Detecting Ambiguity in Prioritized Database Repairing

Benny Kimelfeld[1], Ester Livshits[1], and Liat Peterfreund[1]

1 **Technion, Haifa 32000, Israel, {bennyk,esterliv,liatpf}@cs.technion.ac.il**

Abstract

In its traditional definition, a repair of an inconsistent database is a consistent database that differs from the inconsistent one in a "minimal way." Often, repairs are not equally legitimate, as it is desired to prefer one over another; for example, one fact is regarded more reliable than another, or a more recent fact should be preferred to an earlier one. Motivated by these considerations, researchers have introduced and investigated the framework of preferred repairs, in the context of denial constraints and subset repairs. There, a priority relation between facts is lifted towards a priority relation between consistent databases, and repairs are restricted to the ones that are optimal in the lifted sense. Three notions of lifting (and optimal repairs) have been proposed: Pareto, global, and completion.

In this paper we investigate the complexity of deciding whether the priority relation suffices to clean the database unambiguously, or in other words, whether there is exactly one optimal repair. We show that the different lifting semantics entail highly different complexities. Under Pareto optimality, the problem is coNP-complete, in data complexity, for every set of functional dependencies (FDs), except for the tractable case of (equivalence to) one FD per relation. Under global optimality, one FD per relation is still tractable, but we establish Π_2^{p}-completeness for a relation with two FDs. In contrast, under completion optimality the problem is solvable in polynomial time for every set of FDs. In fact, we present a polynomial-time algorithm for arbitrary conflict hypergraphs. We further show that under a general assumption of transitivity, this algorithm solves the problem even for global optimality. The algorithm is extremely simple, but its proof of correctness is quite intricate.

1998 ACM Subject Classification H.2.4 Systems; F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Inconsistent Databases, Preferred Repairs, Data Cleaning, Functional Dependencies, Conflict Hypergraph

Digital Object Identifier 10.4230/LIPIcs.ICDT.2017.5

1 Introduction

Managing database inconsistency has received a lot of attention in the past two decades. Inconsistency arises for different reasons and in different applications. For example, in common applications of Big Data, information is obtained from imprecise sources (e.g., social encyclopedias or social networks) via imprecise procedures (e.g., natural-language processing). It may also arise when integrating conflicting data from different sources (each of which can be consistent). Arenas, Bertossi and Chomicki [3] introduced a principled approach to managing inconsistency, via the notions of *repairs* and *consistent query answering*. Informally, a *repair* of an inconsistent database I is a consistent database J that differs from I in a "minimal" way, where *minimality* refers to the *symmetric difference*. In the case of anti-symmetric integrity constraints (e.g., denial constraints and the special case of functional dependencies), such a repair is a *subset repair* (i.e., J is a consistent subinstance of I that is not properly contained in any consistent subinstance of I).

Various computational problems around database repairs have been extensively investigated. Most studied is the problem of computing the *consistent answers* of a query q on an inconsistent database

I; these are the tuples in the intersection $\bigcap\{q(J) : J \text{ is a repair of } I\}$ [3, 26]. Hence, in this approach inconsistency is handled at query time by returning the tuples that are guaranteed to be in the result no matter which repair is selected. Another well studied question is that of *repair checking* [1]: given instances I and J, determine whether J is a repair of I. Depending on the type of repairs and integrity constraints, these problems may vary from tractable to highly intractable complexity classes [4].

In the above framework, all repairs of a given database instance are taken into account and treated on a par with each other. There are situations, however, in which it is natural to prefer one repair over another [8, 16, 32, 33]. For example, this is the case if one source is regarded more reliable than another (e.g., enterprise data vs. Internet harvesting, precise vs. imprecise sensing equipment, etc.) or if available timestamp information implies that a more recent fact should be preferred over an earlier fact. Recency may be implied not only by timestamps, but also by evolution semantics; for example, "divorced" is likely to be more updated than "single," and similarly is "Sergeant" compared to "Private." (See [15] for a comprehensive study of data quality.) Motivated by these considerations, Staworko, Chomicki and Marcinkowski [32, 33] introduced the framework of *preferred* repairs, where a *priority* relation between conflicting facts distinguishes a set of *preferred* repairs.

Specifically, the notion of *Pareto optimality* and that of *global optimality* are based on two different notions of *improvement*—the property of one consistent subinstance being preferred to another. Improvements are basically lifting of the priority relation from facts to consistent subinstances; J is an improvement of K if $J \setminus K$ contains a fact that is better than all those in $K \setminus J$ (in the Pareto semantics), or if for every fact in $K \setminus J$ there exists a better fact in $J \setminus K$ (in the global semantics). In each of the two semantics, an *optimal repair* is a repair that cannot be improved. A third semantics proposed by Staworko et al. [32] is that of a *completion-optimal* repair, which is a globally optimal repair under some extension of the priority relation into a *total* relation. In this paper, we refer to these preferred repairs as *p-repair*, *g-repair* and *c-repair*, respectively.

Fagin et al. [13] have built on the concept of preferred repairs (in conjunction with the framework of *document spanners* [14]) to devise a language for declaring *inconsistency cleaning* in text information-extraction systems. They have shown there that preferred repairs capture ad-hoc cleaning operations and strategies of some prominent existing systems for text analytics [2, 9].

Staworko et al. [32] showed several results on preferred repairs. For example, every c-repair is also a g-repair, and every g-repair is also a p-repair. They also showed that p-repair and c-repair checking are solvable in polynomial time (under data complexity) in the case of denial constraints, and that there is a set of functional dependencies (FDs) for which g-repair checking is coNP-complete. Later, Fagin et al. [12] extended that hardness result to a full dichotomy in complexity over all sets of FDs: g-repair checking is solvable in polynomial time whenever the set of FDs is equivalent to a single FD or two key constraints per relation; in every other case, the problem is coNP-complete.

While the classic complexity problems studied in the theory of repairs include repair checking and consistent query answering, the presence of repairs gives rise to the *determinism problem*, which Staworko et al. [32] refer to as *categoricity*: determine whether the provided priority relation suffices to clean the database unambiguously, or in other words, decide whether there is exactly one optimal repair. The problem of repairing uniqueness (in a different repair semantics) is also referred to as *determinism* by Fan et al. [18]. In this paper, we study the three variants of this computational problem, under the three optimality semantics Pareto, global and completion, and denote them as *p-categoricity*, *g-categoricity* and *c-categoricity*, respectively.

It is known that under each of the three semantics there is always at least one preferred repair, and Staworko et al. [32] present a polynomial-time algorithm for finding such a repair. (We recall this algorithm in Section 3.) Hence, the categoricity problem is that of deciding whether the output of this algorithm is the only possible preferred repair. As we explain next, it turns out that each of the three variants of the problem entails quite a unique picture of complexity.

For p-categoricity, we focus on integrity constraints that are FDs, and establish the following dichotomy in data complexity. For a relational schema with a set Δ of FDs:

- If Δ associates (up to equivalence) a single FD with every relation symbol, then p-categoricity is solvable in polynomial time.
- In *any other case*, p-categoricity is coNP-complete.

For example, with the relation symbol $R(A, B, C)$ and the FD $A \to B$, p-categoricity is solvable in polynomial time; but if we add the dependency $B \to A$ then it becomes coNP-complete. While there have been several dichotomy results on the complexity of problems associated with inconsistent data [12, 25, 28, 35], to the best of our knowledge this paper is the first to establish a dichotomy result for any variant of repair uniqueness identification.

We then turn to investigating c-categoricity, and establish a far more positive picture than the one for p-categoricity. In particular, the problem is solvable in polynomial time for every set of FDs. In fact, we present an algorithm for solving c-categoricity in polynomial time, assuming that constraints are given as an input *conflict hypergraph* [10] (hence, we establish polynomial-time data complexity for various types of integrity constraints, such as *conditional FDs* [5] and *denial constraints* [19].) The algorithm is extremely simple, yet its proof of correctness is quite intricate.

Finally, we explore g-categoricity. We show that in the tractable case of p-categoricity (equivalence to a single FD per relation), g-categoricity is likewise solvable in polynomial time. For example, $R(A, B, C, D)$ with the dependency $A \to B$ has polynomial-time g-categoricity. Nevertheless, we prove that if we add $C \to D$, then g-categoricity becomes Π_2^{p}-complete. We do not complete a dichotomy as in p-categoricity, and leave that open for future work. Lastly, we ask whether *transitivity* of the preference relation makes a difference. We show that the three *semantics* of repairs remain different in the presence of transitivity, yet quite remarkably, the *problems* g-categoricity and c-categoricity are actually the same. Hence, in the presence of transitivity g-categoricity is solvable in polynomial time (even when constraints are given as a conflict hypergraph).

For lack of space, most of the proofs are excluded and will appear in the full version of the paper.

Related Work

We are not aware of any work on the complexity of categoricity within the prioritized repairing of Staworko et al. [33]. Fagin et al. [13] investigated a static version of categoricity in the context of text extraction, but the settings and problems are fundamentally different, and so are their complexity results (e.g., Fagin et al. [13] establish undecidability results).

In the framework of *data currency* [16], relations consist of entities with attributes, where each entity may appear in different tuples, every time with possibly different (conflicting) attribute values. A partial order of currency is provided on each attribute. A *completion* of an instance is obtained by completing the partial order on an attribute of every entity, and it defines a *current instance* where each attribute takes its most recent value. In addition, a completion needs to satisfy given (denial) constraints, which may introduce interdependencies among completions of different attributes. Fan et al. [16] have studied the problem of determining whether such a specification induces a single current instance (i.e., the corresponding version of categoricity), and showed that this problem is coNP-complete under data complexity. It is not clear how to simulate their hardness in p-categoricity or g-categoricity, since their hardness is due to the constrains on completions, and these constraints do not have correspondents in our case (beyond the partial orders). A similar argument relates our lower bounds to those in the framework of conflict resolution by Fan Geerts [15, Chapter 7.3], where the focus is on establishing a unique tuple from a collection of conflicting tuples.

Fan et al. [16] show that in the absence of constraints, their categoricity problem can be solved in polynomial time. This tractability result can be used for establishing the tractability side of Theorem 5.1 in the special case where the single FD is a key constraint. In the general case of a single

FD, we need to argue about relationships among sets, and moreover, the differences among the three x-categoricity problems matter.

The work on *certain fixes* [17, 18] considers models that are substantially different from the one adopted here, where repairs are obtained by chasing update rules (rather than tuple deletion), and uniqueness applies to chase outcomes (rather than maximal subinstances w.r.t. preference lifting). The problems relevant to our categoricity are the *consistency problem* [18] (w.r.t. guarantees on the consistency of some attributes following certain patterns), and the *determinism* problem [18].

Finally, we remark that there have several dichotomy results on the complexity of problems associated with inconsistent data [12, 25, 28, 35], but to the best of our knowledge this paper is the first to establish a dichotomy result for any variant of repair uniqueness identification. A valid question for future work is whether one can use the techniques of this paper in order to establish a dichotomy in complexity in *any* of the cleaning frameworks studied in past research.

2 Preliminaries

We now present some general terminology and notation that we use throughout the paper.

Signatures and Instances

A (*relational*) *signature* is a finite set $\mathcal{R} = \{R_1, \dots, R_n\}$ of *relation symbols*, each with a designated positive integer as its *arity*, denoted $\mathrm{arity}(R_i)$. We assume an infinite set Const of *constants*, used as database values. An *instance* I over a signature $\mathcal{R} = \{R_1, \dots, R_n\}$ consists of finite relations $R_i^I \subseteq \mathsf{Const}^{\mathrm{arity}(R_i)}$, where $R_i \in \mathcal{R}$. We write $[\![R_i]\!]$ to denote the set $\{1, \dots, \mathrm{arity}(R_i)\}$, and we refer to the members of $[\![R_i]\!]$ as *attributes* of R_i. If I is an instance over $\mathcal{R}$ and $\mathbf{t}$ is a tuple in R_i^I, then we say that $R_i(\mathbf{t})$ is a *fact of* I. By a slight abuse of notation, we identify an instance I with the set of its facts. For example, $R_i(\mathbf{t}) \in I$ denotes that $R_i(\mathbf{t})$ is a fact of I. As another example, $J \subseteq I$ means that $R_i^J \subseteq R_i^I$ for every $R_i \in \mathcal{R}$; in this case, we say that J is *subinstance* of I.

We use the conventional notation R/k to denote that R is a relation symbol of arity k. In our examples we often name the attributes and refer to them by their names. For instance, in Figure 1a we refer to the relation symbol as $\mathit{CompCEO}(\mathrm{company}, \mathrm{ceo})$ where company and ceo refer to Attributes 1 and 2, respectively. In the case of generic relation symbols, we implicitly name their attributes by capital English letters with the corresponding numeric values; for instance, we may refer to Attributes 1, 2 and 3 of $R/3$ by A, B and C, respectively. We stress that attribute names are not part of our formal model, but are rather used for readability.

Integrity and Inconsistency

Let $\mathcal{R}$ be a signature, and I an instance over $\mathcal{R}$. In this paper we consider two representation systems for integrity constraints. The first is *functional dependencies* and the second is *conflict hypergraphs*.

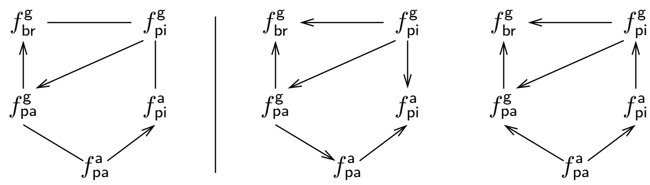
Let $\mathcal{R}$ be a signature. A *Functional Dependency* (*FD* for short) over $\mathcal{R}$ is an expression of the form $R : X \to Y$, where R is a relation symbol of $\mathcal{R}$, and X and Y are subsets of $[\![R]\!]$. When R is clear from the context, we may omit it and write simply $X \to Y$. A special case of an FD is a *key constraint*, which is an FD of the form $R : X \to Y$ where $X \cup Y = [\![R]\!]$. An FD $R : X \to Y$ is *trivial* if $Y \subseteq X$; otherwise, it is *nontrivial*.

When we are using the alphabetic attribute notation, we may write X and Y by simply concatenating the attribute symbols. For example, if we have a relation symbol $R/3$, then $A \to BC$ denotes the FD $R : \{1\} \to \{2, 3\}$. An instance I over R *satisfies* an FD $R : X \to Y$ if for every two facts f and g over R, if f and g agree on (i.e., have the same values for) the attributes of X, then they also agree on the attributes of Y. We say that I satisfies a set Δ of FDs if I satisfies every FD

CompCEO

	company	ceo
$f^{\mathsf{g}}_{\mathsf{pi}}$	Google	Pichai
$f^{\mathsf{g}}_{\mathsf{pa}}$	Google	Page
$f^{\mathsf{g}}_{\mathsf{br}}$	Google	Brin
$f^{\mathsf{a}}_{\mathsf{pa}}$	Alphabet	Page
$f^{\mathsf{a}}_{\mathsf{pi}}$	Alphabet	Pichai

(a) Inconsistent database of the company-CEO running example



(b) The conflict graph $\mathcal{H}^I_{\mathbf{S}}$ and the priority relation $\succ$ for the company-CEO running example (left), and two completions of $\succ$ (middle and right)

Figure 1

in Δ; otherwise, we say that I *violates* Δ. Two sets Δ and Δ' of FDs are *equivalent* if for every instance I over $\mathcal{R}$ it holds that I satisfies Δ if and only if it satisfies Δ'. For example, for $R/3$ the sets $\{A \rightarrow BC, C \rightarrow A\}$ and $\{A \rightarrow C, C \rightarrow AB\}$ are equivalent.

In this work, a *schema* $\mathbf{S}$ is a pair $(\mathcal{R}, \Delta)$, where $\mathcal{R}$ is a signature and Δ is a set of FDs over $\mathcal{R}$. If $\mathbf{S} = (\mathcal{R}, \Delta)$ and $R \in \mathcal{R}$, then we denote by $\Delta_{|R}$ the restriction of Δ to the FDs $R : X \rightarrow Y$ over R.

▶ **Example 2.1.** In our first running example, we use the schema $\mathbf{S} = (\mathcal{R}, \Delta)$, defined as follows. The signature $\mathcal{R}$ consists of a single relation $\mathit{CompCEO}(\mathrm{company}, \mathrm{ceo})$, associating companies with their Chief Executive Officers (CEO). Figure 1a depicts an instance I over $\mathcal{R}$. We define Δ to be $\{\mathrm{company} \rightarrow \mathrm{ceo}\,, \mathrm{ceo} \rightarrow \mathrm{company}\}$, stating that in *CompCEO* each company has a single CEO and each CEO manages a single company. Observe that I violates Δ. For example, Google has three CEOs, Alphabet has two CEOs, and each of Pichai and Page is the CEO of two companies.

While FDs define integrity logically, at the level of the signature, a *conflict hypergraph* [10] provides a direct specification of inconsistencies at the instance level, by explicitly stating sets of facts that cannot co-exist. In the case of FDs, the conflict hypergraph is a graph that has an edge between every two facts that violate an FD. Formally, for an instance I over a signature $\mathcal{R}$, a *conflict hypergraph* $\mathcal{H}$ *(for* I*)* is a hypergraph that has the facts of I as its node set. A subinstance J of I is *consistent* with respect to (w.r.t.) $\mathcal{H}$ if J is an *independent set* of $\mathcal{H}$; that is, no hyperedge of $\mathcal{H}$ is a subset of J. We say that J is *maximal* if $J \cup \{f\}$ is inconsistent for every $f \in I \setminus J$. When all the edges of a conflict hypergraph are of size two, we may call it a *conflict graph*.

Recall that conflict hypergraphs can represent inconsistencies for various types of integrity constraints, including FDs, the more general *conditional FDs* [5], and the more general *denial constraints* [19]. In fact, every constraint that is anti-monotonic (i.e., where subsets of consistent sets are always consistent) can be represented as a conflict hypergraph. In the case of denial constraints, the translation from the logical constraints to the conflict hypergraph can be done in polynomial time under *data complexity* (i.e., when the signature and constraints are assumed to be fixed).

Let $\mathbf{S} = (\mathcal{R}, \Delta)$ be a schema, and let I be an instance over $\mathbf{S}$. Recall that $\mathbf{S}$ is assumed to have only FDs. We denote by $\mathcal{H}^I_{\mathbf{S}}$ the conflict graph for I that has an edge between every two facts that violate some FD of $\mathbf{S}$. Note that a subinstance J of I satisfies Δ if and only if J is consistent w.r.t. $\mathcal{H}^I_{\mathbf{S}}$. As an example, the left graph of Figure 1b depicts the graph $\mathcal{H}^I_{\mathbf{S}}$ for our running example; for now, the reader should ignore the directions on the edges, and view the graph as an undirected one. The following example involves a conflict hypergraph that is not a graph.

▶ **Example 2.2.** In our second running example, we use the toy scenario where the signature has a single relation symbol $\mathit{Follows}/2$, where $\mathit{Follows}(x, y)$ means that person x follows person y (e.g., in a social network). We have two sets of people: a_i for $i = 1, 2, 3$, and b_j for $j = 1, \ldots, 5$. All facts have the form $\mathit{Follows}(\mathsf{a_i}, \mathsf{b_j})$, denoted f_{ij}. The instance I has the following facts: f_{11}, f_{12}, f_{21}, f_{22}, f_{23}, f_{24}, f_{31}, f_{32}, f_{34}, and f_{35}. The hypergraph $\mathcal{H}$ for I encodes the following rules: *(a)* each $\mathsf{a_i}$ can follow at most i people; *and (b)* each $\mathsf{b_j}$ can be followed by at most j people. Specifically, $\mathcal{H}$ contains the following hyperedges:

- $\{f_{11}, f_{12}\}$, $\{f_{21}, f_{22}, f_{23}\}$, $\{f_{21}, f_{22}, f_{24}\}$, $\{f_{21}, f_{23}, f_{24}\}$, $\{f_{22}, f_{23}, f_{24}\}$, $\{f_{31}, f_{32}, f_{34}, f_{35}\}$
- $\{f_{11}, f_{21}\}$, $\{f_{11}, f_{31}\}$, $\{f_{21}, f_{31}\}$, $\{f_{12}, f_{22}, f_{32}\}$

An example of a (maximal) consistent subinstance J is $\{f_{11}, f_{22}, f_{23}, f_{32}, f_{34}, f_{35}\}$.

Prioritizing Inconsistent Databases

We now recall the framework of preferred repairs by Staworko et al. [32]. Let I be an instance over a signature $\mathcal{R}$. A *priority* relation $\succ$ over I is an acyclic binary relation over the facts in I. By *acyclic* we mean that I does not contain any sequence $f_1, \dots, f_k$ of facts with $f_i \succ f_{i+1}$ for all $i = 1, \dots, k-1$ and $f_k \succ f_1$. If $\succ$ is a priority relation over I and K is a subinstance of I, then $\max_{\succ}(K)$ denotes the set of facts $f \in K$ such that no $g \in K$ satisfies $g \succ f$.

An *inconsistent prioritizing instance* over $\mathcal{R}$ is a triple $(I, \mathcal{H}, \succ)$, where I is an instance over $\mathcal{R}$, $\mathcal{H}$ is a conflict hypergraph for I, and $\succ$ is a priority relation over I with the following property: for every two facts f and g in I, if $f \succ g$ then f and g are neighbors in $\mathcal{H}$ (that is, f and g co-occur in some hyperedge).[1] For example, if $\mathcal{H} = \mathcal{H}^I_{\mathsf{S}}$ (where all the constraints in S are FDs), then $f \succ g$ implies that $\{f, g\}$ violates at least one FD.

► **Example 2.3.** We continue our running company-CEO example. We define a priority relation $\succ$ by $f^{\mathsf{g}}_{\mathsf{pi}} \succ f^{\mathsf{g}}_{\mathsf{pa}}$, $f^{\mathsf{g}}_{\mathsf{pa}} \succ f^{\mathsf{g}}_{\mathsf{br}}$ and $f^{\mathsf{a}}_{\mathsf{pa}} \succ f^{\mathsf{a}}_{\mathsf{pi}}$. We denote $\succ$ by corresponding arrows on the left graph of Figure 1b. (Therefore, some of the edges are directed and some are undirected.) We then get the inconsistent prioritizing instance $(I, \mathcal{H}^I_{\mathsf{S}}, \succ)$ over $\mathcal{R}$. Observe that the graph does not contain directed cycles, as required from a priority relation.

► **Example 2.4.** Recall that the instance I of our followers example is defined in Example 2.2. The priority relation $\succ$ is given by $f_{il} \succ f_{jk}$ if one of the following holds: *(a)* $i = j$ and $k = l + 1$, *or (b)* $j = i + 1$ and $l = k$. For example, we have $f_{11} \succ f_{12}$ and $f_{12} \succ f_{22}$. But we do not have $f_{11} \succ f_{22}$ (hence, $\succ$ is not transitive).

Let $(I, \mathcal{H}, \succ)$ be an inconsistent prioritizing instance over a signature $\mathcal{R}$. We say that $\succ$ is *total* if for every two facts f and g in I, if f and g are neighbors in $\mathcal{H}$ then either $f \succ g$ or $g \succ f$. A priority $\succ_c$ over I is a *completion* of $\succ$ (w.r.t. $\mathcal{H}$) if $\succ$ is a subset of $\succ_c$ and $\succ_c$ is total. As an example, the middle and right graphs of Figure 1b are two completions of the priority relation $\succ$ depicted on the left side. A *completion* of $(I, \mathcal{H}, \succ)$ is an inconsistent prioritizing instance $(I, \mathcal{H}, \succ_c)$ where $\succ_c$ is a completion of $\succ$.

Preferred Repairs

Let $D = (I, \mathcal{H}, \succ)$ be an inconsistent prioritizing instance over $\mathcal{R}$. As defined by Arenas et al. [3], J is a *repair* of D if J is a maximal consistent subinstance of I. Staworko et al. [32] define three different notions of *preferred* repairs: *Pareto optimal*, *globally optimal*, and *completion optimal*. The first two are based on checking whether a repair J of I can be improved by replacing a set of facts in J with a "better preferred" set of facts from I; they differ in the way "better preferred" is interpreted. The third notion is based on the concept of completion. Next we give the formal definitions.

[1] This requirement has been made with the introduction of the framework [32]. Obviously, the lower bounds we present hold even without this requirement. Moreover, our main upper bound, Theorem 6.1, holds as well without this requirement. We defer to future work the thorough investigation of the impact of relaxing this requirement.

▶ **Definition 2.5** (Improvement)**.** Let $(I, \mathcal{H}, \succ)$ be an inconsistent prioritizing instance over a signature $\mathcal{R}$, and J and J' be two distinct consistent subinstances of I.

- J is a *Pareto improvement* of J' if there exists $f \in J \setminus J'$ such that $f \succ f'$ for all $f' \in J' \setminus J$.
- J is a *global improvement* of J' if for every $f' \in J' \setminus J$ there exists $f \in J \setminus J'$ such that $f \succ f'$.

That is, J is a Pareto improvement of J' if, to obtain J from J', we insert and delete facts, and one of the inserted facts is preferred to all deleted ones. And J is a global improvement of J' if we similarly obtain J from J', but now for every deleted fact a preferred one is inserted.

▶ **Example 2.6.** We continue the company-CEO running example. We define four consistent subinstances of I: $J_1 = \{f^{\mathsf{g}}_{\mathsf{br}}, f^{\mathsf{a}}_{\mathsf{pi}}\}$, $J_2 = \{f^{\mathsf{g}}_{\mathsf{pa}}, f^{\mathsf{a}}_{\mathsf{pi}}\}$, $J_3 = \{f^{\mathsf{g}}_{\mathsf{br}}, f^{\mathsf{a}}_{\mathsf{pa}}\}$, and $J_4 = \{f^{\mathsf{g}}_{\mathsf{pi}}, f^{\mathsf{a}}_{\mathsf{pa}}\}$. Note the following. First, J_2 is a Pareto improvement of J_1, since $f^{\mathsf{g}}_{\mathsf{pa}} \in J_2 \setminus J_1$ and $f^{\mathsf{g}}_{\mathsf{pa}} \succ f$ for every fact in $J_1 \setminus J_2$ (where in this case there is only one such an f, namely $f^{\mathsf{g}}_{\mathsf{br}}$). Second, J_4 is a global improvement of J_2 because $f^{\mathsf{g}}_{\mathsf{pi}} \succ f^{\mathsf{g}}_{\mathsf{pa}}$ and $f^{\mathsf{a}}_{\mathsf{pa}} \succ f^{\mathsf{a}}_{\mathsf{pi}}$. (We refer to J_3 in later examples.)

We then get the following variants of *preferred repairs*.

▶ **Definition 2.7** (p/g/c-repair)**.** Let D be an inconsistent prioritizing instance $(I, \mathcal{H}, \succ)$, and let J be a consistent subinstance of I. Then J is a:

- *Pareto-optimal repair of D* if there is no Pareto improvement of J.
- *globally-optimal repair of D* if there is no global improvement of J.
- *completion-optimal repair of D* if there exists a completion D_c of D such that J is a globally-optimal repair of D_c.

We abbreviate "Pareto-optimal repair," "globally-optimal repair," and "completion-optimal repair" by *p-repair*, *g-repair* and *c-repair*, respectively.

We remark that in the definition of a completion-optimal repair, we could replace "globally-optimal" with "Pareto-optimal" and obtain an equivalent definition [32].

Let $D = (I, \mathcal{H}, \succ)$ be an inconsistent prioritizing instance over a signature $\mathcal{R}$. We denote the set of all the repairs, p-repairs, g-repairs and c-repairs of D by $\mathrm{Rep}(D)$, $\mathrm{PRep}(D)$, $\mathrm{GRep}(D)$ and $\mathrm{CRep}(D)$, respectively. An easy observation is that when the relation $\succ$ is empty, the four types of repairs coincide. Moreover, the following was shown by Staworko et al. [32].

▶ **Proposition 2.8.** [32] *For all inconsistent prioritizing instances D we have* $\mathrm{CRep}(D) \neq \emptyset$ *and* $\mathrm{CRep}(D) \subseteq \mathrm{GRep}(D) \subseteq \mathrm{PRep}(D) \subseteq \mathrm{Rep}(D)$.

▶ **Example 2.9.** We continue our company-CEO example. Recall the instances J_i defined in Example 2.6. We showed that J_1 has a Pareto improvement, so J_1 is *not* a p-repair (although a repair in the ordinary sense). The reader can verify that J_2 has no Pareto improvements, so J_2 is a p-repair. But J_2 is not a g-repair, as J_4 is a global improvement of J_2. The reader can verify that J_3 is a g-repair (hence, a p-repair). Finally, J_4 is a g-repair w.r.t. the left completion of $\succ$ in Figure 1b (and also w.r.t. the right one). Hence, J_4 is a c-repair (and so a g-repair and a p-repair). In contrast, J_3 has a global improvement (and a Pareto improvement) in both completions; but it does not prove that J_3 is not a c-repair (since, conceptually, one needs to consider all possible completions of $\succ$).

▶ **Example 2.10.** We now continue the follower example. The inconsistent prioritizing instance $(I, \mathcal{H}, \succ)$ is defined in Examples 2.2 and 2.4. Consider the instance $J_1 = \{f_{11}, f_{22}, f_{23}, f_{32}, f_{34}, f_{35}\}$. The reader can verify that J_1 is a c-repair (e.g., by completing $\succ$ through the lexicographic order). The subinstance $J_2 = \{f_{12}, f_{21}, f_{22}, f_{34}, f_{35}\}$ is a repair but not a p-repair, since we can add f_{11} and remove both f_{12} and f_{21}, and thus obtain a Pareto improvement.

3 Categoricity

In this section we define the computational problem of *categoricity*, which is the main problem that we study in this paper. Proposition 2.8 states that, under each of the semantics of preferred repairs, at least one such a repair exists. In general, there can be many possible preferred repairs. The problem of *categoricity* [32] is that of testing whether there is *precisely* one such a repair; that is, there do not exist two distinct preferred repairs, and therefore, the priority relation contains enough information to clean the inconsistent instance unambiguously.

▶ **Problem 3.1.** *The problems* p-categoricity*,* g-categoricity*, and* c-categoricity *are those of testing whether* $|\mathrm{PRep}(D)| = 1$*,* $|\mathrm{GRep}(D)| = 1$ *and* $|\mathrm{CRep}(D)| = 1$*, respectively, given a signature* $\mathcal{R}$ *and an inconsistent prioritizing instance* D *over* $\mathcal{R}$*.*

As defined, categoricity takes as input both the signature $\mathcal{R}$ and the inconsistent prioritizing instance D, where constraints are represented by a conflict hypergraph. We also study this problem from the perspective of *data complexity*; there, we fix a schema $\mathbf{S} = (\mathcal{R}, \Delta)$, where Δ is a set of FDs. In that case, the input consists of an instance I over $\mathcal{R}$ and a priority relation $\prec$ over I. The conflict hypergraph is then implicitly assumed to be $\mathcal{H}^I_{\mathbf{S}}$. We denote the corresponding variants of the problem by p-categoricity$\langle\mathbf{S}\rangle$, g-categoricity$\langle\mathbf{S}\rangle$ and c-categoricity$\langle\mathbf{S}\rangle$, respectively.

▶ **Example 3.2.** Continuing our company-CEO example, we showed in Example 2.9 that there are at least two g-repairs and at least three p-repairs. Hence, a solver for g-categoricity$\langle\mathbf{S}\rangle$ should return false on $(I, \succ)$, and so is a solver for p-categoricity$\langle\mathbf{S}\rangle$. In contrast, we later show that there is precisely one c-repair (Example 6.2); hence, a solver for c-categoricity$\langle\mathbf{S}\rangle$ should return true on $(I, \succ)$. If, on the other hand, we replaced $\succ$ with any of the completions in Figure 1b, then there would be precisely one p-repair and one g-repair (namely, the current single c-repair). This follows from a result of Staworko et al. [32], stating that categoricity holds in the case of total priority relations.

4 Preliminary Insights

We begin with some basic insights into the different variants of the categoricity problem.

Generating an Optimal Repair

We recall an algorithm by Staworko et al. [32] for greedily constructing a c-repair. This is the algorithm FindCRep of Figure 3a. The algorithm takes as input an inconsistent prioritizing instance $(I, \mathcal{H}, \succ)$ and returns a c-repair J. It begins with an empty J, and incrementally inserts tuples to J, as follows. In each iteration of lines 3–6, the algorithm selects a fact f from $\max_{\succ}(I)$ and removes it from I. Then, f is added to J if it does not violate consistency, that is, if $\mathcal{H}$ does not contain any hyperedge e such that $e \subseteq J \cup \{f\}$. The specific way of choosing the fact f among all those in $\max_{\succ}(I)$ is (deliberately) left unspecified, and hence, different executions may result in different c-repairs. In that sense, the algorithm is nondeterministic. Staworko et al. [32] proved that the possible results of these different executions are *precisely* the c-repairs.

▶ **Theorem 4.1.** [32] *Let* $(I, \mathcal{H}, \succ)$ *be an inconsistent prioritizing instance over* $\mathcal{R}$*. Let* J *be a consistent subinstance of* I*. Then* J *is a c-repair if and only if there exists an execution of* FindCRep$(I, \mathcal{H}, \succ)$ *that returns* J*.*

Theorem 4.1, combined with Proposition 2.8, has several implications for us. First, we can obtain an x-repair (where x is either p, g or c) in polynomial time. Hence, if a solver for x-categoricity determines that there is a single x-repair, then we can actually generate that x-repair in polynomial time. Second, c-categoricity is the problem of testing whether $\mathsf{FindCRep}(I, \mathcal{H}, \succ)$ returns the same instance J on every execution. Moreover, due to Proposition 2.8, p-categoricity (resp. g-categoricity) is the problem of testing whether every p-repair (resp. g-repair) is equal to the one that is obtained by some execution of the algorithm.

▶ **Example 4.2.** We consider the application of the algorithm FindCRep to the instance of our company-CEO example (where $\mathcal{H} = \mathcal{H}^I_{\mathbf{S}}$). The following are two different executions: *(1)* $+f^{\mathsf{g}}_{\mathsf{pi}}$, $-f^{\mathsf{g}}_{\mathsf{pa}}$, $-f^{\mathsf{g}}_{\mathsf{br}}$, $+f^{\mathsf{a}}_{\mathsf{pa}}$, $-f^{\mathsf{a}}_{\mathsf{pi}}$, *(2)* $+f^{\mathsf{a}}_{\mathsf{pa}}$, $-f^{\mathsf{a}}_{\mathsf{pi}}$, $+f^{\mathsf{g}}_{\mathsf{pi}}$, $-f^{\mathsf{g}}_{\mathsf{pa}}$, $-f^{\mathsf{g}}_{\mathsf{br}}$. Here, we denote inclusion in J (i.e., the condition of line 5 is true) by plus and exclusion from J by minus. Observe that both executions return $J_4 = \{f^{\mathsf{g}}_{\mathsf{pi}}, f^{\mathsf{a}}_{\mathsf{pa}}\}$. This is on a par with the statement in Example 3.2 that in this running example there is a single c-repair.

Complexity Insights

Our goal is to study the complexity of x-categoricity (where x is g, p and c). This problem is related to that of *x-repair checking*, namely, given $D = (I, \mathcal{H}, \succ)$ and J, determine whether J is an x-repair of D. The following is known about this problem.

▶ **Theorem 4.3.** [12,32] *The following hold.*

- *p-repair checking and c-repair checking are solvable in polynomial time; g-repair checking is in coNP [32].*
- *Let $\mathbf{S} = (\mathcal{R}, \Delta)$ be a fixed schema. If $\Delta_{|R}$ is equivalent to either a single FD or two key constraints for every $R \in \mathcal{R}$, then g-repair checking over $\mathbf{S}$ is solvable in polynomial time; otherwise, g-repair checking over $\mathbf{S}$ is coNP-complete [12].*

Recall from Proposition 2.8 that there is always at least one x-repair. Therefore, given $(I, \mathcal{H}, \succ)$ we can solve the problem x-categoricity using a coNP algorithm with an oracle to x-repair checking: for all two distinct subinstances J_1 and J_2, either J_1 or J_2 is not an x-repair. Therefore, from Theorem 4.3 we conclude the following.

▶ **Corollary 4.4.** *The following hold.*

- *p-categoricity and c-categoricity are in coNP, and g-categoricity is in Π^{p}_2.*
- *For all fixed schemas $\mathbf{S} = (\mathcal{R}, \Delta)$, g-categoricity$\langle\mathbf{S}\rangle$ is in Π^{p}_2, and moreover, if $\Delta_{|R}$ is equivalent to either a single FD or two key constraints for every $R \in \mathcal{R}$ then g-categoricity$\langle\mathbf{S}\rangle$ is in coNP.*

We stress here that if x-categoricity is solvable in polynomial time, then x-categoricity$\langle\mathbf{S}\rangle$ is solvable in polynomial time for *all* schemas $\mathbf{S}$; this is true since for every fixed schema $\mathbf{S}$ the hypergraph $\mathcal{H}^I_{\mathbf{S}}$ can be constructed in polynomial time, given I. Similarly, if x-categoricity$\langle\mathbf{S}\rangle$ is coNP-hard (resp. Π^{p}_2-hard) for *at least one* $\mathbf{S}$, then x-categoricity is coNP-hard (resp. Π^{p}_2-hard).

When we are considering x-categoricity$\langle\mathbf{S}\rangle$, we assume that all the integrity constraints are FDs. Therefore, unlike the general problem of x-categoricity, in x-categoricity$\langle\mathbf{S}\rangle$ conflicting facts always belong to the same relation. It thus follows that our analysis for x-categoricity$\langle\mathbf{S}\rangle$ can be restricted to single-relation schemas. Formally, we have the following.

▶ **Proposition 4.5.** *Let $\mathbf{S} = (\mathcal{R}, \Delta)$ be a schema and x be one of p, g and c. For each relation $R \in \mathcal{R}$, let $\mathbf{S}_{|R}$ be the schema $(\{R\}, \Delta_{|R})$.*

- *If x-categoricity$\langle\mathbf{S}_{|R}\rangle$ is solvable in polynomial time for every $R \in \mathcal{R}$, then x-categoricity$\langle\mathbf{S}\rangle$ is solvable in polynomial time.*
- *If x-categoricity$\langle\mathbf{S}_{|R}\rangle$ is coNP-hard (resp. Π^{p}_2-hard) for at least one $R \in \mathcal{R}$, then x-categoricity$\langle\mathbf{S}\rangle$ is coNP-hard (resp. Π^{p}_2-hard).*

Observe that the phenomenon of Proposition 4.5 *does not* hold for general x-categoricity (where conflicts are given by a conflict hypergraph), since hyperedges may cross relations.

In the following sections we investigate each of the three variants of categoricity: p-categoricity (Section 5), c-categoricity (Section 6) and g-categoricity (Section 7).

5 p-Categoricity

In this section we prove a dichotomy in the complexity of p-categoricity$\langle \mathbf{S} \rangle$ over all schemas $\mathbf{S}$ (where Δ consists of FDs). This dichotomy states that the only tractable case is where the schema associates a single FD (which can be trivial) to each relation symbol, up to equivalence. In all other cases, p-categoricity$\langle \mathbf{S} \rangle$ is coNP-complete. Formally, we prove the following.

▶ **Theorem 5.1.** *Let $\mathbf{S} = (\mathcal{R}, \Delta)$ be a schema. The problem p-categoricity$\langle \mathbf{S} \rangle$ can be solved in polynomial time if $\Delta|_R$ is equivalent to a single FD for every $R \in \mathcal{R}$. In every other case, p-categoricity$\langle \mathbf{S} \rangle$ is coNP-complete.*

The tractability side of Theorem 5.1 is fairly simple to prove. The proof of the hardness side is involved, and we outline it in the rest of this section. Due to Proposition 4.5, it suffices to consider schemas $\mathbf{S}$ with a single relation, which is what we do in the remainder of this section.

5.1 Proof of Hardness

Our proof is based on the concept of a *fact-wise reduction* [23], which is formally defined as follows. Let $\mathbf{S} = (\mathcal{R}, \Delta)$ and $\mathbf{S}' = (\mathcal{R}', \Delta')$ be two schemas. A *mapping* from $\mathcal{R}$ to $\mathcal{R}'$ is a function μ that maps facts over $\mathcal{R}$ to facts over $\mathcal{R}'$. We naturally extend a mapping μ to map instances J over $\mathcal{R}$ to instances over $\mathcal{R}'$ by defining $\mu(J)$ to be $\{\mu(f) \mid f \in J\}$. A *fact-wise reduction* from $\mathbf{S}$ to $\mathbf{S}'$ is a mapping Π from $\mathcal{R}$ to $\mathcal{R}'$ with the following properties: *(a)* Π is injective, that is, for all facts f and g over $\mathcal{R}$, if $\Pi(f) = \Pi(g)$ then $f = g$; *(b)* Π preserves consistency and inconsistency, that is, for every instance J over $\mathbf{S}$, the instance $\Pi(J)$ satisfies Δ' if and only if J satisfies Δ; *and (c)* Π is computable in polynomial time.

Let $\mathbf{S}$ and $\mathbf{S}'$ be two schemas, and let Π be a fact-wise reduction from $\mathbf{S}$ to $\mathbf{S}'$. Given an inconsistent instance I over $\mathbf{S}$ and a priority relation $\succ$ over I, we denote by $\Pi(\succ)$ the priority relation $\succ'$ over $\Pi(I)$ where $\Pi(f) \succ' \Pi(g)$ if and only if $f \succ g$. If D is the inconsistent prioritizing instance $(I, \mathcal{H}^I_{\mathbf{S}}, \succ)$, then we denote by $\Pi(D)$ the triple $(\Pi(I), \mathcal{H}^{\Pi(I)}_{\mathbf{S}'}, \Pi(\succ))$, which is also an inconsistent prioritizing instance. The usefulness of fact-wise reductions is due to the following proposition, which is straightforward.

▶ **Proposition 5.2.** *Let $\mathbf{S}$ and $\mathbf{S}'$ be two schemas, and suppose that Π is a fact-wise reduction from $\mathbf{S}$ to $\mathbf{S}'$. Let I be an inconsistent instance over $\mathbf{S}$, $\succ$ a priority relation over I, and D the inconsistent prioritizing instance $(I, \mathcal{H}^I_{\mathbf{S}}, \succ)$. Then there is a bijection between $\mathrm{PRep}(D)$ and $\mathrm{PRep}(\Pi(D))$.*

We then conclude the following corollary.

▶ **Corollary 5.3.** *If there is a fact-wise reduction from $\mathbf{S}$ to $\mathbf{S}'$, then there is a polynomial-time reduction from p-categoricity$\langle \mathbf{S} \rangle$ to p-categoricity$\langle \mathbf{S}' \rangle$.*

Specific Schemas

In the proof we consider seven specific schemas. The importance of these schemas will later become apparent. We denote these schemas by $\mathbf{S}^i$, for $i = 0, 1, \dots, 6$, where each $\mathbf{S}^i$ is the schema $(\mathcal{R}^i, \Delta^i)$, and $\mathcal{R}^i$ is the singleton $\{R^i\}$. The specification of the $\mathbf{S}^i$ is as follows.

$R^0/2$ and $\Delta^0 = \{A \to B, B \to A\}$	$R^1/3$ and $\Delta^1 = \{AB \to C, BC \to A, AC \to B\}$
$R^2/3$ and $\Delta^2 = \{A \to B, B \to A\}$	$R^3/3$ and $\Delta^3 = \{AB \to C, C \to B\}$
$R^4/3$ and $\Delta^4 = \{A \to B, B \to C\}$	$R^5/3$ and $\Delta^5 = \{A \to C, B \to C\}$
$R^6/3$ and $\Delta^6 = \{\emptyset \to A, B \to C\}$	

(For $\mathbf{S}^6$, recall that $\emptyset \to A$ denotes the FD $\emptyset \to \{1\}$, that is, facts should have the same value on the first attribute.) The proof uses fact-wise reductions from the $\mathbf{S}^i$, as we explain in the next section.

Two Hard Schemas

Our proof boils down to proving coNP-hardness for two specific schemas, namely $\mathbf{S}^0$ and $\mathbf{S}^6$, and then using (known and new) fact-wise reductions in order to cover all the other schemas. For $\mathbf{S}^6$ the proof is fairly simple. However, hardness for $\mathbf{S}^0$ turned out to be highly challenging to prove, and in fact, this part is the hardest in the proof of Theorem 5.1. Note that $\mathbf{S}^0$ is the schema of our company-CEO running example (introduced in Example 2.1).

▶ **Theorem 5.4.** *The problems p-categoricity$\langle \mathbf{S}^0 \rangle$ and p-categoricity$\langle \mathbf{S}^6 \rangle$ are both coNP-hard.*

Applying Fact-Wise Reductions

The following has been proved by Fagin et al. [12].

▶ **Theorem 5.5.** [12] *Let $\mathbf{S} = (\mathcal{R}, \Delta)$ be a schema such that $\mathcal{R}$ consists of a single relation symbol. Suppose that Δ is equivalent to neither any single FD nor any pair of keys. Then there is a fact-wise reduction from some $\mathbf{S}^i$ to $\mathbf{S}$, where $i \in \{1, \dots, 6\}$.*

We complete the proof using the following two lemmas, giving additional fact-wise reductions.

▶ **Lemma 5.6.** *Let $\mathbf{S} = (\mathcal{R}, \Delta)$ be a schema such that $\mathcal{R}$ consists of a single relation symbol. Suppose that Δ is equivalent to a pair of key constraints, and Δ is not equivalent to any single FD. Then there is a fact-wise reduction from $\mathbf{S}^0$ to $\mathbf{S}$.*

▶ **Lemma 5.7.** *For all $i = 1, \dots, 5$ there is a fact-wise reduction from $\mathbf{S}^0$ to $\mathbf{S}^i$.*

The structure of our fact-wise reductions is depicted in Figure 2. Dashed edges are known fact-wise reductions, while solid edges are new. Observe that each single-relation schema on the hardness side of Theorem 5.1 has an ingoing path from either $\mathbf{S}^0$ or $\mathbf{S}^6$, both shown to have coNP-hard p-categoricity (Theorem 5.4).

6 c-Categoricity

We now investigate the complexity of c-categoricity. Our main result is the following.

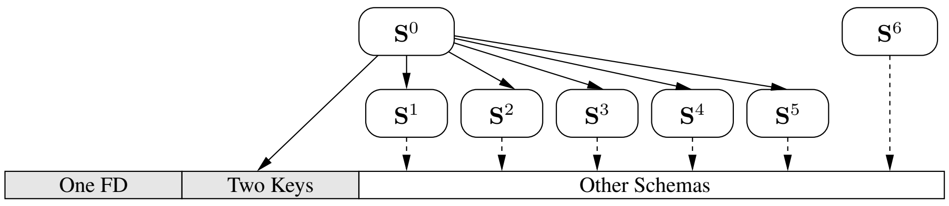


Figure 2 The structure of fact-wise reductions for proving the hardness side of the dichotomy of Theorem 5.1

Algorithm FindCRep$(I, \mathcal{H}, \succ)$

```
1: $J := \emptyset$
2: while $\max_{\succ}(I) \neq \emptyset$ do
3:     choose a fact $f$ in $\max_{\succ}(I)$
4:     $I := I \setminus \{f\}$
5:     if $J \cup \{f\}$ is consistent w.r.t. $\mathcal{H}$ then
6:         $J := J \cup \{f\}$
7: return $J$
```

(a) Finding a c-repair [32]

Algorithm CCategoricity$(I, \mathcal{H}, \succ)$

```
1: $i := 0$
2: $J := \emptyset$
3: while $I \neq \emptyset$ do
4:     $i := i + 1$
5:     $P_i := \max_{\succ^+}(I)$
6:     $J := J \cup P_i$
7:     $N_i := \{f \in I \mid \mathcal{H}$ has a hyperedge $e$ s.t.
       $f \in e$, $(e \setminus \{f\}) \subseteq J$, and $(e \setminus \{f\}) \succ^+ f\}$
8:     $I := I \setminus (P_i \cup N_i)$
9: return true iff $J$ is consistent
```

(b) Algorithm for c-categoricity

Figure 3 Algorithms for the completion semantics

▶ **Theorem 6.1.** *The c-categoricity problem is solvable in polynomial time.*

In the remainder of this section we establish Theorem 6.1 by presenting a polynomial-time algorithm (Figure 3b). The algorithm is very simple, but its proof of correctness is intricate.

To present our algorithm, some notation is required. Let $(I, \mathcal{H}, \succ)$ be an inconsistent prioritizing instance. The *transitive closure* of $\succ$, denoted $\succ^+$, is the priority relation over the facts of I where for every two facts f and g it holds that $f \succ^+ g$ if and only if there exists a sequence $f_0, \dots, f_m$ of facts, where $m > 0$, such that $f = f_0$, $f_m = g$, and $f_i \succ f_{i+1}$ for all $i = 0, \dots, m-1$. Obviously, $\succ^+$ is acyclic (since $\succ$ is acyclic). Yet unlike $\succ$, the relation $\succ^+$ may compare between facts that are not necessarily neighbors in $\mathcal{H}$. Let $(I, \mathcal{H}, \succ)$ be an inconsistent prioritizing instance, let K be a set of facts of I, and let f be a fact of I. By $K \succ^+ f$ we denote the case where $g \succ^+ f$ for *every* fact $g \in K$.

The algorithm is depicted in Figure 3b. The input is $(I, \mathcal{H}, \succ)$, an inconsistent prioritizing instance. (The signature $\mathcal{R}$ is not needed by the algorithm.) The algorithm incrementally constructs a subinstance J of I, starting with an empty J. Later we will prove that there is a single c-repair if and only if J is consistent; and in that case, J is the single c-repair. The loop in the algorithm constructs fact sets $P_1, \dots, P_t$ and $N_1, \dots, N_t$ (where t is the total number of iterations). Each P_i is called a *positive stratum* and each N_i is called a *negative stratum*. Both P_i and N_i are constructed in the ith iteration. On that iteration we add to J every fact in P_i, and remove from I every fact in P_i and every fact in N_i. The sets P_i and N_i are defined as follows.

- P_i consists of the maximal facts in the current I, according to $\succ^+$.
- N_i consists of all the facts f that, together with $P_1 \cup \dots \cup P_i$, complete a hyperedge of preferred facts; that is, $\mathcal{H}$ has a hyperedge that contains f, is contained in $P_1 \cup \dots \cup P_i \cup \{f\}$, and satisfies $g \succ^+ f$ for every incident $g \neq f$.

The algorithm continues to iterate until I gets empty. As said above, in the end the algorithm returns true if J is consistent, and otherwise false. Next, we give execution examples.

▶ **Example 6.2.** Consider $(I, \mathcal{H}, \succ)$ from our company-CEO running example, illustrated on the left side of Figure 1b. The algorithm makes a single iteration on this instance, where $P_1 = \{f^{\mathsf{g}}_{\mathsf{pi}}, f^{\mathsf{a}}_{\mathsf{pa}}\}$ and $N_1 = \{f^{\mathsf{g}}_{\mathsf{pa}}, f^{\mathsf{a}}_{\mathsf{pi}}, f^{\mathsf{g}}_{\mathsf{pa}}\}$. Both $f^{\mathsf{g}}_{\mathsf{pi}}$ and $f^{\mathsf{a}}_{\mathsf{pa}}$ are in P_1 since both are maximal. Also, each of $f^{\mathsf{g}}_{\mathsf{pa}}$, $f^{\mathsf{a}}_{\mathsf{pi}}$ and $f^{\mathsf{g}}_{\mathsf{pa}}$ is in conflict with P_1, and we have $f^{\mathsf{g}}_{\mathsf{pi}} \succ f^{\mathsf{g}}_{\mathsf{pa}}$, $f^{\mathsf{a}}_{\mathsf{pa}} \succ f^{\mathsf{a}}_{\mathsf{pi}}$, and $f^{\mathsf{g}}_{\mathsf{pi}} \succ^+ f^{\mathsf{g}}_{\mathsf{br}}$.

▶ **Example 6.3.** Now consider the inconsistent prioritizing instance $(I, \mathcal{H}, \succ)$ from our followers running example. Figure 4a illustrates the execution of the algorithm, where each column describes

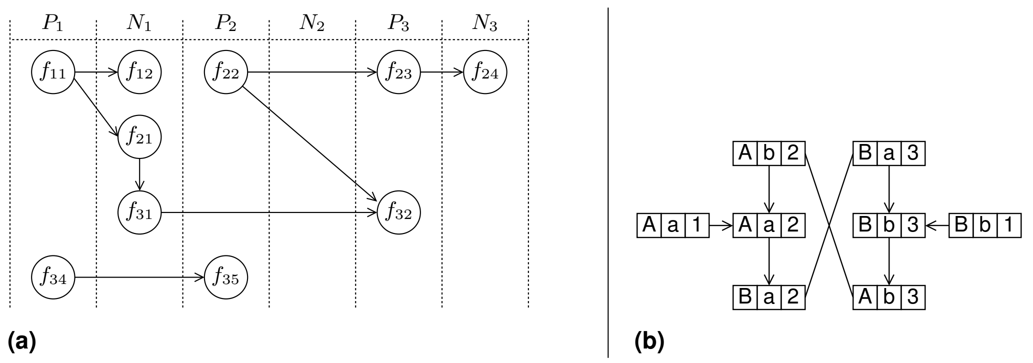


Figure 4 (a) Execution of CCategoricity on the followers example; (b) An inconsistent instance I over $\mathbf{S}^1$ with a priority relation $\succ$ over I

P_i or N_i, from left to right in the order of their construction. For convenience, the priority relation $\succ$, as defined in Example 2.4, is depicted in Figure 4a using corresponding edges between the facts.

On iteration 1, for instance, we have $P_1 = \{f_{11}, f_{34}\}$, since f_{11} and f_{34} are the facts without incoming edges on Figure 4a. Moreover, we have $N_1 = \{f_{12}, f_{21}, f_{31}\}$. The reason why N_1 contains f_{12}, for example, is that $\{f_{11}, f_{12}\}$ is a hyperedge, the fact f_{11} is in P_1, and $f_{11} \succ f_{12}$ (hence, $f_{11} \succ^+ f_{12}$). For a similar reason N_1 contains f_{21}. Fact f_{31} is in N_1 as $\{f_{11}, f_{31}\}$ is a hyperedge, and though $f_{11} \not\succ f_{31}$, we have $f_{11} \succ^+ f_{31}$. As another example, N_3 contains f_{24} since $\mathcal{H}$ has the hyperedge $\{f_{22}, f_{23}, f_{24}\}$, the set $\{f_{22}, f_{23}\}$ is contained in $P_1 \cup P_2 \cup P_3$, and $\{f_{22}, f_{23}\} \succ^+ f_{24}$.

In the end, $J = \{f_{11}, f_{22}, f_{23}, f_{32}, f_{34}, f_{35}\}$, which is also the subinstance J_1 of Example 2.10. Since J is consistent, the algorithm will determine that there is a single c-repair, and that c-repair is J.

▶ **Example 6.4.** We now give an example of an execution on a negative instance of c-categoricity. (In Section 7 we refer to this example for a different reason.) Figure 4b shows an instance I over $\mathbf{S}^1$, which is defined in Section 5.1. Recall that in this schema every two attributes form a key. Each fact $R^1(a_1, a_2, a_3)$ in I is depicted by a tuple that consists of the three values. For example, I contains the (conflicting) facts $R^1(\mathsf{A}, \mathsf{a}, 1)$ and $R^1(\mathsf{A}, \mathsf{a}, 2)$. Hereon, we write Xyi instead of $R^1(\mathsf{X}, \mathsf{y}, \mathsf{i})$. The priority relation $\succ$ is given by the directed edges between the facts; for example, Aa1 $\succ$ Aa2. Undirected edges are between conflicting facts that are incomparable by $\succ$ (e.g., Ab2 and Ab3).

The execution of the algorithm on $(I, \mathcal{H}^I_{\mathbf{S}_1}, \succ)$ is as follows. On the first iteration, $P_1 = \{\mathsf{Aa1}, \mathsf{Ab2}, \mathsf{Ba3}, \mathsf{Bb1}\}$ and $N_1 = \{\mathsf{Aa2}, \mathsf{Bb3}\}$. In particular, note that N_1 does not contain Ba2 since it conflicts only with Ba3 in P_1, but the two are incomparable. Similarly, N_1 does not contain Ab3 since it is incomparable with Ab2. Consequently, in the second iteration we have $P_2 = \{\mathsf{Ba2}, \mathsf{Ab3}\}$ and $N_2 = \emptyset$. In the end, $J = P_1 \cup P_2$ is inconsistent, and therefore, the algorithm will return false. Indeed, the reader can easily verify that each of the following is a c-repair: $\{\mathsf{Aa1}, \mathsf{Ab2}, \mathsf{Ba3}, \mathsf{Bb1}\}$, $\{\mathsf{Aa1}, \mathsf{Ab2}, \mathsf{Ba2}, \mathsf{Bb1}\}$, and $\{\mathsf{Aa1}, \mathsf{Ba3}, \mathsf{Ab3}, \mathsf{Bb1}\}$.

Correctness of CCategoricity is stated in the following theorem.

▶ **Theorem 6.5.** *Let $(I, \mathcal{H}, \succ)$ be an inconsistent prioritizing instance, and let J be the subinstance of I constructed in the execution of* CCategoricity$(I, \mathcal{H}, \succ)$*. Then J is consistent if and only if there is a single c-repair. Moreover, if J is consistent then J is the single c-repair.*

Theorem 6.5, combined with the observation that the algorithm CCategoricity terminates in polynomial time, implies Theorem 6.1. As previously said, the proof of Theorem 6.5 is quite involved. The "only if" direction is that of *soundness*—if the algorithm returns true then there is precisely one c-repair. The other direction is that of *completeness*—if there is precisely one c-repair then the algorithm returns true. Soundness is the easier direction to prove, and we do not discuss the proof here. Proving completeness is more involved. We assume, by way of contradiction, that the constructed J is inconsistent. We are looking at the first positive stratum P_i such that $P_1 \cup \dots \cup P_i$ contains a hyperedge. Then, the crux of the proof is in showing that we can then construct two c-repairs using the algorithm FindCRep: one contains some fact from P_i and another one does not contain that fact. We then establish that there are at least two c-repairs, hence a contradiction.

7 g-Categoricity

We now investigate the complexity of g-categoricity. We begin with a tractability result. Recall from Theorem 5.1 that, assuming P $\neq$ NP, the problem p-categoricity$\langle \mathbf{S} \rangle$ is solvable in polynomial time if and only if $\mathbf{S}$ consists (up to equivalence) of a single FD per relation. The proof works for g-categoricity$\langle \mathbf{S} \rangle$, so the tractable schemas of p-categoricity remain tractable for g-categoricity.

▶ **Theorem 7.1.** *Let $\mathbf{S} = (\mathcal{R}, \Delta)$ be a schema. The problem g-categoricity$\langle \mathbf{S} \rangle$ can be solved in polynomial time if $\Delta|_R$ is equivalent to a single FD for every $R \in \mathcal{R}$.*

It is left open whether there is any schema $\mathbf{S}$ that is not as in Theorem 7.1 where g-categoricity$\langle \mathbf{S} \rangle$ is solvable in polynomial time. In the next section we give an insight into this open problem.

7.1 Intractable Schemas

Our next result shows that g-categoricity$\langle \mathbf{S} \rangle$ hits a harder complexity class than p-categoricity$\langle \mathbf{S} \rangle$. In particular, while p-categoricity$\langle \mathbf{S} \rangle$ is always in coNP (due to Corollary 4.4), we will show a schema $\mathbf{S}$ where g-categoricity$\langle \mathbf{S} \rangle$ is Π_2^{p}-complete. This schema is the schema $\mathbf{S}^6$ from Section 5.1.

▶ **Theorem 7.2.** *g-categoricity$\langle \mathbf{S}^6 \rangle$ is Π_2^{p}-complete.*

The proof of Theorem 7.2 is by a reduction from the Π_2^{p}-complete problem QCNF$_2$: Given a CNF formula $\psi(\mathbf{x}, \mathbf{y})$, determine whether it is the case that for every truth assignment to $\mathbf{x}$ there exists a truth assignment to $\mathbf{y}$ such that the two assignments satisfy ψ.

We can generalize Theorem 7.2 to a broad set of schemas, by using fact-wise reductions from $\mathbf{S}^6$. This is done in the following theorem.

▶ **Theorem 7.3.** *Let $\mathbf{S} = (\mathcal{R}, \Delta)$ be a schema such that $\mathcal{R}$ consists of a single relation symbol R and Δ consists of two nontrivial FDs $X \to Y$ and $W \to Z$. Suppose that each of W and Z contains an attribute that is in none of the other three sets. Then g-categoricity$\langle \mathbf{S} \rangle$ is Π_2^{p}-complete.*

As an example, recall that in $\mathbf{S}^6$ we have $\Delta = \{\emptyset \to A, B \to C\}$. This schema is a special case of Theorem 7.3, since we can use $\emptyset \to A$ as $X \to Y$ and $B \to C$ as $W \to Z$; and indeed, each of W and Z contains an attribute (namely B and C, respectively) that is not in any of the other three sets. Additional examples that satisfy the conditions of Theorem 7.3 (and hence the corresponding g-categoricity$\langle \mathbf{S} \rangle$ is Π_2^{p}-complete) are the following: $\{A \to B, C \to D\}$, $\{A \to C, AB \to CD\}$, $\{A \to B, ABC \to D\}$, and $\{A \to B, C \to ABD\}$. All of these sets are over a relation symbol $R/4$. (And in each of these sets, the first FD corresponds to $X \to Y$ and the second to $W \to Z$.)

Unlike $\mathbf{S}^6$, to this day we do not know what is the complexity of g-categoricity$\langle \mathbf{S}^i \rangle$ for any of the other $\mathbf{S}^i$ (defined in Section 5.1). This includes $\mathbf{S}^0$, for which all we know is membership in coNP (as stated in Corollary 4.4). However, except for this open problem, the proof technique of Theorem 5.1 is valid for g-categoricity$\langle \mathbf{S} \rangle$. Consequently, we can show the following.

▶ **Theorem 7.4.** *The following are equivalent.*

- *g-categoricity$\langle \mathbf{S}^0 \rangle$ is coNP-hard.*

- *g-categoricity*$\langle \mathbf{S} \rangle$ *is coNP-hard for every schema* $\mathbf{S}$ *that falls outside the polynomial-time cases of Theorem 7.1.*

7.2 Transitive Priority

Let $(I, \mathcal{H}, \succ)$ be an inconsistent prioritizing instance. We say that $\succ$ is *transitive* if for every two facts f and g in I, if f and g are neighbors in $\mathcal{H}$ and $f \succ^{+} g$, then $f \succ g$. Transitivity is a natural assumption when $\succ$ is interpreted as a partial order such as "is of better quality than" or "is more current than." In this section we consider g-categoricity in the presence of this assumption. The following example shows that a g-repair is not necessarily a c-repair, even if $\succ$ is transitive. This example provides an important context for the results that follow.

▶ **Example 7.5.** Consider again I and $\succ$ from Example 6.4 (depicted in Figure 4b). Observe that $\succ$ is transitive. In particular, there is no priority between Ab2 and Ba2, even though Ab2 $\succ^{+}$ Ba2, because Ab2 and Ba2 are not in conflict (or, put differently, they are not neighbors in $\mathcal{H}^{I}_{\mathbf{S}^1}$). Consider the subinstance $J = \{\mathsf{Aa1}, \mathsf{Ba2}, \mathsf{Ab3}, \mathsf{Bb1}\}$ of I. The reader can verify that J is a g-repair, but not a c-repair (since no execution of FindCRep can generate J).

Example 7.5 shows that global and completion optimality are different notions, even if the priority is transitive. Yet, quite remarkably, in the presence of transitivity the two coincide on categoricity.

▶ **Theorem 7.6.** *Let* $D = (I, \mathcal{H}, \succ)$ *be an inconsistent prioritizing instance such that* $\succ$ *is transitive.* $|\mathrm{CRep}(D)| = 1$ *if and only if* $|\mathrm{GRep}(D)| = 1$.

Proof. The "if" direction follows from Proposition 2.8, since every c-repair is also a g-repair. The proof of the "only if" direction is based on the special structure of the c-repair, as established in Section 6, in the case where only one c-repair exists. Specifically, suppose that there is a single c-repair J and let $J' \neq J$ be a consistent subinstance of I. We need to show that J' has a global improvement. We claim that J is a global improvement of J'. This is clearly the case if $J' \subseteq J$. So suppose that $J' \not\subseteq J$. Let f' be a fact in $J' \setminus J$. We need to show that there is a fact $f \in J \setminus J'$ such that $f \succ f'$. We complete the proof by finding such an f.

Recall from Theorem 6.5 that J is the result of executing CCategoricity$(I, \mathcal{H}, \succ)$. Consider the positive strata P_i and the negative strata N_j constructed in that execution. Since J is the union of the positive strata, we get that f' necessarily belongs to a negative stratum, say N_j. From the definition of N_j it follows that $\mathcal{H}$ has a hyperedge e such that $f' \in e$, $(e \setminus \{f'\}) \subseteq P_1 \cup \cdots \cup P_j$, and $(e \setminus \{f'\}) \succ^{+} f'$. Let e be such a hyperedge. Since J' is consistent, it cannot be the case that J' contains all the facts in e. Choose a fact $f \in e$ such that $f \notin J'$. Then $f \succ^{+} f'$, and since $\succ$ is transitive (and f and f' are neighbors), we have $f \succ f'$. So $f \in J \setminus J'$ and $f \succ f'$, as required. ◀

Interestingly, the proof of Theorem 7.6 is based on the correctness of the algorithm CCategoricity of Figure 3b. Combining Theorems 6.1 and 7.6, we get the following.

▶ **Corollary 7.7.** *For transitive priority relations, the problems g-categoricity and c-categoricity coincide, and in particular, g-categoricity is solvable in polynomial time.*

We conclude with two comments. First, the reader may wonder whether Theorem 7.6 and Corollary 7.7 hold for p-categoricity as well. This is not the case. Hardness of p-categoricity$\langle \mathbf{S}^6 \rangle$ is proved by a reduction to a transitive priority relation. Second, in their analysis Fagin et al. [12] have constructed various reductions for proving coNP-hardness of g-repair checking. In several of these, the priority relation is transitive. We conclude that there are schemas $\mathbf{S}$ such that, on transitive priority relations, g-repair checking is coNP-complete whereas g-categoricity is solvable in polynomial time.

8 Concluding Remarks

We investigated the complexity of the categoricity problem, which is that of determining whether the provided priority relation suffices to repair the database unambiguously, in the framework of preferred repairs [32]. In this framework, integrity constraints are anti-monotonic and repairing operations are tuple deletions (i.e., *subset repairs*). Following the three semantics of optimal repairs, we investigated the three variants of this problem: p-categoricity, g-categoricity and c-categoricity. We established a dichotomy in the data complexity of p-categoricity for the case where constraints are FDs, partitioning the cases into polynomial time and coNP-completeness. We further showed that the tractable side of p-categoricity extends to g-categoricity, but the latter can reach Π_2^p-completeness already for two FDs. Finally, we showed that c-categoricity is solvable in polynomial time in the general case where integrity constraints are given as a conflict hypergraph.

We did not address here any qualitative discrimination among the three notions of x-repairs. Rather, we continue the line of work [13, 33] that explores the impact of the choice on the entailed computational complexity. It has been established that, as far as repair checking is concerned, the Pareto and the completion semantics behave much better than the global one, since g-repair checking is tractable only for a very restricted class of schemas [13]. In this work we have shown that from the viewpoint of categoricity, the Pareto semantics becomes likewise intractable (while the global semantics hits an even higher complexity class), and the completion semantics outstands so far as the most efficient option to adopt.

We complete this paper by discussing directions for future research. It would be interesting to further understand the complexity of g-categoricity, towards a dichotomy (at least for FDs). We have left open the question of whether there exists a schema with a single relation and a set of FDs, *not* equivalent to a single FD, such that g-categoricity is solvable in polynomial time. Another interesting direction is the generalization of categoricity to the problems of *counting* and *enumerating* the preferred repairs. For classical repairs (without a priority relation), Maslowski and Wijsen [28, 29] established dichotomies (FP vs. #P-completeness) in the complexity of counting in the case where constraints are primary keys. For the general case of denial constraints, counting the classical repairs reduces to the enumeration of independent sets of a hypergraph with a bounded edge size, a problem shown by Boros et al. [7] to be solvable in incremental polynomial time (and in particular polynomial input-output complexity). For a general given conflict hypergraph, repair enumeration is the well known problem of enumerating the *minimal hypergraph transversals*; whether this problem is solvable in polynomial total time is a long standing open problem [20].

A natural continuation of this work would be to chart the complexity boundaries for more general cleaning frameworks that feature preferences between repairs, including different types of integrity constraints, different cleaning operations (e.g., tuple addition and cell update [34]), and different priority specifications among repairs. The latter includes preferences by means of general scoring functions [22, 30], aggregation of scores on the individual cleaning operations [6, 11, 18, 18, 24], priorities among resolution policies [27] and preferences based on soft rules [21, 31].

Acknowledgments

The authors are very grateful to Ronald Fagin and Phokion Kolaitis for insightful discussions on the categoricity problem. Benny Kimelfeld is a Taub Fellow, supported by the Taub Foundation. Kimelfeld's research is also supported by the Israeli Science Foundation, Grants #1295/15 and #1308/15, and the US-Israel Binational Science Foundation, Grant #2014391.

References

1 Foto N. Afrati and Phokion G. Kolaitis. Repair checking in inconsistent databases: algorithms and complexity. In *ICDT*, pages 31–41. ACM, 2009.

2 Douglas E. Appelt and Boyan Onyshkevych. The common pattern specification language. In *TIPSTER Text Program: Phase III*, pages 23–30. Association for Computational Linguistics, 1998.

3 Marcelo Arenas, Leopoldo E. Bertossi, and Jan Chomicki. Consistent query answers in inconsistent databases. In *PODS*, pages 68–79. ACM, 1999.

4 Leopoldo E. Bertossi. *Database Repairing and Consistent Query Answering*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011.

5 Philip Bohannon, Wenfei Fan, Floris Geerts, Xibei Jia, and Anastasios Kementsietsidis. Conditional functional dependencies for data cleaning. In *ICDE*, pages 746–755. IEEE, 2007.

6 Philip Bohannon, Michael Flaster, Wenfei Fan, and Rajeev Rastogi. A cost-based model and effective heuristic for repairing constraints by value modification. In *SIGMOD*, pages 143–154. ACM, 2005.

7 Endre Boros, Khaled M. Elbassioni, Vladimir Gurvich, and Leonid Khachiyan. An efficient incremental algorithm for generating all maximal independent sets in hypergraphs of bounded dimension. *Parallel Processing Letters*, 10(4):253–266, 2000.

8 Yang Cao, Wenfei Fan, and Wenyuan Yu. Determining the relative accuracy of attributes. In *SIGMOD*, pages 565–576. ACM, 2013.

9 Laura Chiticariu, Rajasekar Krishnamurthy, Yunyao Li, Sriram Raghavan, Frederick Reiss, and Shivakumar Vaithyanathan. SystemT: An algebraic approach to declarative information extraction. In *ACL*, pages 128–137, 2010.

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

11 Michele Dallachiesa, Amr Ebaid, Ahmed Eldawy, Ahmed K. Elmagarmid, Ihab F. Ilyas, Mourad Ouzzani, and Nan Tang. NADEEF: a commodity data cleaning system. In *SIGMOD*, pages 541–552. ACM, 2013.

12 Ronald Fagin, Benny Kimelfeld, and Phokion G. Kolaitis. Dichotomies in the complexity of preferred repairs. In *PODS*, pages 3–15. ACM, 2015.

13 Ronald Fagin, Benny Kimelfeld, Frederick Reiss, and Stijn Vansummeren. Cleaning inconsistencies in information extraction via prioritized repairs. In *PODS*, pages 164–175. ACM, 2014.

14 Ronald Fagin, Benny Kimelfeld, Frederick Reiss, and Stijn Vansummeren. Document spanners: A formal approach to information extraction. *J. ACM*, 62(2):12, 2015.

15 Wenfei Fan and Floris Geerts. *Foundations of Data Quality Management*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2012.

16 Wenfei Fan, Floris Geerts, and Jef Wijsen. Determining the currency of data. *ACM Trans. Database Syst.*, 37(4):25, 2012.

17 Wenfei Fan, Jianzhong Li, Shuai Ma, Nan Tang, and Wenyuan Yu. Towards certain fixes with editing rules and master data. *VLDB J.*, 21(2):213–238, 2012.

18 Wenfei Fan, Shuai Ma, Nan Tang, and Wenyuan Yu. Interaction between record matching and data repairing. *J. Data and Information Quality*, 4(4):16:1–16:38, 2014.

19 Terry Gaasterland, Parke Godfrey, and Jack Minker. An overview of cooperative answering. *J. Intell. Inf. Syst.*, 1(2):123–157, 1992.

20 Georg Gottlob and Enrico Malizia. Achieving new upper bounds for the hypergraph duality problem through logic. In *CSL-LICS*, pages 43:1–43:10. ACM, 2014.

21 Sergio Greco, Cristina Sirangelo, Irina Trubitsyna, and Ester Zumpano. Feasibility conditions and preference criteria in querying and repairing inconsistent databases. In *DEXA*, volume 3180 of *LNCS*, pages 44–55. Springer, 2004.

22 Sergio Greco, Cristina Sirangelo, Irina Trubitsyna, and Ester Zumpano. Preferred repairs for inconsistent databases. In *Encyclopedia of Database Technologies and Applications*, pages 480–485. Idea Group, 2005.

23 Benny Kimelfeld, Jan Vondrák, and Ryan Williams. Maximizing conjunctive views in deletion propagation. *ACM Trans. Database Syst.*, 37(4):24, 2012.

24 Solmaz Kolahi and Laks V. S. Lakshmanan. On approximating optimum repairs for functional dependency violations. In *ICDT*, volume 361 of *ACM International Conference Proceeding Series*, pages 53–62. ACM, 2009.

25 Paraschos Koutris and Dan Suciu. A dichotomy on the complexity of consistent query answering for atoms with simple keys. In *ICDT*, pages 165–176. OpenProceedings.org, 2014.

26 Paraschos Koutris and Jef Wijsen. The data complexity of consistent query answering for self-join-free conjunctive queries under primary key constraints. In *PODS*, pages 17–29. ACM, 2015.

27 Maria Vanina Martinez, Francesco Parisi, Andrea Pugliese, Gerardo I. Simari, and V. S. Subrahmanian. Inconsistency management policies. In *KR*, pages 367–377. AAAI Press, 2008.

28 Dany Maslowski and Jef Wijsen. A dichotomy in the complexity of counting database repairs. *J. Comput. Syst. Sci.*, 79(6):958–983, 2013.

29 Dany Maslowski and Jef Wijsen. Counting database repairs that satisfy conjunctive queries with self-joins. In *ICDT*, pages 155–164. OpenProceedings.org, 2014.

30 Amihai Motro, Philipp Anokhin, and Aybar C. Acar. Utility-based resolution of data inconsistencies. In *IQIS*, pages 35–43. ACM, 2004.

31 Davy Van Nieuwenborgh and Dirk Vermeir. Preferred answer sets for ordered logic programs. *TPLP*, 6(1-2):107–167, 2006.

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

33 Slawomir Staworko, Jan Chomicki, and Jerzy Marcinkowski. Preference-driven querying of inconsistent relational databases. In *EDBT Workshops*, volume 4254 of *LNCS*, pages 318–335. Springer, 2006.

34 Jef Wijsen. Database repairing using updates. *ACM Trans. Database Syst.*, 30(3):722–768, 2005.

35 Jef Wijsen. Charting the tractability frontier of certain conjunctive query answering. In *PODS*, pages 189–200. ACM, 2013.