arXiv:cs.DB/0506063 v1 14 Jun 2005

Priority-Based Conflict Resolution in Inconsistent Relational Databases*

Slawomir Staworko Jan Chomicki
University at Buffalo
{staworko,chomicki}@cse.buffalo.edu

Abstract

We study here the impact of priorities on conflict resolution in inconsistent relational databases. We extend the framework of [1], which is based on the notions of repair and consistent query answer. We propose a set of postulates that an extended framework should satisfy and consider two instantiations of the framework: (locally preferred) ℓ-repairs and (globally preferred) g-repairs. We study the relationships between them and the impact each notion of repair has on the computational complexity of repair checking and consistent query answers.

1 Introduction

The main purpose of integrity constraints is to express semantic properties of the data stored in the database. Usually, it is the database management system that is responsible for maintaining the integrity of the database. However, in many recent applications the integrity enforcement becomes a problematic issue. For example in the data integration setting, even when the data contained by a data source satisfies the integrity constrains, a different data source may contribute conflicting information. At the same time data sources may be autonomous and it may be impossible to modify their contents in order to remove the conflicts. Integrity constraints may also fail to be enforced because of efficiency considerations. Finally, in the case of long running operations, integrity violations may be only temporary and will be eliminated by further operations.

Typically, the user formulates a query with the assumption that the database is consistent (i.e. satisfies the integrity constraints). A simple evaluation of the query over an inconsistent database may return incorrect answers. To address this problem Arenas, Bertossi, and Chomicki [1] proposed the framework of

[†]UB CSE Technical Report 2005-11

*Research supported by NSF Grants IIS-0119186 and IIS-0307434.

consistent query answers. They introduced the notion of a *repair*: a consistent database that is minimally different from the original one. A *consistent answer* to a query is an answer *true* in *every* repair. The framework of [1] is used as a foundation for most of the work in the area of querying inconsistent databases [2, 3, 7, 5, 11, 15, 14, 4].

Example 1.1. Consider a database consisting of two tables Emp and Mgr whose instance I_0 can be found in Table 1.

Emp

Name	Dept
Alice	A
Alice	B

Mgr

Dept	Name	T
A	Mary	2
B	Bob	1
B	Mary	3

Table 1: Instance I_0

Assume that we have two functional dependencies $Emp : Name \rightarrow Dept$ and $Mgr : Dept \rightarrow Name$. This database contains two conflicts: 1) in relation Emp between the tuples $(Alice, A)$ and $(Alice, B)$; 2) in relation Mgr between the tuples $(B, Mary, 3)$ and $(B, Bob, 1)$ (Note that one person can be the manager of more than one department). Each of those conflicts can be resolved in two different ways by assuming that one tuple is correct and removing the other. This leads to four different repairs:

$$I_1 = \{Emp(Alice, A), Mgr(A, Mary, 2), Mgr(B, Bob, 1)\},$$
$$I_2 = \{Emp(Alice, B), Mgr(A, Mary, 2), Mgr(B, Bob, 1)\},$$
$$I_3 = \{Emp(Alice, A), Mgr(A, Mary, 2), Mgr(B, Mary, 3)\},$$
$$I_4 = \{Emp(Alice, B), Mgr(A, Mary, 2), Mgr(B, Mary, 3)\}.$$

For example, the repair I_1 is obtained by assuming that *Alice* works in department A and the manager of department B is *Bob*. Since in every repair $Mary$ is the manager of the department A, we can infer that *true* is the consistent answer to the query

$$\phi_1 = Mgr(A, Mary).$$

However it is not certain that *Alice* works in a department managed by $Mary$, i.e. *true* is not the consistent answer to the following query

$$\phi_2 = \exists x.Emp(Alice, x) \wedge Mgr(x, Mary).$$

This is because of the repair I_2, where ϕ_2 is false.

As it is shown in the previous example, each conflict can be resolved in two different ways. The framework of [1] does not provide any means to favor one way over another. However, in many cases some additional information which can be used to provide a resolution of some conflicts is available. For example:

- In e-commerce applications, data are accompanied with the timestamp of creation/last modification — the conflicts can be resolved by removing from consideration old, outdated tuples.
- In data integration scenarios, it is often possible to provide a (partial) order on the sources, capturing the reliability of contributed information — the most reliable data can be used to resolve conflicts.
- Statistics can be used to resolve conflicts created by misspellings.

Example 1.2 (cont. Example 1.1)**.** Suppose that the column T of the table Mgr contains for each tuple its creation timestamp (lower values correspond to older tuples). We can use this information to express the preference that if some tuples of Mgr are conflicting, the older should be removed from consideration (but not removed from the database). Since the tuple $(B, Bob, 1)$ is older than $(B, Mary, 3)$, we consider only the repairs containing the latter one: I_3 and I_4. In such a case we can also infer that it is certain that *Alice* works in the department managed by $Mary$, i.e. *true* is the *preferred* consistent answer to the query ϕ_2.

In this paper we extend the framework of consistent query answers with an additional input consisting of preference information Φ. We use Φ to define the set of *preferred* repairs Rep^{Φ}. When we compute consistent answers, instead of considering the set of all repairs Rep, we use the set of preferred repairs. We assume that there exists a (possibly partial) operation of extending Φ with some additional preference information and we write $\Phi \subseteq \Psi$ when Ψ is an *extension* of Φ. We consider Φ to be *maximal* when it cannot be extended further. The main objective of our research is to develop a framework of preferred repairs that fulfills the following postulates:

1. **Non-emptiness**

   $(\mathcal{P}1)$ $$\mathrm{Rep}^{\Phi} \neq \varnothing.$$

2. **Non-discrimination**: if no preference information is given, then no repair is removed from consideration

   $(\mathcal{P}2)$ $$\mathrm{Rep}^{\varnothing} = \mathrm{Rep}\,.$$

3. **Monotonicity**: extending preferences can only narrow the set of preferred repairs

   $(\mathcal{P}3)$ $$\Phi \subseteq \Psi \Rightarrow \mathrm{Rep}^{\Psi} \subseteq \mathrm{Rep}^{\Phi}\,.$$

4. **Categoricity**: given maximal preference information we obtain exactly one repair

   $(\mathcal{P}4)$ $$\Phi \text{ is maximal} \Rightarrow |\,\mathrm{Rep}^{\Phi}\,| = 1.$$

We note here that the postulates $\mathcal{P}1$ and $\mathcal{P}2$ together imply an important property of **conservativeness**: preferred repairs are a subset of the standard repairs.

Another important goal of our research is to determine the computational implications of introducing preferences. For this purpose we study here two fundamental decision problems in inconsistent databases [9]: (i) *repair checking* — finding if a given database is a preferred repair; (ii) *computing consistent answers* — finding if an answer to a query is present in every preferred repair.

The main contributions of this paper are:

- A general and intuitive framework for incorporating preferences into inconsistency handling based on the notion of priority.
- A study of the semantic and computational properties of two instantiations of the framework: (locally preferred) ℓ-repairs and (globally preferred) g-repairs.

2 Basic notions and definitions

In this paper, we work with databases over a schema consisting of only one relation R with attributes from U. We use $A, B, \ldots$ to denote elements of U and $X, Y, \ldots$ to denote subsets of U. We consider two disjoint domains: uninterpreted names D and natural numbers N. Every attribute in U is typed. We assume that constants with different names are different and that symbols $=, \neq, <, >$ have the natural interpretation over N.

The instances of R, denoted by $r, r', \ldots$, can be seen as finite, first-order structures, that share the domains D and N. For any tuple t from r by $t.A$ we denote the value associated with the attribute A. In this paper we consider first-order queries over the alphabet consisting of R and binary relation symbols $=, \neq, <$, and $>$.

The limitation to only one relation is made only for the sake of clarity and along the lines of [10] the framework can be easily extended to handle databases with multiple relations.

2.1 Inconsistency and repairs

The class of integrity constraints we study consists of functional dependencies. We use $X \rightarrow Y$ to denote the following constraint:

$$\forall t_1, t_2 \in R. \bigwedge_{A \in X} t_1.A = t_2.A \Rightarrow \bigwedge_{B \in Y} t_1.B = t_2.B;$$

We use this formula to identify tuples creating conflicts.

Definition 2.1 (Conflicting tuples)**.** Given a set of functional dependencies F, two tuples t_1, t_2 are *conflicting* w.r.t F, denoted $t_1 \nleftrightarrow_F t_2$, if and only if there exists a functional dependency $X \rightarrow Y \in F$ such that $t_1.A = t_2.A$ for all $A \in X$ and $t_1.B \neq t_2.B$ for some $B \in Y$.

Definition 2.2 (Inconsistent database)**.** A database r is *inconsistent* with a set of constraints F if and only if r contains some conflicting tuples. Otherwise, the database is *consistent*.

In the general framework when repairing a database we consider two operations: adding or removing a tuple. Because in the presence of functional dependencies adding new tuples cannot remove conflicts, we only consider repairs obtained by deleting tuples from the original instance.

Definition 2.3 (Repair)**.** Given a database r and a set of integrity constraints F, a database r' is a *repair* of r w.r.t. F if r' is a maximal subset of r consistent with F.
We denote by $\mathrm{Rep}_F(r)$ the set of all repairs of r w.r.t F.

A repair can be viewed as the result of a process of cleaning the input relation. Note that since every conflict can be resolved in two different ways and conflict are often independent, there may be an exponential number of repairs. Also, the set of repairs of a consistent relation r contains only r.

2.1.1 Conflict graphs

Definition 2.4 (Conflict graph)**.** [3] A *conflict graph* $G_{r,F}$ is a graph whose set of vertices is equal to r and two tuples t_1, t_2 are adjacent only if they are conflicting (i.e. $t_1 \leftrightsquigarrow_F t_2$).

Recall that a maximal independent set of a graph G is a maximal set of vertices that contains no edge from G. By $\mathrm{MIS}(G)$ we denote the set of all maximal independent sets of G. The following observation explains why the conflict graph is considered a *compact representation of all repairs*.

Fact 2.5. *For any database r and any set of functional dependencies F we have that*

$$\mathrm{Rep}_F(r) = \mathrm{MIS}(G_{r,F}).$$

2.2 Priorities and preferred repairs

For the clarity of presentation we assume that from now on we work with a fixed database instance r and a fixed set of functional dependencies F.

To represent the preference information, we use (possibly partial) orientations of the conflict graph. It allows us to express preferences at the level of single conflicts.

Definition 2.6 (Priority)**.** A binary relation $\prec \subseteq r \times r$ is a *priority* if:

1. $\prec$ is asymmetric, i.e.

$$\forall x, y \in r. \neg[x \prec y \wedge y \prec x],$$

2. $\prec$ is defined only on conflicting tuples, i.e.

$$\forall x, y \in r. x \prec y \Rightarrow x \nleftrightsquigarrow_F y.$$

If $x \prec y$ we say that the pair $\{x, y\}$ is *prioritized* and that y *dominates* over x. A priority $\prec$ is *total* if every pair of conflicting tuples is prioritized by $\prec$. A priority $\prec$ is *acyclic* if there does not exist $x \in r$ such that $x \prec^* x$, where $\prec^*$ is the transitive closure of $\prec$.

The first condition of priority demands the preference information to be unambiguous for a single conflict. The second condition ensures that we are given only the relevant preference information. If the second condition is not fulfilled, then it can be easily enforced by intersecting $\prec$ with $\nleftrightsquigarrow_F$.

This form of preference information allows us to easily define the the preference extension: we orient some conflicting edges that were not oriented before.

Definition 2.7 (Priority extension)**.** A priority $\prec'$ is an *extension* of a priority $\prec$ if $\prec'$ agrees with $\prec$ where $\prec$ is defined (i.e. $\prec' \supseteq \prec$).

Note that $\prec$ cannot be extended further only if $\prec$ is total. Also an extension $\prec'$ of a priority $\prec$ is also a priority and therefore $\prec'$ is antisymmetric and defined only on pairs of conflicting tuples.

Now we present two methods of using a priority to restrict the set of all repairs of a given relation. The first one, ℓ-repairs, uses the priority to restrict the ways of constructing a repair (cleaning the database). The process consists of multiple iterative steps and in each of them only a limited number of conflicts is considered. The use of the priority has a local character because the subset of priority used in one step is not used in any further steps. The second method, g-repairs, uses the priority in a global fashion by selecting most preferred repairs according to an order induced by the priority.

2.2.1 Locally preferred repairs

Recall a general nondeterministic procedure for constructing a maximal independent set of a graph: as long as the graph is not empty, we *choose* a vertex, add it to the constructed set, and remove the vertex and all its neighbors from the graph. Depending on the choices of vertices we make, we can construct any maximal independent set of the input graph. Now, let's look at this procedure from the point of constructing a repair. Each choice of a vertex corresponds to taking a single repair action: keeping the corresponding tuple in the relation and removing all tuples conflicting with it.

Since the choice of the tuple to keep is unconstrained, every conflict can be resolved in several different ways. We use the priority to restrict the possible ways of choosing the tuple that will be kept and whose conflicts will be resolved. The chosen tuple is among those that are not dominated at the given step of the repairing process. We use the *winnow operator* [8] to formally describe the set of tuples that we choose from:

$$\omega_{\prec}(s) = \{t \in s | \neg \exists t' \in s. t \prec t'\}.$$

Algorithm 1 implements the construction of preferred repairs. An *ℓ-repair* (or a *locally preferred* repair) is any instance r' we can obtain with this Algorithm. We denote the set of all ℓ-repairs of r w.r.t. F and $\prec$ by $\mathrm{LRep}_F^{\prec}(r)$. Note that

Algorithm 1 Nondeterministic construction of an ℓ-repair

1: $r' \leftarrow \varnothing$
2: $s \leftarrow r$
3: **while** $\omega_{\prec}(s) \neq \varnothing$ **do**
4: choose any $x \in \omega_{\prec}(s)$
5: $r' \leftarrow r' \cup \{x\}$
6: $s \leftarrow s \setminus v(x)$ ▷ where $v(x) = \{x\} \cup \{y | x \leftrightsquigarrow_F y\}$
7: **return** r'

an ℓ-repair can be characterized by the sequence of choices made in the step 4 in Algorithm 1 (however there can be more than one such sequence). This observation allows us to state an alternative definition of an ℓ-repair.

Proposition 2.8. *Given a priority $\prec$, a set of tuples X is an ℓ-repair, if and only if there exists an ordering $x_1, \ldots, x_n$ of X such that for every $i \in \{0, \ldots, n-1\}$ the following set is non-empty*

$$(X \setminus \{x_1, \ldots, x_i\}) \cap \omega_{\prec}\left(r \setminus \left(v(x_1) \cup \ldots \cup v(x_i)\right)\right)$$

and $\omega_{\prec}(r \setminus (v(x_1) \cup \ldots \cup v(x_n)) = \varnothing$.

2.2.2 Globally preferred repairs

The next construction uses the priority directly to compare two repairs. Intuitively, one repair is better than another if all the differences between them are justified by the priority. Formally, we define g-repairs in the following way.

Definition 2.9 (Globally preferred repair)**.** Given a priority $\prec$ and two repairs $r_1, r_2 \in \mathrm{Rep}_F(r)$, we say that r_2 is *preferred* over r_1, and write $r_1 \ll r_2$, if

$$\forall x \in r_1 \setminus r_2.\ \exists y \in r_2 \setminus r_1.\ x \prec y.$$

A repair is a g-repair (or a *globally* preferred repair) if it is a $\ll$-maximal repair. By $\mathrm{GRep}_F^{\prec}(r)$ we denote the set of all g-repairs.

This particular "lifting" of a preference on objects to a preference on sets of objects can be found in other contexts. For example, a similar definition is used for a preference among different models of a logic program [23], or for a preference among different worlds [19].

2.3 Consistent query answers

In this paper, we use a generalized notion of consistent query answers. Instead of taking the set of all repairs, as in [1], we consider families of preferred repairs.

We only study closed first-order logic queries. We can easily generalize our approach to open queries along the lines of [1, 10]. For a given query φ we say that *true* is an answer to φ in r, if $r \models \varphi$ in the standard model-theoretic sense.

Definition 2.10 ($\mathcal{H}$-Consistent query answer)**.** Given a closed query φ and a family of repairs $\mathcal{H} \subseteq \mathrm{Rep}_F(r)$, *true* is the *$\mathcal{H}$-consistent query answer* to a query φ if for every repair $r' \in \mathcal{H}$ we have $r' \models \varphi$.

Note that we obtain the original notion of consistent query answer [1] if we take for $\mathcal{H}$ the whole set of repairs $\mathrm{Rep}_F(r)$.

In this paper, we study the cases when we take for $\mathcal{H}$ either the set of ℓ-repairs or the set of g-repairs. This gives us two notions:

1. *ℓ-preferred consistent query answer* if $\mathcal{H} = \mathrm{LRep}_F^{\prec}(r)$,
2. *g-preferred consistent query answer* if $\mathcal{H} = \mathrm{GRep}_F^{\prec}(r)$.

We write $r \models^{\ell}_{F,\prec} \phi$ ($r \models^{g}_{F,\prec} \phi$) to denote that *true* is the ℓ-preferred (resp. g-preferred) consistent answer to φ (in r w.r.t. F and $\prec$).

3 Basic properties

3.1 Cyclic priorities

Before discussing specific properties of preferred repairs, we present reasons for removing cyclic priorities from consideration.

Example 3.1. Assume a database schema $R(A, B)$ and a set of functional dependencies $F = \{A \to B, B \to A\}$. Consider the following database

$$r = \{t_a = (1, 1), t_b = (1, 2), t_c = (2, 2), t_d = (2, 1)\}$$

and a total cyclic priority $\prec = \{(t_a, t_b), (t_b, t_c), (t_c, t_d), (t_d, t_a)\}$. The set of all repairs is

$$\mathrm{Rep}_F(r) = \{r_1 = \{t_a, t_c\}, r_2 = \{t_b, t_d\}\}.$$

As we can easily find $\mathrm{LRep}_F^{\prec}(r)$ is empty. It is also easy to see that $r_1 \ll r_2$ and $r_2 \ll r_1$ and thus $\mathrm{GRep}_F^{\prec} = \varnothing$. This violates the postulates $\mathcal{P}1$ and $\mathcal{P}4$.

Intuitively, a cycle in the conflict graph represents a mutually dependent group of conflicts (a solution of one conflict may restrict the ways of solving other conflicts). Our intention is to break the cycle by choosing a $\prec$-maximal element. If $\prec$ is cyclic, then such element does not exist, which makes the construction of a preferred repair impossible. We find this kind of preference information (cyclic priority) to be incoherent and we exclude it form our considerations.

3.2 Order properties of $\ll$

When we restrict our considerations only to acyclic priorities, the relation $\ll$ has interesting order properties.

Proposition 3.2. *If $\prec$ is an acyclic priority and the binary relation $\ll$ on $Rep_F(r)$ is defined in terms of $\prec$ as in Definition 2.9, then*

1. *$\ll$ is reflexive,*
2. *$\ll$ is anti-symmetric,*
3. *$\ll$ is transitive, provided that $\prec$ is transitive.*

Proof. Before proving the main thesis we will introduce one definition and show its two properties

Definition 3.3 (Alternating chain)**.** Given two sets $A, B \subseteq r$ and a priority $\prec$, an (A, B)-*alternating* $\prec$-*chain* is a (possibly infinite) sequence $\alpha_1, \alpha_2, \ldots$ such that:

- every element with even index belongs to A

$$\alpha_{2*i} \in A$$

- every element with odd index belongs to B

$$\alpha_{2*i+1} \in B$$

- $\prec$ holds between every two consecutive elements, i.e.

$$\alpha_i \prec \alpha_{i+1}$$

We say that an (A, B)-alternating $\prec$-chain is *maximal* if it's not a proper prefix of some (A, B)-alternating $\prec$-chain[1].

When $\prec$ will be know from the context instead of saying that $\{\alpha_i\}$ is an (A, B)-alternating $\prec$-chain we will simply say that $\{\alpha_i\}$ is an (A, B)-chain.

Proposition 3.4. *For any acyclic priority $\prec$ and any two sets $A, B \subseteq r$ every (A, B)-chain is finite.*

Proof. Suppose there exists such an infinite (A, B)-chain $\{\alpha_i\}$. Because r is finite, $\{\alpha_i\}$ contains a recurrent element x. Thus

$$x \prec \ldots \prec x.$$

This gives us a contradiction with $\prec$ being acyclic. □

[1] A sequence $\{a_i\}_{i=1}^{n}$ is a proper prefix of a sequence $\{b_i\}_{i=1}^{m}$ if and only if $n < m$ and $a_i = b_i$ for every $i \in \{1, \ldots, n\}$. Note that $\{b_i\}_{i=1}^{m}$ can be infinite ($m = \infty$), but an infinite sequence cannot have a proper prefix.

Proposition 3.5. *For any acyclic priority $\prec$, and any two sets $X, Y \subseteq r$ such that $X \ll Y$ (where $\ll$ is defined in terms of $\prec$), any maximal $(X \setminus Y, Y \setminus X)$-chain is of even length (it ends with an element from $Y \setminus X$).*

Proof. By previous proposition we have that any $(X \setminus Y, Y \setminus X)$-chain is finite. Assume now that, there exists a maximal $(X \setminus Y, Y \setminus X)$-chain of odd length (i.e. ending with an element from $X \setminus Y$):

$$x_1 \prec y_1 \prec x_2 \prec y_2 \prec \ldots \prec x_k. \tag{1}$$

Since $X \ll Y$, there exists $y_k \in Y \setminus X$ such that $x_k \prec y_k$. Thus (1) is a prefix of the following $(X \setminus Y, Y \setminus X)$-chain:

$$x_1 \prec y_1 \prec x_2 \prec y_2 \prec \ldots \prec x_k \prec y_k.$$

This contradicts the maximality of (1). □

We also state a trivial fact

Fact 3.6. *For any acyclic priority $\prec$, any two sets $X, Y \subseteq r$ such that $X \ll Y$, and any $x \in X \setminus Y$ there exists an $(X \setminus Y, Y \setminus X)$-chain that starts with x.*

Now, we show the order properties of $\ll$:

1. $\ll$ is reflexive.

   Because universal quantification over empty set is true, then trivially $X \ll X$ for any set $X \subseteq r$.

2. $\ll$ is asymmetric.

   Take two different sets $X, Y \subseteq r$ such that $X \ll Y$ and $X \ll Y$, i.e.:

   $$\forall x \in X \setminus Y. \exists y \in Y \setminus X. x \prec y, \tag{2}$$
   $$\forall y \in Y \setminus X. \exists x \in X \setminus Y. y \prec x. \tag{3}$$

   W.l.o.g we can assume that $X \setminus Y \neq \varnothing$. Take any $x_1 \in X \setminus Y$. By (2) we are able to find $y_1 \in Y \setminus X$ such that $x_1 \prec y_1$. Now, by (3) we are able to find $x_2 \in X \setminus Y$ such that $y_1 \prec x_2$. This way we can construct an infinite $(X \setminus Y, Y \setminus X)$-chain. This contradicts Proposition 3.4.

3. If $\prec$ is transitive, then $\ll$ is transitive.

   Assume $\prec$ is transitive and take three different sets $X, Y, Z \subseteq r$ such that $X \ll Y$ and $Y \ll Z$ (the case when two sets are equal is trivial). Note that:

   $$\forall x \in X \setminus Y. \exists y \in Y \setminus X. x \prec y, \tag{4}$$
   $$\forall y \in Y \setminus Z. \exists z \in Z \setminus Y. y \prec z. \tag{5}$$

   Now we take any $x \in X \setminus Z$ and consider two cases depending if $x \in Y$ or not.

Suppose $x \in Y$. Let $x \prec \ldots \prec z$ be a maximal $(Y \setminus Z, Z \setminus Y)$-chain where $z \in Z \setminus Y$. (the existence of such a chain is by Proposition 3.5 and Fact 3.6). If there exists an element z' of this chain that belongs to $Z \setminus X$ then by transitivity of $\prec$ we have $x \prec z'$ (which end this path of the proof). Suppose that none of the elements of the $(Y \setminus Z, Z \setminus Y)$-chain belongs to $Z \setminus X$, then in particular z belongs to $X \setminus Y$. By (4) there exists $y \in Y \setminus X$ such that $z \prec y$. Moreover $y \in Z$ or otherwise we get a contradiction of the maximality of the $(Y \setminus Z, Z \setminus Y)$-chain. By transitivity of $\prec$ we get $x \prec y$ and obviously $y \in Z \setminus X$;

Similarly we deal with the case when $x \notin Y$. Take $x \prec \ldots \prec y$ to be a maximal $(X \setminus Y, Y \setminus X)$-chain, where $y \in Y \setminus X$. If there exists an element z' of this sequence that belongs to $Z \setminus X$, then by transitivity of $\prec$ we have $x \prec z'$ (which end this path of the proof). Suppose that none of the elements of the $(X \setminus Y, Y \setminus X)$-chain belongs to $Z \setminus X$, then in particular y belongs to $Y \setminus Z$. By (5) there exists $z \in Z \setminus Y$ such that $y \prec z$. Moreover $z \notin X$ or otherwise we get a contradiction of the maximality of the $(X \setminus Y, Y \setminus X)$-chain. Finally, by transitivity of $\prec$ we get $x \prec z$ and obviously $z \in Z \setminus X$. This ends the proof.

□

The following example shows that $\ll$ may not be transitive if the underlying priority is not transitive.

Example 3.7. Consider a database

$$r = \{t_a = (1,1), t_b = (1,2), t_c = (1,3)\}$$

over the schema $R(A, B)$ with one functional dependency $F = \{A \to B\}$ and with priority $\prec = \{(t_a, t_b), (t_b, t_c)\}$. There are three repairs of r:

$$\mathrm{Rep}_F(r) = \{A = \{t_a\}, B = \{t_b\}, C = \{t_c\}\}$$

The corresponding conflict graph is presented on Figure 1. We note that $A \ll B$

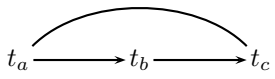

Figure 1: Conflict graph $G_{r,F}$ with orientation $\prec$

and $B \ll C$ but $A \not\ll C$.

3.3 Fulfillment of the postulates

Before we prove the fulfillment of the postulates $\mathcal{P}1$–$\mathcal{P}4$ we state an important property of the two instantiations of preferred repairs: constructing a repair

from the locally best tuples by the notion of ℓ-repairs conforms with the global notion of preference (g-repairs).

Theorem 3.8. *If $\prec$ is an acyclic priority, then*

$$\mathrm{LRep}_F^{\prec}(r) \subseteq \mathrm{GRep}_F^{\prec}(r).$$

Proof. Induction over the size of r. Trivial for $r = \varnothing$.

Assume the hypothesis holds for any proper subset of r and there exists $X \in \mathrm{LRep}_F^{\prec}(r)$ such that $X \ll Y$ for some $Y \in \mathrm{Rep}_F(r)$. By Proposition 2.8 $\omega_{\prec}(r) \cap X$ is non-empty. Take then any $x \in \omega_{\prec}(r) \cap X$. $x \in Y$ or otherwise we receive a contradiction $X \ll Y$. Note that $Y \setminus \{x\}$ is a repair of $r \setminus v(x)$ and $X \setminus \{x\}$ even a ℓ-repair of $r \setminus v(x)$. Moreover $X \setminus \{x\} \ll Y \setminus \{x\}$ in terms of the database $r \setminus v(x)$. Thus $X \setminus \{x\}$ is not g-repair of $r \setminus \{x\}$, which is a contradiction of the inductive hypothesis. □

In the following example we observe that the reverse containment does not hold for an arbitrary acyclic priority, i.e. the construction of ℓ-repairs by choosing only the best elements locally (as in ℓ-repairs) may miss a g-repair.

Example 3.9. Consider a database

$$r = \{t_a = (1,1,1), t_b = (2,1,2), t_c = (3,1,3), t_d = (4,1,3)\}$$

over the schema $R(A,B,C)$ with a set of functional dependencies $F = \{B \to C\}$ and a acyclic priority

$$\prec = \{(t_c, t_a), (t_d, t_b)\}$$

The set of repairs is $\mathrm{Rep}_F(r) = \{r_1 = \{t_a\}, r_2 = \{t_b\}, r_3 = \{t_c, t_d\}\}$. As we can easily find $\mathrm{GRep}_F^{\prec}(r) = \mathrm{Rep}_F(r)$. Because each of the t_c and t_d is dominated, the g-repair r_3 is not an ℓ-repair, and thus $\mathrm{LRep}_F^{\prec}(r) = \{r_1, r_2\}$.

Later on we present sufficient conditions under which both instantiations of preferred repairs are equivalent (Theorem 3.12).

We recall that extending priority consists of prioritizing conflicts not prioritized before and a priority that cannot be extended further (i.e. is maximal) is a total priority. Both classes of referred repairs that we consider satisfy the postulates $\mathcal{P}1$ – $\mathcal{P}4$:

Theorem 3.10 ($\mathcal{P}1$–$\mathcal{P}4$ for LRep)**.** *For every relation instance r, set of functional dependencies F, and acyclic priority $\prec$, $\mathrm{LRep}_F^{\prec}(r)$ satisfies $\mathcal{P}1$–$\mathcal{P}4$.*

Proof. We receive $\mathcal{P}1$ from the fact that if $\prec$ is acyclic then $\omega_{\prec}(X)$ is non-empty if and only if X is non-empty.

$\mathcal{P}2$ is implied by the fact that $\omega_{\varnothing}$ is an identity function what makes LRep a generic procedure for constructing all maximal independent sets of $G_{r,C}$.

To prove $\mathcal{P}3$ assume that $\prec', \prec$ are acyclic priorities such that $\prec' \subseteq \prec$. Take then any $X \in \mathrm{LRep}_F^{\prec}(r)$ and let σ be any ordering of X from Proposition 2.8. Note that since for any set A we have $\omega_{\prec}(A) \subseteq \omega_{\prec'}(A)$ then σ also fulfills conditions of Proposition 2.8 in terms of $\prec'$.

$\mathcal{P}4$ is a consequence of $\mathcal{P}1$ for LRep, Theorem 3.8, and $\mathcal{P}4$ for GRep. □

Theorem 3.11 ($\mathcal{P}1$–$\mathcal{P}4$ for GRep)**.** *For every relation instance r, set of functional dependencies F, and acyclic priority $\prec$,* $\mathrm{GRep}_F^{\prec}(r)$ *satisfies* $\mathcal{P}1$–$\mathcal{P}4$.

Proof. We get $\mathcal{P}1$ from the definition.

With an empty priority we cannot justify $X \ll Y$ for any two different repairs X and Y, what implies $\mathcal{P}2$.

To show $\mathcal{P}3$ assume that $\prec', \prec$ are acyclic priorities such that $\prec' \subseteq \prec$, $X \in \mathrm{GRep}_F^{\prec}(r)$, and suppose there exists $Y \in \mathrm{GRep}_F^{\prec'}(r)$ such that Y is preferred over X in terms of $\prec'$. But since $\prec' \subseteq \prec$ this implies that Y is also preferred over X in terms of $\prec$. This is a contradiction.

In order to prove $\mathcal{P}4$ assume there exist two different repairs X and Y in $\mathrm{GRep}_F^{\prec}(r)$. $X \not\ll Y$ implies that there exists an element $x \in X \setminus Y$ such that for any conflicting with x tuple y from $Y \setminus X$ we have $x \not\prec y$. Since $\prec$ is total for any such y we have $y \prec x$. Take all such tuples $y_1, \ldots, y_n$ and by Y' denote any repair that contains the following elements

$$Y \setminus \{y_1, \ldots, y_n\} \cup \{x\}$$

Such a repair exists because this set contains no conflicting tuples. Obviously $Y' \neq Y$ and at the same time $Y \ll Y'$. This contradicts that $Y \in \mathrm{GRep}_F^{\prec}(r)$. □

3.4 Equivalence of LRep and GRep

As we showed in Example 3.9 LRep doesn't have to be equal to GRep. It suffices, however, to remove from consideration priorities with cyclic extensions to obtain the equivalence of the two notions of preferred repair:

Theorem 3.12. *If $\prec$ is a priority having only acyclic extensions, then*

$$\mathrm{GRep}_F^{\prec}(r) = \mathrm{LRep}_F^{\prec}(r).$$

Proof. We need to show $\mathrm{GRep}_F^{\prec}(r) \subseteq \mathrm{LRep}_F^{\prec}(r)$. Take any $X \in \mathrm{GRep}_F^{\prec}(r)$ and construct $\prec'$ a total extension of $\prec$ by prioritizing (un-prioritized by $\prec$) conflicts in favor for X, i.e. $\prec'$ is any total priority such that for any $x \in X$ and any y if $x \rightsquigarrow_F y$ and $x \not\prec y$ then $y \prec x$. Since $\prec$ has only acyclic extensions $\prec'$ is acyclic. It should be clear from the construction that $X \in \mathrm{GRep}_F^{\prec'}(r)$. By $\mathcal{P}1$, $\mathcal{P}2$, $\mathcal{P}4$ and Theorem 3.8 this implies that $X \in \mathrm{LRep}_F^{\prec'}(r)$. This by $\mathcal{P}3$ gives us that $X \in \mathrm{LRep}_F^{\prec}(r)$. □

The following example shows, however, that the requirement of no cyclic extensions is not necessary for the equality above to hold.

Example 3.13. Consider schema $R(A, B, C)$ together with a set of functional dependencies $F = \{B \to C\}$. Suppose we have a database:

$$r = \{t_a = (1, 1, 1), t_b = (2, 1, 1), t_c = (3, 1, 2), t_d = (4, 1, 2)\}$$

with a priority $\prec = \{(t_c, t_a), (t_d, t_b)\}$. The conflict graph is presented on Figure 2. $\prec$ has a cyclic extension $\prec' = \prec \cup \{(t_a, t_d), (t_b, t_c)\}$. At the same time $\mathrm{LRep}_F^{\prec}(r) = \mathrm{GRep}_F^{\prec}(r) = \{\{t_a, t_b\}\}$.

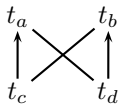

Figure 2: Conflict graph $G_{r,F}$ with orientation $\prec$

4 Computational properties

We study two fundamental problems of handling inconsistencies with priorities: (i) *repair checking* – determining if a database is a preferred repair of a given database; (ii) *consistent query answers* – checking if *true* is an answer to a given query in every preferred repair. We use the notion of *data complexity* [24] which captures the complexity of a problem as a function of the number of tuples in the database. The database schema, the integrity constraints, and the query are assumed to be fixed.

4.1 Locally preferred repairs

Recall Algorithm 1 and note that because the consecutive choices made in the step 4 consist of mutually non-conflicting tuples, the state of the computation is independent of the order of the choices[2]. Given a repair r', we can "simulate" its construction by restricting the choices in the step 4 to $r' \cap \omega_{\prec}(r)$. The simulation succeeds if and only if r' is an ℓ-repair.

Theorem 4.1. *Given a fixed set of functional dependencies F, the set*

$$B_F^{\ell} = \{(r, r', \prec) | r' \in \mathrm{LRep}_F^{\prec}(r)\}$$

is in PTIME.

It is shown in [9] that computing consistent answers to conjunctive queries is co-NP-complete, but if we consider only ground quantifier-free queries, the problem is in PTIME. On the other hand, computing ℓ-preferred consistent answers turns out to be an intractable problem even if we consider very simple, single-atom queries.

Theorem 4.2. *There exists a set of four functional dependencies F and a quantifier-free ground query φ (consisting of one atom only) such that the set*

$$D_{F,\varphi}^{\ell} = \{(r, \prec) | r \models_{F,\prec}^{\ell} \varphi\},$$

is co-NP-complete.

[2]The state of computation means the repair being constructed and the possible further choices.

Proof. It's easy to construct a nondeterministic Turing machine for $D^l_{F,\varphi}$ following informal description presented here: The machine uses nondeterministic transitions to compute all ℓ-preferred repairs of r and for each one checks the answer to φ. Note that

$$r \models^l_{F,\prec} \varphi \iff \forall r' \in \mathrm{LRep}^{\prec}_F(r).r' \models \varphi \iff \neg\exists r' \in \mathrm{LRep}^{\prec}_F(r).r' \models \neg\varphi.$$

This allows us to state that the constructed machine decides the complement of $D^l_{F,\varphi}$.
Now, consider the schema $R(A_1, B_1, \ldots, A_4, B_4)$ with the set of functional dependencies $F = \{A_1 \to B_1, \ldots, A_4 \to B_4\}$ and a ground query $\neg R(b)$, where the value of b can be found in Table 2.
We show here a polynomial reduction of the complement of $3SAT$ to $D^l_{\neg R(b),F}$, i.e. for any boolean formula φ in $3CNF$ we construct a pair $(r_\varphi, \prec_\varphi)$ of a polynomial size in the size of φ and such that

$$(r_\varphi, \prec_\varphi) \in D^l_{F,\neg R(b)} \iff \varphi \notin 3SAT.$$

Take then any formula φ in $3CNF$ and let n be the number of variables used in φ and k the number of conjuncts of φ. For simplicity we assume that:

- used variables have consecutive indexes $x_1, \ldots, x_n$,
- $\varphi = c_1 \wedge \ldots \wedge c_k$
- each conjunct consists of exactly three literals $c_j = l_{j,1} \vee l_{j,2} \vee l_{j,3}$ for $(j = 1, \ldots, k)$.

We define two auxiliary functions *var* and *sgn* on literals in the following fashion:

$$\begin{aligned} var(x_i) &= i, & sgn(x_i) &= 1, \\ var(\neg x_i) &= i, & sgn(\neg x_i) &= -1. \end{aligned}$$

The constructed database contains the following elements:

$$r_\varphi = \{v_1, \bar{v}_1, \ldots, \bar{v}_n, v_n, d_1 \ldots, d_k, b\}$$

whose exact values can be found in Table 2. The priority relation $\prec_\varphi$ is the

	A_1	B_1	A_2	B_2	A_3	B_3	A_4	B_4
v_i	i	1	i	-1	i	-1	i	-1
$\bar{v}_i$	i	2	i	1	i	1	i	1
d_j	0	1	$var(l_{j,1})$	$sgn(l_{j,1})$	$var(l_{j,2})$	$sgn(l_{j,2})$	$var(l_{j,3})$	$sgn(l_{j,3})$
b	0	0	0	0	0	0	0	0

Table 2: Values of tuples in r_φ

unique minimal binary relation on r_φ satisfying the following conditions:

$$\begin{aligned} d_j &\prec_\varphi v_{var(l_{j,i})}, && \text{for } j \in \{1,\dots,k\},\ i \in \{1,2,3\} \text{ such that } sgn(l_{j,i}) = 1,\\ d_j &\prec_\varphi \bar{v}_{var(l_{j,i})}, && \text{for } j \in \{1,\dots,k\},\ i \in \{1,2,3\} \text{ such that } sgn(l_{j,i}) = -1,\\ b &\prec_\varphi d_j, && \text{for } j \in \{1,\dots,k\}. \end{aligned}$$

Note that this priority relation is acyclic. Also note that construction of $(r_\varphi, \prec_\varphi)$ can be implemented in time polynomial in the size of the of the input formula φ. On Figure 3 we can find a conflict graph of an instance received from reduction of a formula $\varphi = (\neg x_1 \vee x_2 \vee x_3) \wedge (x_3 \vee \neg x_4 \vee x_5) \wedge (\neg x_5 \vee \neg x_6 \vee x_7)$.

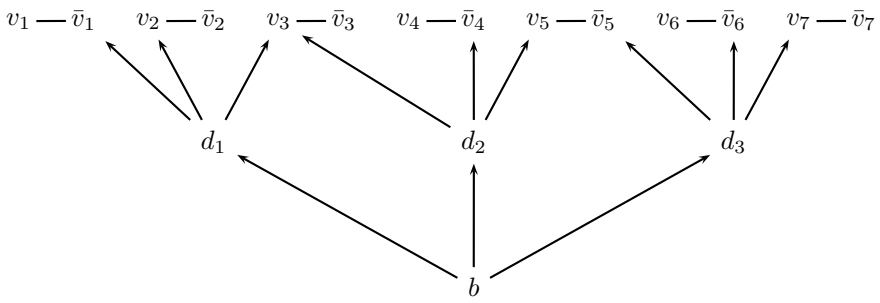

Figure 3: Conflict graph for $\varphi = (\neg x_1 \vee x_2 \vee x_3) \wedge (x_3 \vee \neg x_4 \vee x_5) \wedge (\neg x_5 \vee \neg x_6 \vee x_7)$ and orientation $\prec_\varphi$.

Now, we show that

$$\exists r' \in \mathrm{LRep}_F^{\prec_\varphi}(r_\varphi).b \in r' \iff \varphi \in 3SAT$$

$\boxed{\Rightarrow}$ Fist note that since $b \in r'$ then none of the tuples $d_1, \dots, d_k$ belongs to r'. Therefore for every $i \in \{1,\dots,n\}$ either v_i or $\bar{v}_i$ belongs to r'. Thus the following is a proper definition of a boolean valuation:

$$V(x_i) = \begin{cases} true & \text{if } v_i \in r' \\ false & \text{if } \bar{v}_i \in r' \end{cases}$$

Next, we show that φ is true for V. Suppose otherwise, i.e. there exists a conjunct c_m that is not true for V. W.l.o.g. we can assume that $c_m = x_1 \vee \neg x_2 \vee x_3$. This implies that $\{v_1, \bar{v}_2, v_3\} \cap r' = \varnothing$ and thus $\bar{v}_1, v_2, \bar{v}_3 \in r'$.

Take $t_1, \dots, t_n$ to be the ordering of r' from Proposition 2.8. Since no d_j tuples are present in r', and the tuple b is dominated by every d_j tuple (which in turn is dominated by some v_i and $\bar{v}_i$ tuples) then $t_n = b$. Let s be the last index of this sequence that t_s is equal to either $\bar{v}_1$, v_2, or $\bar{v}_3$. Since d_m is dominated only by v_1, $\bar{v}_2$, and v_3 we have for any $p \geq s$

$$d_m \in \omega_{\prec_\varphi}\big(r_\varphi \setminus (v(t_1) \cup \ldots \cup v(t_p))\big).$$

This implies that $\omega_{\prec_\varphi}(r_\varphi \setminus (v(t_1) \cup \ldots \cup v(t_n))) \neq \varnothing$ which gives a contradiction.

$\boxed{\Leftarrow}$ Take any valuation V for which φ is true and construct the following set

$$r' = \{b\} \cup \{v_i | V(x_i)\} \cup \{\bar{v}_i | \neg V(x_i)\}.$$

First, note that r' is a repair: it contains no conflicting tuples and for every tuple from $r_\varphi \setminus r'$ there exists a conflicting tuple in r'.

Next, we show that $r \in \mathrm{LRep}_F^{\prec_\varphi}(r_\varphi)$. In order to prove that we note that for any subset $X \subseteq r' \setminus \{b\}$ we have

$$d_j \notin \omega_{\prec_\varphi}\left(r_\varphi \setminus \bigcup_{x \in X} v(x)\right), \quad \text{for } j = 1, \ldots, k. \tag{6}$$

Suppose otherwise, i.e. there exists a set $X \subseteq r' \setminus \{b\}$ and m such that

$$d_m \in \omega_{\prec_\varphi}\left(r_\varphi \setminus \bigcup_{x \in X} v(x)\right).$$

W.l.o.g. we can assume that $c_m = x_1 \vee \neg x_2 \vee x_3$. From the construction of r_φ and $\prec_\varphi$ this implies that $\bar{v}_1, v_2, \bar{v}_3 \in X$ which is equivalent with $V(x_1) = false$, $V(x_2) = true$, and $V(x_3) = false$. This implies that c_m is not true for V which yields a contradiction with φ being satisfied by V.

The property (6) allows us to use Proposition 2.8 (take any ordering of r' with b on the last position) to state that r' is ℓ-preferred repair w.r.t F and $\prec_\varphi$.

It should be noted here that adding just one tuple $b' = (0, 0, 0, 1, 0, 1, 0, 1)$ and extending the priority with $b' \prec_\varphi b$ constructs a reduction of $3SAT$ to the complement of $D^l_{F,R(b')}$. And therefore computing ℓ-preferred consistent answers is intractable also for a query consisting only of one positive literal. □

4.2 Globally preferred repairs

Unlike ℓ-repairs, the notion of g-repairs, because of its global character, cannot be captured without an essential use of nondeterminism.

Theorem 4.3. *There exists a set of five functional dependencies F such that the set*

$$B^g_F = \{(r, r', \prec) | r' \in \mathrm{GRep}_F^{\prec}(r)\}$$

is co-NP-complete.

Proof. It's easy to construct a nondeterministic Turing machine B^g_F. The machine first checks if r' is a repair; if yes the machine nondeterministically computes every repair and checks if any of them (different than r') is preferred over r' w.r.t. $\prec$. This machine decides the complement of B^g_F.

Now, we show that the problem co-NP-hard by reducing the complement of $3SAT$ to B_F^g. Consider the database schema $R(A_1, B_1, \ldots, A_5, B_5)$ with the following set of integrity constraints $F = \{A_1 \to B_1, \ldots, A_5 \to B_5\}$. For any boolean formula φ in $3CNF$ we construct a triple $(r_\varphi, X_\varphi, \prec_\varphi)$ of size polynomial in the size of φ and such that

$$(r_\varphi, X_\varphi, \prec_\varphi) \in B_F^g \iff \varphi \notin 3SAT.$$

Moreover the reduction can be implemented in time polynomial in the size of φ.

Take then any formula φ in $3CNF$ and let n be the number of variables used in φ and k the number of conjuncts of φ. For simplicity we assume that:

- used variables have consecutive indexes $x_1, \ldots, x_n$,
- $\varphi = c_1 \wedge \ldots \wedge c_k$
- each conjunct consists of exactly three literals $c_j = l_{j,1} \vee l_{j,2} \vee l_{j,3}$ for $(j = 1, \ldots, k)$.

We define two auxiliary functions var and sgn on literals as follows:

$$\begin{aligned} var(x_i) &= i, & sgn(x_i) &= 1, \\ var(\neg x_i) &= i, & sgn(\neg x_i) &= -1. \end{aligned}$$

The constructed database contains the following elements

$$r_\varphi = \{v_1, \bar{v}_1, \ldots, v_n, \bar{v}_n, w_1, \ldots, w_n, d_1, \ldots, d_k, s, t\},$$

whose exact values can be found in Table 3.

	A_1	B_1	A_2	B_2	A_3	B_3	A_4	B_4	A_5	B_5
v_i	1	1	i	1	i	-1	i	-1	i	-1
$\bar{v}_i$	1	1	i	2	i	1	i	1	i	1
w_i	2	2	i	3	0	0	0	0	0	0
s	1	2	$n+1$	1	0	0	0	0	0	0
t	2	1	$n+1$	2	0	0	0	0	0	0
d_j	2	2	0	0	$var(l_{j,1})$	$sgn(l_{j,1})$	$var(l_{j,2})$	$sgn(l_{j,2})$	$var(l_{j,3})$	$sgn(l_{j,3})$

Table 3: Values of tuples in r_φ

The set X_φ consists of the following elements

$$X_\varphi = \{w_1, \ldots, w_n, d_1, \ldots, d_n, s\}.$$

It's easy to note that X_φ is a repair of r_φ w.r.t. F. Clearly $X_\varphi \subseteq r_\varphi$, no two elements of X_φ are conflicting, and for every element from the set $r_\varphi \setminus X_\varphi$ there exists a conflicting element from X_φ (s for t and w_i for v_i or $\bar{v}_i$).

The priority relation $\prec_\varphi$ is the unique minimal binary relation on r_φ satisfying the following conditions:

$$\begin{aligned}
s &\prec_\varphi t, && \\
w_i &\prec_\varphi v_i, && \text{for } i \in \{1, \ldots, n\}, \\
w_i &\prec_\varphi \bar{v}_i, && \text{for } i \in \{1, \ldots, n\}, \\
d_j &\prec_\varphi v_i, && \text{if } c_j \text{ uses a positive literal } x_i, \\
d_j &\prec_\varphi \bar{v}_i, && \text{if } c_j \text{ uses a negative literal } \neg x_i.
\end{aligned}$$

Note that this priority relation is acyclic. Also note that the triple $(r_\varphi, X_\varphi, \prec_\varphi)$ can be constructed in the time polynomial in the size of the formula φ. On Figure 4 we can find a conflict graph of the instance received from reduction of the formula $\varphi = (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_2 \vee \neg x_3 \vee x_4)$.

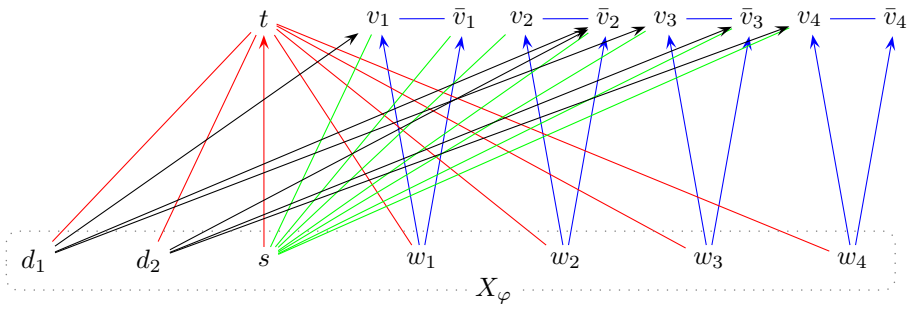

Figure 4: Conflict graph for $\varphi = (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_2 \vee \neg x_3 \vee x_4)$ and orientation $\prec_\varphi$.

Now, we show that for any φ using variables $x_1, \ldots, x_n$ the following holds

$$X_\varphi \notin \mathrm{GRep}_F^{\prec_\varphi}(r_\varphi) \iff \varphi \in 3SAT.$$

$\boxed{\Leftarrow}$ Suppose $\varphi \in 3SAT$ and take $V : \{x_1, \ldots, x_n\} \to \mathcal{B}$ to be the valuation for which φ is true. Consider the following set

$$Y_V = \{t\} \cup \{v_i | V(x_i)\} \cup \{\bar{v}_i | \neg V(x_i)\}$$

It's easy to find that Y_V is a repair and moreover $X_\varphi \ll Y_V$. Thus X_φ is not a maximally g-preferred repair.

$\boxed{\Rightarrow}$ Suppose $X_\varphi \notin \mathrm{GRep}_F^{\prec}(r)$, i.e. there exists $Y \in \mathrm{Rep}_F(r)$ such that $X \ll Y$ and $Y \neq X$.

First note that $t \in Y$. Otherwise for Y to be preferred over X the tuple s has to be contained in Y because there is no element dominating s except

for t. Since s is adjacent with every v_i and $\bar{v}_i$ then also none of v_i and $\bar{v}_i$ belongs to Y. This implies that $Y = X$ which is a contradiction.

Since t is adjacent to every element of X_φ and $t \in Y$ the sets Y and X_φ are disjoint. This implies that for every i the set Y contains either v_i or $\bar{v}_i$ (from maximality, independence, and the fact that $X \ll Y$).

Take now the following boolean valuation

$$V_Y(x_i) = \begin{cases} true & \text{if } v_i \in Y \\ false & \text{if } \bar{v}_i \in Y \end{cases}$$

We show that V_Y is a valuation for which φ is true. Suppose otherwise, that there exists a conjunct c_m that is not true under V_Y. W.l.o.g we can assume that $c_m = x_1 \vee \neg x_2 \vee x_3$. This implies that $\{v_1, \bar{v}_2, v_3\} \cap Y = \varnothing$. From the construction of $\prec_\varphi$ we know that there are no elements dominating over d_m except for $v_1, \bar{v}_2, v_3$. And since obviously $d_m \in X \setminus Y$, we receive $X \not\ll Y$ which is a contradiction.

□

Using the notion of g-repairs also leads to a significant increase of computational complexity when computing g-preferred consistent query answers.

Theorem 4.4. *There exists a set of four functional dependencies F and a quantifier-free ground query φ (consisting of one atom only) such that the set*

$$D^g_{\varphi,F} = \{(r, \prec) | r \models^g_{F,\prec} \varphi\}$$

is Π^p_2-complete.

Proof. The membership of $D^g_{F,\varphi}$ in Π^p_2 follows from the definition of g-preferred consistent query answer: query is not g-consistently true if it is false in some g-repair, and checking if a given set is a g-repair is in co-NP. We show Π^p_2-hardness below.

Consider a quantified boolean formula ψ of the form

$$\psi = \forall x_1, \ldots, x_n . \exists y_1, \ldots, y_m . \phi, \tag{7}$$

where ϕ is quantifier-free and is in 3CNF, i.e ϕ equals to $c_1 \wedge \ldots \wedge c_s$, and c_k are clauses of three literals $l_{k,1} \vee l_{k,2} \vee l_{k,3}$. We will construct a database instance r_ψ (over the schema $R(A_1, B_1, \ldots)$) and a priority relation $\prec_\psi$ such that true is a g-preferred consistent answer to a query $R(Y)$ if and only if ψ is true (the value of Y can be found in Table 4). The set of integrity constraints is $C = \{A_1 \to B_1, \ldots, A_4 \to B_4\}$.

We define two auxiliary functions *var* and *sgn* on literals in the following fashion:

$$var(x_i) = var(\neg x_i) = i, \qquad sgn(x_i) = sgn(y_j) = 1,$$
$$var(y_j) = var(\neg y_j) = n + j, \qquad sgn(\neg x_i) = sgn(\neg y_j) = -1.$$

Now, we describe the tuples contained in r_ψ.

$$r_\psi = \{p_1, \bar{p}_1, \ldots, p_n, \bar{p}_n, q_1, \bar{q}_1, \ldots, q_m, \bar{q}_m, d_1 \ldots, d_s\}.$$

The exact values of tuples can be found in Table 4. The priority relation $\prec_\psi$ is

	A_1	B_1	A_2	B_2	A_3	B_3	A_4	B_4
q_j	1	1	$n+j$	-1	$n+j$	-1	$n+j$	-1
$\bar{q}_j$	1	1	$n+j$	1	$n+j$	1	$n+j$	1
Y	1	1	0	0	0	0	0	0
X	1	2	0	0	0	0	0	0
p_i	1	2	i	1	i	1	i	1
$\bar{p}_i$	1	2	i	-1	i	-1	i	-1
d_k	1	2	$var(l_{k,1})$	$sgn(l_{k,1})$	$var(l_{k,2})$	$sgn(l_{k,2})$	$var(l_{k,3})$	$sgn(l_{k,3})$

Table 4: Values of tuples in r_ψ

the unique minimal priority relation that satisfies the following conditions:

$$\begin{aligned}
&d_k \prec_\psi p_i, && \text{if } c_k \text{ uses a positive literal } x_i,\\
&d_k \prec_\psi \bar{p}_i, && \text{if } c_k \text{ uses a negative literal } \neg x_i,\\
&d_k \prec_\psi q_j, && \text{if } c_k \text{ uses a positive literal } y_j,\\
&d_k \prec_\psi \bar{q}_j, && \text{if } c_k \text{ uses a negative literal } \neg y_j,\\
&p_i \prec_\psi Y, && \text{for all } i \in \{1, \ldots, n\},\\
&\bar{p}_i \prec_\psi Y, && \text{for all } i \in \{1, \ldots, n\},\\
&X \prec_\psi Y. &&
\end{aligned}$$

In Figure 5 we can find a conflict graph of an instance obtained from the reduction of a formula

$$\forall x_1, x_2, x_3 . \exists y_1, y_2 . (\neg x_1 \vee y_1 \vee x_2) \wedge (\neg x_2 \vee \neg y_2 \vee \neg x_3).$$

We partition the set of all repairs of r_ψ into two (separate) classes:

1. $\mathcal{Y}$-repairs: repairs that contain Y.
2. $\mathcal{X}$-repairs: repairs that don't contain Y.

We will use $\mathcal{X}$- and $\mathcal{Y}$-repairs to 'simulate' all possible valuations of variables $x_1, \ldots, x_n$ and $y_1, \ldots, y_m$ respectively.

$\mathcal{Y}$-repairs

Because of the functional dependency $A_1 \to B_1$ a repair is $\mathcal{Y}$-repair if and only if it contains any of q_j or $\bar{q}_j$. Moreover for any $\mathcal{Y}$-repair r' and for any j either

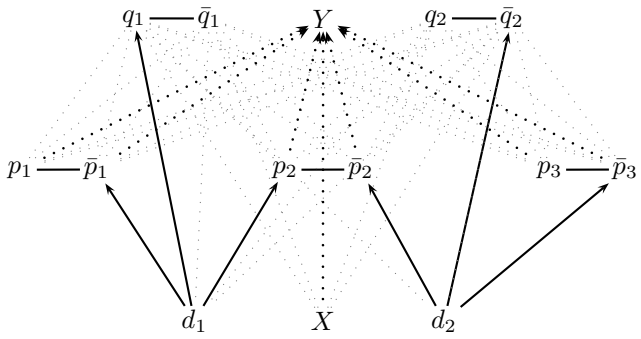

Figure 5: Conflict graph for $\forall x_1, x_2, x_3.\exists y_1, y_2.(\neg x_1 \vee y_1 \vee x_2) \wedge (\neg x_2 \vee \neg y_2 \vee \neg x_3)$ and orientation $\prec_\psi$. The conflicts generated by $A_1 \rightarrow B_1$ are marked with dotted lines.

q_j or $\bar{q}_j$ belongs to r'. Therefore there is one-to-one correspondence between $\mathcal{Y}$-repairs and valuations of y_j variables. To easily move from the world of repairs to the world of valuations and vice versa we define the following two operators (for r' being a $\mathcal{Y}$-repair and V being a valuation of variables in ϕ):

$$V_{\mathcal{Y}}[r'](y_j) = \begin{cases} true & q_j \in r' \\ false & \bar{q}_j \in r' \end{cases} \qquad r_{\mathcal{Y}}[V] = \{q_j | V \models y_j\} \cup \{\bar{q}_j | V \models \neg y_j\} \cup \{Y\}.$$

$\mathcal{X}$-repairs

We will partition further the class of $\mathcal{X}$-repairs depending on their 'conformance' with ϕ. Because $\mathcal{X}$-repairs will correspond only to valuations of x_j we remove any usage of y_j from ψ in the following way:

$$\begin{aligned} \tilde{y}_j = \neg\tilde{y}_j &= false, \\ \tilde{x}_i &= x_i, \\ \neg\tilde{x}_i &= \neg x_i, \\ \tilde{c}_k &= \tilde{l}_{k,1} \vee \tilde{l}_{k,2} \vee \tilde{k}_{j,3}, \\ \tilde{\phi} &= \tilde{c}_1 \wedge \ldots \wedge \tilde{c}_s. \end{aligned}$$

For a given valuation of x_i construct the following set of tuples:

$$r_{\mathcal{X}}[V] = \{p_i | V \models x_i\} \cup \{\bar{p}_i | V \models \neg x_i\} \cup \{d_k | V \not\models \tilde{c}_k\} \cup \{X\}.$$

It's easy to verify that $r_{\mathcal{X}}[V]$ is a $\mathcal{X}$-repair. An $\mathcal{X}$-repair r' is *strict* if and only if there exists a valuation V such that $r' = r_{\mathcal{X}}[V]$. Otherwise the $\mathcal{X}$-repair is *non-strict*.

It's clear that there is a one-to-one correspondence between strict $\mathcal{X}$-repairs and valuations of x_i. Construction of a valuation of x_i from a strict $\mathcal{X}$-repair r' is also straightforward, for technical reasons we extend it to any $\mathcal{X}$-repair:

$$V_{\mathcal{X}}[r'](x_i) = \begin{cases} true & p_i \in r' \\ false & \bar{p}_i \in r' \\ false & \text{otherwise} \end{cases}$$

Note that $\mathcal{X}$-repairs can be characterized in a alternative way:

Proposition 4.5. *A repair of r_ψ is an $\mathcal{X}$-repair if and only if it contains X.*

In the main proof we use only strict X-repairs. The following observation will allow us to remove non-strict repairs from consideration.

Claim 4.6. *Strict $\mathcal{X}$-repairs are $\ll$-maximal $\mathcal{X}$-repairs.*

Proof. First we show how for any non-strict $\mathcal{X}$-repair r' we construct a (strict) $\mathcal{X}$-repair r'' such that $r' \ll r''$. Take the valuation $V = V_{\mathcal{X}}[r']$ and let $r'' = r_{\mathcal{X}}[V]$. The repair r'' is strict and therefore $r' \neq r''$. We show that $r' \ll r''$, i.e.

$$\forall t \in r' \setminus r''. \exists t' \in r'' \setminus r'. t \prec t'.$$

There are three cases of values of t to consider:

1^o $X \in r' \setminus r''$. Implies that r'' is not an $\mathcal{X}$-repair, a contradiction.

2^o For some i we have $p_i \in r' \setminus r''$ or $\bar{p}_i \in r' \setminus r''$. W.l.o.g assume that $p_1 \in r' \setminus r''$. This implies that $V(x_1) = true$. From construction of $r_{\mathcal{X}}[V]$ this implies that $p_1 \in r''$, a contradiction.

3^o For some k we have $d_k \in r' \setminus r''$. W.l.o.g. assume that $k = 1$ and $c_1 = x_1 \vee y_1 \vee \neg x_2$. Then $p_1 \notin r'$ and $\bar{p}_2 \notin r'$ (it's the neighborhood of d_1). From the construction of r'' we have that

$$d_1 \notin r'' \iff V \not\models \tilde{c}_1 \iff V \models x_1 \text{ or } V \models \neg x_2 \iff p_1 \in r'' \text{ or } \bar{p}_2 \in r''.$$

And both p_1 and $\bar{p}_2$ dominate over d_1.

Now, suppose that there exists a strict $\mathcal{X}$-repair r' such that there exists an $\mathcal{X}$-repair r'' preferred over r'. We show that $r' = r''$. Note that r' and r'' must agree on the tuples corresponding to the valuation of variables $x_1, \ldots, x_n$, i.e.

$$r' \cap \{p_1, \bar{p}_1, \ldots, p_n, \bar{p}_n\} = r'' \cap \{p_1, \bar{p}_1, \ldots, p_n, \bar{p}_n\}.$$

Since r' is strict, its content is determined by the corresponding valuation of variables $x_1, \ldots, x_n$. Therefore $r' = r_{\mathcal{X}}[V_{\mathcal{X}}[r'']]$. We showed in the previous part of the proof that $r'' \ll r'$. Since $\prec_\psi$ is acyclic this implies that $r' = r''$. □

Claim 4.7. *For any valuation V of x_i and y_j we have $r_{\mathcal{X}}[V] \ll r_{\mathcal{Y}}[V]$ if and only if $V \models \phi$.*

Proof. We prove implication in two directions:

$\boxed{\Leftarrow}$ By contradiction. Suppose $V \models \phi$ and there exists a tuple t of $r_{\mathcal{X}}[V]$ which is not dominated by any tuple from $r_{\mathcal{Y}}[V]$. Obviously (from dependency $A_1 \rightarrow B_1$) t can be only one of d_k. W.l.o.g. assume that $k = 1$ and $c_1 = x_1 \vee y_1 \vee \neg x_2$. By construction of r_ψ this implies that $p_1 \notin r_{\mathcal{X}}[V]$, $q_1 \notin r_{\mathcal{Y}}[V]$, and $\bar{p}_2 \notin r_{\mathcal{X}}[V]$. From the definition of $r_{\mathcal{X}}[V]$ and $r_{\mathcal{Y}}[V]$ we receive that $V(x_1) = false$, $V(x_2) = true$, and $V(y_1) = false$. This gives us $V \not\models c_i$ which is a contradiction.

$\boxed{\Rightarrow}$ By contradiction. Suppose $r_{\mathcal{X}}[V] \ll r_{\mathcal{Y}}[V]$ and there exists conjunct c_k such that $V \not\models c_k$. W.l.o.g. assume that $k = 1$ and $c_1 = x_1 \vee y_1 \vee \neg x_2$. Then $V(x_1) = false$, $V(x_2) = true$, and $V(y_1) = false$. Consider d_1 and note that it belongs to $r_{\mathcal{X}}[V]$ (by definition of $r_{\mathcal{X}}$). From the construction of $\prec_\phi$ we know that only p_1, $\bar{p}_2$, and q_1 dominate over d_1. $V_{\mathcal{Y}}[V]$ doesn't contain any of those and this gives us a contradiction.

□

Proposition 4.8. *QBF ψ is true if and only if for any strict $\mathcal{X}$-repair r' there exists a $\mathcal{Y}$-repair r'' such that $r' \ll r''$.*

By Claim 4.6 we have that only a $\mathcal{Y}$-repair can be more preferred than a strict $\mathcal{X}$-repair and for any non-strict $\mathcal{X}$-repair there always exists a more preferred repair.

Corollary 4.9. *QBF ψ is true if and only if for any $\mathcal{X}$-repair r' there exists a different repair r'' such that $r' \ll r''$.*

From the partition of repairs we know that $\mathcal{X}$-repairs can be characterized with a formula $\neg R(Y)$.

$$\models \psi \iff \models \forall x_1, \ldots, x_n . \exists y_1, \ldots, y_m . \phi \iff$$
$$\forall r' \in \mathrm{Rep}_F(r_\psi). [r' \models \neg R(Y) \Rightarrow \exists r'' \in \mathrm{Rep}_F(r_\psi). r' \neq r'' \wedge r' \ll r'']. \iff$$
$$\forall r' \in \mathrm{Rep}_F(r_\psi). [\neg \exists r'' \in \mathrm{Rep}_F(r_\psi). r' \neq r'' \wedge r' \ll r''] \Rightarrow r' \models R(Y) \iff$$
$$\forall r' \in \mathrm{GRep}_F^{\prec_\psi}(r_\psi). r' \models R(Y) \iff$$
$$(r_\psi, \prec_\psi) \in D^g_{F,R(Y)}$$

Corollary 4.10. *QBF ψ is true if and only if true is g-preferred consistent answer to $R(Y)$ in r_ψ w.r.t. F and $\prec_\psi$.*

If we use as characterization of $\mathcal{X}$-repairs the formula $R(X)$ then we can reduce QBF to answering to a query with one negated atom.

Corollary 4.11. *QBF ψ is true if and only if true is g-preferred consistent answer to $\neg R(X)$ in r_ψ w.r.t. F and $\prec_\psi$.*

□

4.3 Database cleaning

The postulate $\mathcal{P}4$ allows us to think of a total acyclic priority as a cleaning program — an exact specification of how to resolve all conflicts. To run this program we simply use Algorithm 1 and obtain a unique ℓ-preferred repair. Thanks to Theorem 3.12, this is also the unique g-repair.

Proposition 4.12. *Given a a total acyclic priority $\prec$, the unique ℓ-repair (which if also the unique g-repair) can be computed in time polynomial in the size of the database.*

5 Related work

We limit our discussion to work on using priorities to maintain consistency and facilitate resolution of conflicts.

The first to notice the importance of priorities in information systems is [12]. The authors study there the problem of updates of databases containing propositional sentences. The priority is expressed by storing a natural number with each clause (the integrity constraints should be tagged with the highest priority 0). If an update (inserting or deleting a sentence) leads to inconsistency, among all consistent and realizing the update databases the minimally *different* are selected. A database E is less different than a database F w.r.t. D if either for some $i \in \{0, 1, \ldots, n\}$

$$\begin{cases} D^{i-1} \setminus E^{i-1} = D^{i-1} \setminus F^{i-1}, \\ D^i \setminus E^i \subset D^i \setminus F^i, \end{cases} \qquad \text{or} \qquad \begin{cases} D^n \setminus E^n = D^n \setminus F^n, \\ E \setminus D \subset F \setminus D, \end{cases}$$

where n is the lowest priority in D and D^k consists of all sentences from D with priority less or equal to k. Although this framework does not define a notion of a conflict, we note that more than two facts can create a conflict w.r.t some constraint. For sake of the comparison, assume that the conflicts are generated only by pairs of facts (together with one of the constraints). Then, the selected minimally different consistent databases are equivalent to g-repairs (and because the considered class of priorities has only acyclic extensions it is equivalent to ℓ-repairs). We note, however, that the chosen representation of priorities imposes a significant restriction on the class of considered priorities. In particular it assumes transitivity of the priority on conflicting facts i.e. if facts a, b, and c are pair-wise conflicting and a has a higher priority than b and b has a higher priority than c, then the priority of a is higher than c. This assumption cannot be always fulfilled in the context of inconsistent databases. For example the conflicts between a and b, and between b and c may be caused by violation of one integrity constraints while the conflict between a and c is introduced by a different constraint. While the user may supply us with a rule assigning priorities to conflicts created by the first integrity constraint, the user may not wish to put any priorities on any conflicts created by the other constraint.

A similar representation of priorities used to resolve inconsistency in first-order theories is studied in [6], where the inconsistent set of clauses is stratified

(again the lowest strata has the highest priority). Then preferred maximal consistent subtheories are constructed in a manner analogous to ℓ-repairs. Furthermore, this approach is generalized to priorities being a partial orders, by considering all extensions to weak orders. Again, however, this approach assumes transitivity of priority on conflicts, which as we explained previously may be considered a significant restriction.

In [21] priorities are studied to facilitate the process of *belief revision*. A belief state is represented as an ordered list of propositional formulae and the revision operation simply adds the given sentence at the end of the given belief state. This representation of belief state allows to keep track of revision history, which is later used to impose a preference order on the possible interpretations of the belief state. Only maximally preferred interpretations are used when defining the entailment relation.

In the context of logic programs, priorities among rules can be used to handle inconsistent logic programs (where rules imply contradictory facts). More preferred rules are satisfied, possibly at the cost of violating less important ones. In a manner analogous to $\ll$, [23] lifts a total order on rules to a preference on (extended) answers sets. When computing answers only maximally preferred answers sets are considered.

[22] investigate disjunctive logic programs with priorities on facts. The authors use a transitive and reflexive closure (denoted here $\preceq$) of a user supplied set of priorities on facts. The preference on answer sets $\sqsubseteq$ is defined as follows:

- $X \sqsubseteq X$ for every answer set X
- $X \sqsubseteq Y$ if

$$\exists y \in Y \setminus X. \Big[\exists x \in X \setminus Y. x \preceq y \wedge \neg \exists x' \in X \setminus Y. y \prec x'\Big],$$

  where $x \prec y$ stands for $x \preceq y \wedge y \not\preceq x$.
- if $X \sqsubseteq Y$ and $Y \sqsubseteq Z$, then $X \sqsubseteq Z$.

The answer to a program in the extended framework consists of all maximally preferred answer sets. The main shortcoming of using this framework is it's computational infeasibility (which is specific to decision problems involving general disjunctive programs): computing answers to ground queries to disjunctive prioritized logic programs under cautious (brave) semantics is Π_3^p-complete (resp. Σ_3^p-complete).

A simpler approach to the problem of inconsistent logic programs is presented in [18]. There conflicting facts are removed from the model unless the priority specifies how to resolve the conflict. Because only programs without disjunction are considered, this approach always returns exactly one model of the input program. Constructing preferred repairs in a corresponding fashion (by removing all conflicts unless the priority indicates a resolution) would similarly return exactly one database instance (fulfillment of $\mathcal{P}1$ and $\mathcal{P}4$). However, if the priority does not specify how to resolve every conflict, the returned instance is not a maximal set of tuples and therefore it is not a repair. Such an

approach leads to a loss of (disjunctive) information and violates postulates $\mathcal{P}2$ and $\mathcal{P}3$.

[13] proposes a framework of *conditioned active integrity constraints*, which allows the user to specify the way some of the conflicts can be resolved. This notion syntactically extends the notion of embedded dependency $\forall X.[\phi \supset \exists Y.\psi]$, where X and Y are sets of variables, ϕ and ψ are two conjunctions of literals, and each of existential variables Y is used only once. A conditioned active integrity constraint is obtained by adding a disjunctive list of update atoms ($+C_1, \ldots, +C_k$ for adding, and $-D_{k+1}, \ldots, -D_n$ for deletion) together with conditions $\theta_1, \ldots, \theta_n$ specifying when a corresponding update atom can be used. Such an extended constraint is denoted as

$$\forall X.[(\phi \supset \exists Y.\psi) \supset \theta_1 : +C_1 \vee \ldots \vee \theta_k : +C_k \vee \theta_{k+1} : -D_{k+1} \vee \ldots \vee \theta_n : -D_n]$$

A constraint (or rather its grounded version) is said to be *applied* to by a repair if the original integrity constraint ($\phi \subset \exists Y.\psi$) is satisfied in the database and the repair is obtained by performing updates satisfying the conditional update atom lists (one of the atoms $C_1, \ldots .C_k$ has been added and the corresponding condition $\theta_1, \ldots, \theta_k$ is satisfied, or one of the atoms $C_{k+1}, \ldots, C_n$ has been removed and the corresponding condition $\theta_{k+1}, \ldots, \theta_n$ is satisfied). On all repairs, which are obtained in the standard way by taking as integrity constraints only the heads of the conditioned action integrity constraints, we define relation of preference: a repair r_1 is preferred over r_2 if every (ground) constraint applied in r_1 is also applied in r_2. We note here that when restricted to functional dependencies the set of preferred repairs is a superset of ℓ-repairs. Inclusion in the other direction doesn't always hold, which is illustrated on the following example.

Example 5.1. Consider a database $R(A_1, B_1, A_2, B_2)$ consisting of three tuples $r = \{t_1 = (1, 1, 0, 0), t_2 = (1, 2, 3, 3), t_3 = (0, 0, 3, 4)\}$ and suppose we work in the presence of two functional dependencies $A_1 \rightarrow B_1$ and $A_2 \rightarrow B_2$. Suppose also, that the user specifies that if two tuples are conflicting w.r.t. the FD $A_1 \rightarrow B_1$, then the tuple with higher value of the field B_1 should be preferred when repairing the database. A similar wish is expressed for conflicts generated by the second functional dependency. This can be expressed using the following two conditioned active integrity constraints

$$\begin{aligned}\forall x, y_1, y_2, z_1, z_2, s_1, s_2.[(R(x, y_1, z_1, s_1) \wedge R(x, y_2, z_2, s_2) \supset y_1 \neq y_2) \supset \\ y_1 > y_2 : -R(x, y_2, z_2, s_2)], \\ \forall x_1, x_2, y_1, y_2, z, s_1, s_2.[(R(x_1, y_1, z, s_1) \wedge R(x_2, y_2, z, s_2) \supset s_1 \neq s_2) \supset \\ s_1 > s_2 : -R(x_2, y_2, z, s_2)].\end{aligned}$$

After grounding we remove constraints with their head equal to false and we

obtain the following set

(I1) $R(1,1,0,0) \wedge R(1,2,3,3) \supset 1 > 2 : -R(1,2,3,3),$

(I2) $R(1,2,3,3) \wedge R(1,1,0,0) \supset 2 > 1 : -R(1,1,0,0),$

(I3) $R(1,2,3,3) \wedge R(0,0,3,4) \supset 3 > 4 : -R(0,0,3,4),$

(I4) $R(0,0,3,4) \wedge R(1,2,3,3) \supset 4 > 3 : -R(1,2,3,3).$

The corresponding priority relation is $\prec = \{(t_1, t_2), (t_2, t_3)\}$. Note that in the context of the database r, the user has provided information sufficient to solve all the conflicts, i.e. among the repairs $Rep_F(r) = \{r_1 = \{t_1, t_3\}, r_2 = \{t_2\}\}$ the repair r_1 is the unique repair selected by $LRep_C^{\prec}$. At the same time only (I2) is applied to r_1 and only (I4) is applied to r_2, what makes both repairs incomparable in terms of the framework of [13].

This example also shows that the discussed framework violates the postulate $\mathcal{P}3$. Note also that removing preference information on how to resolve the conflict between t_2 and t_3 will yield only one repair r_1. This shows that this framework violates the postulate $\mathcal{P}4$. At the same time this framework fulfills the property of conservativeness (the preferred repairs are a subset of standard repairs) and non-emptiness (there is always at least one preferred repair). [13] also describes how to translate conditioned active integrity constraints into a prioritized logic program [22], whose preferred models correspond to maximally preferred repairs. Note that the framework of prioritized logic programming is computationally more powerful (answering answers under the brave semantics is Σ_3^p-complete) than required by the problem of finding if an atom is present in any repair (Σ_2^p-complete). It is yet to be seen if less powerful programming environment (like general disjunctive logic programs) can be used to compute preferred answers.

[20] uses ranking functions on tuples to resolve conflicts by taking only the tuple with highest rank and removing others. This approach constructs a unique repair under the assumption that no two different tuples are of equal rank (postulates $\mathcal{P}1$ and $\mathcal{P}4$). If this assumption is not satisfied and the tuples contain numeric values, a new value, called the fusion, can be calculated from the conflicting tuples (then, however, the constructed instance is not a repair in the sense of Definition 2.3).

A different approach based on ranking is studied in [17]. The authors consider polynomial functions that are used to rank repairs. When computing preferred consistent query answers, only repairs with the highest rank are considered. The postulates $\mathcal{P}1$ and $\mathcal{P}2$ are trivially satisfied, but because this form of preference information does not have natural notions of extensions and maximality, it is hard to discuss postulates $\mathcal{P}3$ and $\mathcal{P}4$. Also, the preference among repairs in this method is not based on the way in which the conflicts are resolved.

An approach where the user has a certain degree of control over the way the conflicts are resolved is presented in [16]. Using repair constraints the user can restrict considered repairs to those where tuples from one relation have been removed only if similar tuples have been removed from some other relation. This

approach is monotonic, but not necessarily non-empty. The authors propose method of weakening the repair constraints to restore non-emptiness, however this comes at the price of losing monotonicity.

6 Conclusions and future work

In this paper we proposed a general framework of preferred repairs and preferred consistent query answers by formulating a set of intuitive postulates. We proposed two instantiations of the framework and studied their semantic and computational properties. Table 5 summarizes the computational complexity results; its first row is taken from [9].

	Repair Check	Consistent Answers to	
		$\{\forall, \exists\}$-free queries	conjunctive queries
All repairs	PTIME	PTIME	co-NP-complete
ℓ-repairs	PTIME	co-NP-complete	
g-repairs	co-NP-complete	Π_2^p-complete	

Table 5: Summary of complexity results

We envision several directions for further work. The postulates $\mathcal{P}1$–$\mathcal{P}4$ can be refined, so that only non-trivial instantiations are captured. For example, the following instantiation fulfills the postulates: we ignore any priority which is not total and return all repairs in this case; when the priority is total we return the unique ℓ-repair. This approach, however, is trivial and obviously does not increase the computational complexity of any of considered problems. Also, the computational consequences of further refining the postulates should be examined.

Along the lines of [3], the computational complexity results could be further studied, by assuming a limit on the number of functional dependencies or their conformance with BCNF.

The last is generalization of our framework to broader class of constraints. Conflict graphs can be generalized to hypergraphs [9], which allow to handle broader class of denial constraints. Then, more than two tuples can be involved in a single conflict and the current notion of priority does not have a clear meaning.

References

[1] M. Arenas, L. Bertossi, and J. Chomicki. Consistent Query Answers in Inconsistent Databases. In *ACM Symposium on Principles of Database Systems (PODS)*, pages 68–79, 1999.

[2] M. Arenas, L. Bertossi, and J. Chomicki. Answer Sets for Consistent Query Answering in Inconsistent Databases. *Theory and Practice of Logic Programming*, 3(4–5):393–424, 2003.

[3] M. Arenas, L. Bertossi, J. Chomicki, X. He, V. Raghavan, and J. Spinrad. Scalar Aggregation in Inconsistent Databases. *Theoretical Computer Science*, 296(3):405–434, 2003.

[4] P. Bohannon, M. Flaster, W. Fan, and R. Rastogi. A Cost-Based Model and Effective Heuristic for Repairing Constraints by Value Modification. In *ACM SIGMOD International Conference on Management of Data*, 2005.

[5] L. Bravo and L. E. Bertossi. Logic Programs for Consistently Querying Data Integration Systems. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 10–15, 2003.

[6] G. Brewka. Preferred Subtheories: An Extended Logical Framework for Default Reasoning. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1043–1048, 1989.

[7] A Cali, D. Lembo, and R. Rosati. On the Decidability and Complexity of Query Answering over Inconsistent and Incomplete Databases. In *ACM Symposium on Principles of Database Systems (PODS)*, pages 260–271, 2003.

[8] J. Chomicki. Preference Formulas in Relational Queries. *ACM Transactions on Database Systems*, 28(4):427–466, December 2003.

[9] J. Chomicki and J. Marcinkowski. Minimal-Change Integrity Maintenance Using Tuple Deletions. *Information and Computation*, pages 90–121, 2005.

[10] J. Chomicki, J. Marcinkowski, and S. Staworko. Computing Consistent Query Answers Using Conflict Hypergraphs. In *International Conference on Information and Knowledge Management (CIKM)*, pages 417–426, November 2004.

[11] T. Eiter, M. Fink, G. Greco, and D. Lembo. Efficient Evaluation of Logic Programs for Querying Data Integration Systems. In *International Conference on Logic Programming (ICLP)*, pages 163–177, 2003.

[12] R. Fagin, J. D. Ullman, and M. Y. Vardi. On the Semantics of Updates in Databases. In *ACM Symposium on Principles of Database Systems (PODS)*, pages 352–356, 1983.

[13] S. Flesca, S. Greco, and E. Zumpano. Active Integrity Constraints. In *ACM SIGPLAN International Conference on Principles and Practice of Declarative Programming (PPDP)*, pages 98–107, 2004.

[14] A. Fuxman, E. Fazli, and R. J. Miller. Efficient Management of Inconsistent Databases. In *ACM SIGMOD International Conference on Management of Data*, 2005.

[15] A. Fuxman and R. J. Miller. First-Order Query Rewriting for Inconsistent Databases. In *International Conference on Database Theory (ICDT)*, 2005.

[16] G. Greco and D. Lembo. Data Integration with Preferences Among Sources. In *International Conference on Conceptual Modeling (ER)*, pages 231–244, November 2004.

[17] S. Greco, C. Sirangelo, I. Trubitsyna, and E. Zumpano. Feasibility Conditions and Preference Criteria in Quering and Repairing Inconsistent Databases. In *International Conference on Database and Expert Systems Applications (DEXA)*, pages 44–55, 2004.

[18] B. N. Grosof. Prioritized Conflict Handling for Logic Programs. In *International Logic Programming Symposium*, pages 197–211, 1997.

[19] J. Y. Halpern. Defining Relative Likehood in Partially-Ordered Preferential Structures. *Journal of Artificial Intelligence Research*, 1997.

[20] A. Motro, P. Anokhin, and A. C. Acar. Utility-based Resolution of Data Inconsistencies. In *International Workshop on Information Quality in Information Systems (IQIS)*, pages 35–43. ACM, 2004.

[21] M. Ryan. Belief Revision and Ordered Theory Presentations. In *Logic, Action, and Information*, pages 129–151, 1996.

[22] C. Sakama and K. Inoue. Prioritized logic programming and its application to commonsense reasoning. *Artificial Intelligence*, 123:185–222, 2000.

[23] D. Van Nieuwenborgh and D. Vermeir. Preferred Answer Sets for Ordered Logic Programs. In *European Conference on Logics for Artificial Intelligence (JELIA)*, pages 432–443. Springer-Verlag, LNCS 2424, 2002.

[24] M. Y. Vardi. The Complexity of Relational Query Languages. In *ACM Symposium on Theory of Computing (STOC)*, pages 137–146, 1982.