{o},\mathsf{cherries}) \gets \mathsf{main}(\mathsf{o},\mathsf{beef}),$
- $\rho'_3 \colon \mathsf{dessert}(\mathsf{o},\mathsf{pie}) \succ \mathsf{dessert}(\mathsf{o},\mathsf{cake}) \leftarrow \exists Y \mathsf{hasDrink}(\mathsf{o}), \mathsf{side}(\mathsf{o},Y).$

Notice that  $\rho'_1$  is applicable in all repairs. Repairs  $R_1$  and  $R_2$  satisfy  $\rho'_1$  because they both contain red wine;  $R_4$  satisfies  $\rho'_1$  because it does not include any wine or beer;  $R_3$  does not satisfy  $\rho'_1$  because it includes beer and no wine.

The preference rule  $\rho'_2$  is applicable in all repairs as well, and it is satisfied by all of them, essentially because none of them contain dessert(0, cherries).

Finally,  $R_1$ ,  $R_3$ , and  $R_4$  satisfy  $\rho'_3$ , 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.

We now provide our inconsistency-tolerant semantics over preferred repairs.

**Definition 8.** A prioritized knowledge base  $\mathcal{K} = (D, \Sigma, \Pi)$  entails a BCQ Q under the

- PAR semantics if  $(R, \Sigma) \models Q$ , for every  $R \in PRep(\mathcal{K})$ .
- *IPAR* semantics if  $(D_I, \Sigma) \models Q$ , where  $D_I = \bigcap \{R \mid R \in PRep(\mathcal{K})\}$ .
- ICPR semantics if  $(D_C, \Sigma) \models Q$ , where  $D_C = \bigcap \{Cn((R, \Sigma)) \mid R \in PRep(\mathcal{K})\}.$

**Example 9.** 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 7). 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.

#### 4. Complexity Results

We performed a thorough study of the data and combined complexity of the "preferred" variant of classical problems in the context of querying inconsistent knowledge bases. The first problem, which we dub PREFERRED REPAIR CHECKING (PRC), given a prioritized knowledge base  $\mathcal{K}$  and a database D', asks whether  $D' \in PRep(\mathcal{K})$ . The remaining problems, called S-ENTAIL, with  $S \in \{\text{PAR, IPAR, ICPR}\}$ , given a prioritized knowledge base  $\mathcal{K}$  and a BCQ Q, ask whether  $\mathcal{K}$  entails Q under the S semantics.

Language	PRC		PAR		IPAR		ICPR	
	Data	Comb.	Data	Comb.	Data	Comb.	Data	Comb.
L, LF, AF		PSpace		PSpace		PSpace		PSpace
SF		Exp		Exp		Ехр		Exp
F, GF		Exp		Exp		Exp		Exp
WA	coNP	2Exp	$\Pi_2^{P}$	2Exp	$\Pi_2^{P}$	2Exp	$\Pi_2^{P}$	2Exp
S		Exp		Exp		Ехр		Exp- 2Exp
Α		P <sup>NExp</sup>		P <sup>NExp</sup>		P <sup>NExp</sup>		P <sup>NExp</sup> - Exp <sup>NExp</sup>
G, WS		2Exp		2Exp		2Exp		2Exp- 3Exp

#### Table 1

Data and combined complexity of preferred repair checking and preferred entailment.

Table 1 reports the data and combined complexity of the above problems, when considering different TGD languages. 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: the data complexity goes from membership in P to coNP-completeness. For the AR and ICR semantics, the data complexity goes from coNP-completeness of the standard setting to  $\Pi_2^p$ -completeness of our preferred framework. For the IAR semantics, the data complexity increases from membership in AC<sup>0</sup> or coNP-completeness (depending on the language) to  $\Pi_2^p$ -completeness. In the combined complexity, moving from the classical semantics to the preferred ones does not yield an increase of complexity, except for the languages S, A, G, and WS under the ICPR semantics. Indeed, despite our efforts, in such cases, we did not manage to obtain completeness results. We point out that some of our results immediately extend to other TGD languages. For example, the  $\Pi_2^p$  memberships in data complexity also hold for TGDs classes for which the so-called skolem chase always terminates, e.g. see [12, 13] – for such classes, classical BCQ entailment is always tractable in data complexity.

#### 5. Conclusion

Expressing preferences is a common need in various domains, such as querying inconsistent knowledge bases [14, 15], game theory [16], and answer set programming [17]. 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.

**Related work.** In the context of logic programming, logic programs have been combined with preferences so as to determine a set of *preferred answer sets*—e.g., see [17, 18]— while we use preferences to determine a set of preferred repairs.

Another related framework are *(existential) active integrity constraints* (EAICs) [19], i.e. integrity constraints that also specify the updates that are allowed to restore consistency when

the constraint is violated. *Founded* repairs are those repairs obtained by applying only allowed updates. This expresses a sort of hard preference by partitioning conflicting facts into allowed and non-allowed ones, while our framework allows us to express preferences among arbitrary facts, and thus do not forbid any fact a priori.

**Future work.** 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. Also, it would be interesting to extend the notion of an *explanation*, already investigated under existential rules [20, 21], to our framework with preferences. Finally, considering the intractability of our reasoning tasks, another interesting direction is to consider approximations, i.e., compute a subset of query answers over our semantics in polynomial time, as done in the context of querying incomplete databases [22, 23]. Moreover, one could also consider the counting variants of the PAR semantics, i.e., to count the number of preferred repairs that entail the query (together with the TGDs). This is a more refined measure providing useful insights when the query is not entailed by *every* preferred repair and has been thoroughly studied in the database setting for different integrity constraints, e.g., see [24, 25, 26, 27].

#### References

- M. Arenas, L. E. Bertossi, J. Chomicki, Consistent query answers in inconsistent databases, in: Proc. PODS, 1999, pp. 68–79.
- [2] D. Lembo, M. Lenzerini, R. Rosati, M. Ruzzi, D. F. Savo, Inconsistency-tolerant semantics for description logics, in: RR, 2010, pp. 103–117.
- [3] M. Bienvenu, On the complexity of consistent query answering in the presence of simple ontologies, in: Proc. AAAI, 2012, pp. 705–711.
- [4] M. Calautti, S. Greco, C. Molinaro, I. Trubitsyna, Preference-based inconsistency-tolerant query answering under existential rules, in: KR, 2020, pp. 203–212.
- [5] A. Calì, G. Gottlob, T. Lukasiewicz, A general Datalog-based framework for tractable query answering over ontologies, J. Web Sem. 14 (2012) 57–83.
- [6] G. Gottlob, E. Malizia, Achieving new upper bounds for the hypergraph duality problem through logic, in: Proc. LICS, 2014, pp. 43:1–43:10.
- [7] A. Calì, G. Gottlob, M. Kifer, Taming the infinite chase: Query answering under expressive relational constraints, J. Artif. Intell. Res. 48 (2013) 115–174.
- [8] A. Calì, G. Gottlob, A. Pieris, Towards more expressive ontology languages: The query answering problem, Artif. Intell. 193 (2012) 87–128.
- [9] R. Fagin, P. G. Kolaitis, R. J. Miller, L. Popa, Data exchange: semantics and query answering, Theor. Comput. Sci. 336 (2005) 89–124.
- [10] T. Lukasiewicz, M. V. Martinez, A. Pieris, G. I. Simari, From classical to consistent query answering under existential rules, in: Proc. AAAI, 2015, pp. 1546–1552.

- [11] T. Lukasiewicz, E. Malizia, C. Molinaro, Complexity of approximate query answering under inconsistency in Datalog+/-, in: Proc. IJCAI, 2018, pp. 1921–1927.
- [12] M. Calautti, S. Greco, C. Molinaro, I. Trubitsyna, Logic program termination analysis using atom sizes, in: Proc. IJCAI, volume 2015-January, 2015, pp. 2833–2839.
- [13] M. Calautti, S. Greco, I. Trubitsyna, Detecting decidable classes of finitely ground logic programs with function symbols, ACM Trans. Comput. Log. 18 (2017) 28:1–28:42.
- [14] M. Bienvenu, C. Bourgaux, F. Goasdoué, Querying inconsistent description logic knowledge bases under preferred repair semantics, in: Proc. AAAI, 2014, pp. 996–1002.
- [15] S. Staworko, J. Chomicki, J. Marcinkowski, Prioritized repairing and consistent query answering in relational databases, Ann. Math. Artif. Intell. 64 (2012) 209–246.
- [16] G. Greco, E. Malizia, L. Palopoli, F. Scarcello, Non-transferable utility coalitional games via mixed-integer linear constraints, J. Artif. Intell. Res. 38 (2010) 633–685.
- [17] G. Brewka, I. Niemelä, M. Truszczynski, Answer set optimization, in: Proc. IJCAI, 2003, pp. 867–872.
- [18] S. Greco, I. Trubitsyna, E. Zumpano, On the semantics of logic programs with preferences, J. Artif. Intell. Res. 30 (2007) 501–523.
- [19] M. Calautti, L. Caroprese, S. Greco, C. Molinaro, I. Trubitsyna, E. Zumpano, Existential active integrity constraints, Expert Syst. Appl. 168 (2021) 114297.
- [20] İ. İ. Ceylan, T. Lukasiewicz, E. Malizia, A. Vaicenavicius, Explanations for query answers under existential rules, in: Proc. IJCAI, 2019, pp. 1639–1646.
- [21] T. Lukasiewicz, E. Malizia, C. Molinaro, Explanations for inconsistency-tolerant query answering under existential rules, in: Proc. AAAI, 2020, pp. 2909–2916.
- [22] N. Fiorentino, S. Greco, C. Molinaro, I. Trubitsyna, ACID: A system for computing approximate certain query answers over incomplete databases, in: Proc. SIGMOD, 2018, pp. 1685–1688.
- [23] S. Greco, C. Molinaro, I. Trubitsyna, Approximation algorithms for querying incomplete databases, Inf. Syst. 86 (2019) 28–45.
- [24] S. Greco, C. Molinaro, Probabilistic query answering over inconsistent databases, Ann. Math. Artif. Intell. 64 (2012) 185–207.
- [25] S. Greco, C. Molinaro, I. Trubitsyna, Computing approximate query answers over inconsistent knowledge bases, in: IJCAI, 2018, pp. 1838–1846.
- [26] M. Calautti, M. Console, A. Pieris, Counting database repairs under primary keys revisited, in: D. Suciu, S. Skritek, C. Koch (Eds.), Proc. PODS, ACM, 2019, pp. 104–118.
- [27] E. Livshits, B. Kimelfeld, J. Wijsen, Counting subset repairs with functional dependencies, J. Comput. Syst. Sci. 117 (2021) 154–164.