Principles of Progress Indicators for Database Repairing (Extended Version)

Authors anonymized for double-blind reviewing

Abstract

How should a cleaning system measure the amount of inconsistency in the database? Proper measures are important for quantifying the progress made in the cleaning process relative to the remaining effort and resources required. Similarly to effective progress bars in interactive systems, inconsistency should ideally diminish steadily and continuously while aiming to avoid jitters and jumps. Moreover, measures should be computationally tractable towards online applicability. Building on past research on inconsistency measures for knowledge bases, we embark on a systematic investigation of the rationality postulates of inconsistency measures in the context of database repairing systems. Such postulates should take into consideration the interplay between the database tuples, the integrity constraints, and the space of atomic repairing operations. We shed light on classic measures by examining them against a set of rationality postulates, and propose a measure that is both rationale and tractable for the general class of denial constraints.

1 Introduction

Inconsistency of databases may arise in a variety of situations, for a variety of reasons. Database records may be collected from imprecise sources (social encyclopedias/networks, sensors attached to appliances, cameras, etc.) via imprecise procedures (natural-language processing, signal processing, image analysis, etc.), or be integrated from different sources with conflicting information or formats. Common principled approaches to handling inconsistency consider databases that violate *integrity constraints*, but can nevertheless be *repaired* by means of *operations* that revise the database and resolve inconsistency [Arenas *et al.*, 1999]. Instantiations of these approaches differ in the supported types of integrity constraints and operations. The constraints may be Functional Dependencies (FDs) or the more general Equality-Generating Dependencies (EGDs) or, even more generally, Denial Constraints (DCs) and they may be referential (foreign-key) constraints or the more general inclusion dependencies [Afrati and Kolaitis, 2009]. A repair operation can be a *deletion* of a tuple, an *insertion* of a tuple, or an *update* of an attribute value. Operations may be associated with different *costs*, representing levels of trust in data items or extent of impact [Lopatenko and Bertossi, 2007].

Various approaches and systems have been proposed for data cleaning and, in particular, data repairing (e.g., [Ebaid *et al.*, 2013; Geerts *et al.*, 2013; Rekatsinas *et al.*, 2017] to name a few). We explore the question of *how to measure database inconsistency* in a manner that reflects the progress of repairing. Such a measure is useful not just for implementing *progress bars*, but also for recommending next steps in interactive systems, and estimating the potential usefulness and cost of incorporating databases for downstream analytics [Kruse *et al.*, 2015]. Example measures include the number of violations in the database, the number of tuples involved in violations, and the number of operations needed to reach consistency. However, for a measure to effectively communicate *progress indication* in repairing, it should feature certain characteristics. For example, it should minimize jitters, jumps and sudden changes to have a good correlation with "the expected waiting time"—an important aspect in interacting with the users. It should also be computationally tractable so as not to compromise the efficiency of the repair and to allow for interactive systems. Luo et al. [2004] enunciate the importance of these properties, referring to them as "acceptable pacing" and "minimal overhead," respectively, in progress indicators for database queries.

As a guiding principle, we adopt the approach of *rationality postulates* of *inconsistency measures* over knowledge bases that have been investigated in depth by the Knowledge Representation (KR) and Logic communities [Konieczny *et al.*, 2003; Knight, 2003; Grant and Hunter, 2006; Hunter and Konieczny, 2008; Hunter and Konieczny, 2010; Grant and Hunter, 2017; Thimm, 2017]. Yet, the studied measures and postulates fall short of capturing our desiderata, for various reasons. First, inconsistency is typically measured over a knowledge base of logical sentences (formulas without free variables). In databases, we reason about *tuples* (facts) and fixed integrity constraints, and inconsistency typically refers to the tuples rather than the constraints (which are treated as exogenous information). In particular, while a collection of sentences might form a contradiction, a set of tuples can be inconsistent only in the presence of integrity constraints. Hence, as recently acknowledged [Bertossi, 2018], it is of

importance to seek inconsistency measures that are closer to database applications. Perhaps more fundamentally, in order to capture the repairing *process* and corresponding changes to the database, the measure should be aware of the underlying *repairing operations* (e.g., tuple insertion, deletion, revision).

For illustration, let us consider the case where all constraints are anti-monotonic (i.e., consistency cannot be violated by deleting tuples), and we allow only tuple deletions as repairing operations. One simple measure is the *drastic measure* $\mathcal{I}_{\mathsf{d}}$, which is 1 if the database is inconsistent, and 0 otherwise [Thimm, 2017]. Of course, this measure hardly communicates progress, as it does not change until the very end. What about the measure $\mathcal{I}_{\mathsf{P}}$ that counts the number of *problematic tuples*, which are the tuples that participate in (minimal) witnesses of inconsistency [Hunter and Konieczny, 2008; Hunter and Konieczny, 2010]? This measure suffers from a disproportional reaction to repairing operations, since the removal of a single tuple (e.g., a misspelling of a country name) can cause a drastic reduction in inconsistency. As another example, take the measure $\mathcal{I}_{\mathsf{MC}}$ that counts the number of maximal consistent subsets. This measure suffers from various problems: adding constraints can cause a *reduction* in inconsistency, it may fail to reflect change for *any* deletion of a tuple and, again, it may react disproportionally to a tuple deletion. Moreover, it is is hard to compute (#P-complete) already for simple FDs [Livshits and Kimelfeld, 2017].

In a recent attention in the community of database theory, measures have been proposed based on the concept of a *minimal repair*—the minimal number of deletions needed to obtain consistency [Bertossi, 2018]. We refer to this measure as $\mathcal{I}_{\mathcal{R}}$. Our exploration shows that $\mathcal{I}_{\mathcal{R}}$ indeed satisfies the rationality criteria that we define later on, and so, we provide a formal justification to its semantics. Yet, it is again intractable (NP-hard) even for simple sets of FDs [Livshits *et al.*, 2018]. Interestingly, we are able to show that a *linear relaxation* of this measure, which we refer to as $\mathcal{I}^{\mathsf{lin}}_{\mathcal{R}}$, provides a combination of rationality and tractability.

We make a step towards formalizing the features and shortcomings of inconsistency measures such as the aforementioned ones. We consider four rationality postulates for database inconsistency in the context of a *repairing system* (i.e., a space of weighted repairing operations): *positivity*—the measure is strictly positive if and only if the database is inconsistent; *monotonicity*—inconsistency cannot be reduced by adding constraints; *progression*—we can always find an operation that reduces inconsistency; and *continuity*—a single operation can have a limited relative impact on inconsistency. We examine a collection of measures against these postulates, and show that $\mathcal{I}_{\mathcal{R}}$ stands out. Nevertheless, this measure is intractable. In particular, we show that computing $\mathcal{I}_{\mathcal{R}}$ is hard already for the case of a single EGD. Finally, we prove that for tuple deletions, $\mathcal{I}^{\mathsf{lin}}_{\mathcal{R}}$ satisfies all four postulates and is computable in polynomial time, even for the general case of arbitrary sets of denial constraints.

Our work is complementary to that of Grant and Hunter [2011] who also studied repairing (or *resolution*) operators, but they focus on a different aspect of repairing—the trade-off between inconsistency reduction and *information loss*. An operation is beneficial if it causes a high reduction in inconsistency alongside a low loss of information. Instead, our focus here is on measuring *progress* of repairing, and it is an interesting future direction to understand how the two relate to each other and/or can be combined. Another complementary problem is that of associating *individual facts* with portions of the database inconsistency (e.g., the Shapley value of the fact) and using these portions to define preferences among repairs, as studied by Yun et al. [2018].

In the remainder of the paper, we present preliminary concepts and terminology (Section 2), discuss inconsistency measures (Section 3), their rationality postulates (Section 4) and complexity aspects (Section 5), and make concluding remarks (Section 6).

2 Preliminaries

We first give the basic terminology and concepts.

Relational Model A *relation signature* is a sequence $\alpha = (A_1, \ldots, A_k)$ of distinct *attributes* A_i, where k is the *arity* of α. A *relational schema* (or just *schema* for short) $\mathbf{S}$ has a finite set of *relation symbols* R, each associated with a signature that we denote by $sig(R)$. If $sig(R)$ has arity k, then we say that R is k-ary. A *fact* f over $\mathbf{S}$ is an expression of the form $R(c_1, \ldots, c_k)$, where R is a k-ary relation symbol of $\mathbf{S}$, and $c_1, \ldots, c_k$ are *values*. When $\mathbf{S}$ is irrelevant or clear from the context, we may call a fact over $\mathbf{S}$ simply a *fact*. If $f = R(c_1, \ldots, c_k)$ is a fact and $sig(R) = (A_1, \ldots, A_k)$, then we refer to the value c_i as $f.A_i$.

A *database* D over $\mathbf{S}$ is a mapping from a finite set $ids(D)$ of *record identifiers* to facts over $\mathbf{S}$. The set of all databases over a schema $\mathbf{S}$ is denoted by $\mathrm{DB}(\mathbf{S})$. We denote by $D[i]$ the fact that D maps to the identifier i. A database D is a *subset* of a database D', denoted $D \subseteq D'$, if $ids(D) \subseteq ids(D')$ and $D[i] = D'[i]$ for all $i \in ids(D)$.

An *integrity constraint* is a first-order sentence over $\mathbf{S}$. A database D satisfies a set Σ of integrity constraints, denoted $D \models \Sigma$, if D satisfies every constraint $\sigma \in \Sigma$. If Σ and Σ' are sets of integrity constraints, then we write $\Sigma \models \Sigma'$ to denote that Σ *entails* Σ'; that is, every database that satisfies Σ also satisfies Σ'. We also write $\Sigma \equiv \Sigma'$ to denote that Σ and Σ' are *equivalent*, that is, $\Sigma \models \Sigma'$ and $\Sigma' \models \Sigma$. By a *constraint system* we refer to a class $\mathbf{C}$ of integrity constraints (e.g., the class of all functional dependencies).

As a special case, a *Functional Dependency* (FD) $R : X \rightarrow Y$, where R is a relation symbol and $X, Y \subseteq sig(R)$, states that every two facts that agree on (i.e., have equal values in) every attribute in X should also agree on every attribute in Y. The more general *Equality Generating Dependency* (EGD) has the form $\forall \vec{x}\,[\varphi_1(\vec{x}) \wedge \cdots \wedge \varphi_k(\vec{x}) \rightarrow y_1 = y_2]$, where each $\varphi_j(\vec{x})$ is an atomic formula over the schema and y_1 and y_2 are variables in $\vec{x}$. Finally, a *Denial Constraint* (DC) has the form $\forall \vec{x}\,[\varphi_1(\vec{x}) \wedge \cdots \wedge \varphi_k(\vec{x}) \wedge \psi(\vec{x})]$, where each $\varphi_j(\vec{x})$ is an atomic formula and $\psi(\vec{x})$ is a conjunction of atomic comparisons over $\vec{x}$.

Repair Systems Let $\mathbf{S}$ be a schema. A *repairing operation* (or just *operation*) o transforms a database D over $\mathbf{S}$ into another database $o(D)$ over $\mathbf{S}$, that is, $o : \mathrm{DB}(\mathbf{S}) \rightarrow \mathrm{DB}(\mathbf{S})$. An example is *tuple deletion*, denoted $\langle -i \rangle(\cdot)$, parameterized by a tuple identifier i and applicable to a database D if

$i \in ids(D)$; the result $\langle -i \rangle(D)$ is obtained from D by deleting the tuple identifier i (along with the corresponding fact $D[i]$). Another example is *tuple insertion*, denoted $\langle +f \rangle(\cdot)$, parameterized by a fact f; the result $\langle +f \rangle(D)$ is obtained from D by adding f with a new tuple identifier. (For convenience, this is the minimal integer i such that $i \notin ids(D)$.) A third example is *attribute update*, denoted $\langle i.A \leftarrow c \rangle(\cdot)$, parameterized by a tuple identifier i, an attribute A, and a value c, and applicable to D if $i \in ids(D)$ and A is an attribute of the fact $D[i]$; the result $\langle i.A \leftarrow c \rangle(D)$ is obtained from D by setting the value $D[i].A$ to c. By convention, if o is not applicable to D, then it keeps D intact, that is, $o(D) = D$.

A *repair system* (over a schema $\mathbf{S}$) is a collection of repairing operations with an associated *cost* of applying to a given database. For example, a smaller change of value might be less costly than a greater one [Gardezi *et al.*, 2011], and some facts might be more costly than others to delete [Lopatenko and Bertossi, 2007; Livshits *et al.*, 2018] or update [Kolahi and Lakshmanan, 2009; Livshits *et al.*, 2018; Bertossi *et al.*, 2008]; changing a person's zip code might be less costly than changing the person's country, which, in turn, might be less costly than deleting the entire person's record. Formally, a repair system $\mathcal{R}$ is a pair (O, κ) where O is a set of operations and $\kappa : O \times \mathrm{DB}(\mathbf{S}) \to [0, \infty)$ is a cost function that assigns the cost $\kappa(o, D)$ to applying o to D. We require that $\kappa(o, D) = 0$ if and only if $D = o(D)$; that is, the cost is nonzero when, and only when, an actual change occurs.

For a repair system $\mathcal{R}$, we denote by $\mathcal{R}^\star = (O^\star, \kappa^\star)$ the repair system of all *sequences* of operations from $\mathcal{R}$, where the cost of a sequence is the sum of costs of the individual operations thereof. Formally, for $\mathcal{R} = (O, \kappa)$, the repair system $\mathcal{R}^\star$ is $(O^\star, \kappa^\star)$, where $O^\star$ consists of all compositions $o = o_m \circ \cdots \circ o_1$, with $o_j \in O$ for all $j = 1, \ldots, m$, defined inductively by $o_m \circ \cdots \circ o_1(D) = o_m(o_{m-1} \circ \cdots \circ o_1(D))$ and $\kappa^\star(o_m \circ \cdots \circ o_1, D) = \kappa(o_m, o_{m-1} \circ \cdots \circ o_1(D)) + \kappa^\star(o_{m-1} \circ \cdots \circ o_1, D)$. Let $\mathbf{C}$ be a constraint system and $\mathcal{R}$ a repair system. We say that $\mathbf{C}$ is *realizable by* $\mathcal{R}$ if it is always possible to make a database satisfy constraints of $\mathbf{C}$ by repeatedly applying operations from $\mathcal{R}$. Formally, $\mathbf{C}$ is realizable by $\mathcal{R}$ if for every database D and a finite set $\Sigma \subseteq \mathbf{C}$ there is a sequence o in $\mathcal{R}^\star$ such that $o(D) \models \Sigma$. An example of $\mathbf{C}$ is the system $\mathbf{C}_{\mathsf{FD}}$ of all FDs $R : X \to Y$. An example of $\mathcal{R}$ is the *subset* system, denoted $\mathcal{R}_\subseteq$, where O is the set of all tuple deletions (hence, the result is always a subset of the original database), and κ is determined by a special *cost* attribute, $\kappa(\langle -i \rangle(D)) = D[i].\mathtt{cost}$, if a cost attribute exists, and otherwise, $\kappa(\langle -i \rangle(D)) = 1$ (every tuple has unit cost for deletion). Observe that $\mathcal{R}_\subseteq$ realizes $\mathbf{C}$, since the latter consists of anti-monotonic constraints.

3 Inconsistency Measures

Let $\mathbf{S}$ be a schema, and let $\mathbf{C}$ be a constraint system. An *inconsistency measure* is a function $\mathcal{I}$ that maps a finite set $\Sigma \subseteq \mathbf{C}$ of integrity constraints and a database D to a number $\mathcal{I}(\Sigma, D) \in [0, \infty)$. Intuitively, a higher $\mathcal{I}(\Sigma, D)$ implies that D is farther from satisfying Σ. We make two standard requirements:

- $\mathcal{I}$ is zero on consistent databases; that is, $\mathcal{I}(\Sigma, D) = 0$ whenever $D \models \Sigma$;
- $\mathcal{I}$ is invariant under logical equivalence of constraints; that is, $\mathcal{I}(\Sigma, D) = \mathcal{I}(\Sigma', D)$ whenever $\Sigma \equiv \Sigma'$.

Next, we discuss several examples of inconsistency measures. Some of these measures (namely, $\mathcal{I}_\mathsf{d}$, $\mathcal{I}_{\mathsf{MI}}$, $\mathcal{I}_\mathsf{P}$ and $\mathcal{I}_{\mathsf{MC}}$) are adjusted from the survey of Thimm [2017]. The simplest measure is the *drastic inconsistency value*, denoted $\mathcal{I}_\mathsf{d}$, which is the indicator function of inconsistency.

$$\mathcal{I}_\mathsf{d}(\Sigma, D) := \begin{cases} 0 & \text{if } D \models \Sigma; \\ 1 & \text{otherwise.} \end{cases}$$

The next measure assumes an underlying repair system $\mathcal{R}$ and an underlying constraint system $\mathbf{C}$ such that $\mathbf{C}$ is realizable by $\mathcal{R}$. The measure $\mathcal{I}_\mathcal{R}$ is the minimal cost of a sequence of operations that repairs the database. It captures the intuition around various notions of repairs known as *cardinality* repairs and *optimal* repairs [Kolahi and Lakshmanan, 2009; Livshits *et al.*, 2018; Afrati and Kolaitis, 2009].

$$\mathcal{I}_\mathcal{R}(\Sigma, D) := \min\{\kappa^\star(o, D) \mid o \in O^\star \text{ and } o(D) \models \Sigma\}$$

Note that $\mathcal{I}_\mathcal{R}$ is the distance from satisfaction used in *property testing* [Goldreich *et al.*, 1998] in the special case where the repair system consists of unit-cost insertions and deletions.

Measures for anti-monotonic constraints. The next measures apply to systems $\mathbf{C}$ of *anti-monotonic* constraints. Recall that an integrity constraint σ is anti-monotonic if for all databases D and D', if $D \subseteq D'$ and $D' \models \sigma$, then $D \models \sigma$. Examples of anti-monotonic constraints are the *Denial Constraints* (DCs) [Gaasterland *et al.*, 1992], the classic *Functional Dependencies* (FDs), *conditional FDs* [Bohannon *et al.*, 2007], and *Equality Generating Dependencies* (EDGs) [Beeri and Vardi, 1981].

For a set $\Sigma \subseteq \mathbf{C}$ of constraints and a database D, denote by $\mathsf{MI}_\Sigma(D)$ the set of all *minimal inconsistent* subsets of D; that is, the set of all $E \subseteq D$ such that $E \not\models \Sigma$ and, moreover, $E' \models \Sigma$ for all $E' \subsetneq E$. Again using our assumption that constraints are anti-monotonic, it holds that $D \models \Sigma$ if and only if $\mathsf{MI}_\Sigma(D)$ is empty. Drawing from known inconsistency measures [Hunter and Konieczny, 2008; Hunter and Konieczny, 2010], the measure $\mathcal{I}_{\mathsf{MI}}$, also known as *MI Shapley Inconsistency*, is the cardinality of this set.

$$\mathcal{I}_{\mathsf{MI}}(\Sigma, D) := |\mathsf{MI}_\Sigma(D)|$$

A fact f that belongs to a minimal inconsistent subset K (that is, $f \in K \in \mathsf{MI}_\Sigma(D)$) is called *problematic*, and the measure $\mathcal{I}_\mathsf{P}$ counts the problematic facts [Grant and Hunter, 2011].

$$\mathcal{I}_\mathsf{P}(\Sigma, D) := |\cup\, \mathsf{MI}_\Sigma(D)|$$

For a finite set $\Sigma \subseteq \mathbf{C}$ of constraints and a database D, we denote by $\mathsf{MC}_\Sigma(D)$ the set of all *maximal consistent* subsets of D; that is, the set of all $E \subseteq D$ such that $E \models \Sigma$ and, moreover, $E' \not\models \Sigma$ whenever $E \subsetneq E' \subseteq D$. Observe that if $D \models \Sigma$, then $\mathsf{MC}_\Sigma(D)$ is simply the singleton $\{D\}$. Moreover, under the assumption that constraints are anti-monotonic, the set $\mathsf{MC}_\Sigma(D)$ is never empty (since, e.g., the empty set is consistent). The measure $\mathcal{I}_{\mathsf{MC}}$ is the cardinality of $\mathsf{MC}_\Sigma(D)$, minus one.

$$\mathcal{I}_{\mathsf{MC}}(\Sigma, D) := |\mathsf{MC}_\Sigma(D)| - 1$$

4 Rationality for Progress Indication

We now propose and discuss several properties (*postulates*) of general inconsistency measures that capture the rationale for usability for progress estimation in database repairing. We illustrate the satisfaction or violation of these postulates over the different measures that we presented in the previous section. The behavior of these measures with respect to the postulates is summarized in Table 1, which we discuss later on. The measures are all defined in Section 3, except for $\mathcal{I}^{\mathsf{lin}}_{\mathcal{R}}$ that we define later in Section 5.

A basic postulate is *positivity*, sometimes referred to as *consistency* [Grant and Hunter, 2017].

Positivity: $\mathcal{I}(\Sigma, D) > 0$ whenever $D \not\models \Sigma$.

For illustration, each of $\mathcal{I}_{\mathsf{d}}$, $\mathcal{I}_{\mathsf{MI}}$, $\mathcal{I}_{\mathsf{P}}$, and $\mathcal{I}_{\mathcal{R}}$ satisfies positivity, but not $\mathcal{I}_{\mathsf{MC}}$. For example, let D consist of two facts, $R(a)$ and $R(b)$, and Σ consist of the constraint $\neg R(a)$; that is, $R(a)$ is not in the database. Then $\mathcal{I}_{\mathsf{MC}}(\Sigma, D) = 0$ since $\mathsf{MC}_\Sigma(D) = \{R(b)\}$. Yet, in the case of FDs (i.e., $\mathbf{C} = \mathbf{C}_{\mathsf{FD}}$), every violation involves two facts, and so $|\mathsf{MC}_\Sigma(D)| \geq 2$ and positivity is satisfied.

The next postulate is *monotonicity*—inconsistency cannot decrease if the constraints get stricter.

Monotonicity: $\mathcal{I}(\Sigma, D) \leq I(\Sigma', D)$ whenever $\Sigma' \models \Sigma$.

For example, $\mathcal{I}_{\mathsf{d}}$ and $\mathcal{I}_{\mathcal{R}}$ satisfy monotonicity, since every repair w.r.t. Σ' is also a repair w.r.t. Σ. The measures $\mathcal{I}_{\mathsf{MI}}$ and $\mathcal{I}_{\mathsf{P}}$ also satisfy monotonicity in the special case of FDs, since in this case $|\mathsf{MI}_\Sigma(D)|$ is the number of fact pairs that jointly violate an FD, which can only increase when adding or strengthening FDs. Yet, they may violate monotonicity when going beyond FDs to the more general class of DCs.

Proposition 1. *In the case of $\mathcal{I}_{\mathsf{MI}}$ and $\mathcal{I}_{\mathsf{P}}$, monotonicity can be violated already for the class of DCs.*

Proof. We begin with $\mathcal{I}_{\mathsf{MI}}$. Consider a schema with a single relation symbol, and for a natural number $k > 0$, let Σ_k consist of a single DC stating that *there are at most $k-1$ facts in the database*. (The reader can easily verify that, indeed, Σ_k can be expressed as a DC.) Then, $\mathcal{I}_{\mathsf{MI}}(\Sigma_k, D) = \binom{n}{k}$ whenever D has $n \geq k$ facts. In particular, whenever $k' > k$ and D has $n \geq 2k'$ facts, it holds that $\mathcal{I}_{\mathsf{MI}}(\Sigma_{k'}, D) > \mathcal{I}_{\mathsf{MI}}(\Sigma_k, D)$ while $\Sigma_k \models \Sigma_{k'}$.

We now consider $\mathcal{I}_{\mathsf{P}}$. Let $\mathbf{S}$ be a schema that contains two relation symbols $R(A, B)$ and $S(A, B)$. Consider the following two EGDs (which are, of course, special cases of DCs):

$$\sigma_1 = \forall x, y, z, w[\big(R(x, y), S(x, z), S(x, w)\big) \Rightarrow z = w]$$
$$\sigma_2 = \forall x, z, w[\big(S(x, z), S(x, w)\big) \Rightarrow z = w]$$

Let $\Sigma_1 = \{\sigma_1\}$ and $\Sigma_2 = \{\sigma_1, \sigma_2\}$. Every set in $\mathsf{MI}_{\Sigma_1}(D)$ is of size three, while the size of the sets in $\mathsf{MI}_{\Sigma_2}(D)$ is two. Hence, in a database where $|\mathsf{MI}_{\Sigma_1}(D)| = |\mathsf{MI}_{\Sigma_2}(D)|$ (i.e., a database where σ_1 is violated by $\{R(\mathtt{a}, \mathtt{b}), S(\mathtt{a}, \mathtt{c}), S(\mathtt{a}, \mathtt{d})\}$ if and only if σ_2 is violated by $\{S(\mathtt{a}, \mathtt{c}), S(\mathtt{a}, \mathtt{d})\}$), it will hold that $|P_{\Sigma_1}(D)| > |P_{\Sigma_2}(D)|$, while $\Sigma_2 \models \Sigma_1$. □

The measure $\mathcal{I}_{\mathsf{MC}}$, on the other hand, can violate monotonicity even for functional dependencies.

Proposition 2. *In the case of $\mathcal{I}_{\mathsf{MC}}$, monotonicity can be violated already for the class of FDs.*

Proof. Let D consist of these facts over $R(A, B, C, D)$:

$$f_1 = R(0, 0, 0, 0) \quad f_2 = R(1, 0, 0, 0)$$
$$f_3 = R(1, 1, 0, 1) \quad f_4 = R(0, 1, 0, 1)$$

Let $\Sigma_1 = \{A \to B\}$ and $\Sigma_2 = \{A \to B, C \to D\}$. Then $\Sigma_2 \models \Sigma_1$ and the following hold.

$$\mathsf{MC}(\Sigma_1, D) = \{\{f_1, f_2\}, \{f_1, f_3\}, \{f_2, f_4\}, \{f_3, f_4\}\}$$
$$\mathsf{MC}(\Sigma_2, D) = \{\{f_1, f_2\}, \{f_3, f_4\}\}$$

Hence, we have $\mathcal{I}_{\mathsf{MC}}(\Sigma_1, D) = 3$ and $\mathcal{I}_{\mathsf{MC}}(\Sigma_2, D) = 1$, proving that monotonicity is violated. □

Positivity and monotonicity serve as sanity conditions that the measure at hand indeed quantifies the inconsistency in the database—it does not ignore inconsistency, and it does not reward strictness of constraints. The next two postulates aim to capture the rationale of using the inconsistency measure as progress indication for data repairing. Such a measure should not dictate the repairing operations to the data cleaner, but rather accommodate the process with a number that suitably communicates progress in inconsistency elimination. As an example, the measure $\mathcal{I}_{\mathsf{d}}$ is useless in this sense, as it provides no indication of progress until the very end; in contrast, a useful progress bar progresses steadily and continuously. To the aim, we propose two postulates that are aware of the underlying repair system $\mathcal{R} = (O, \kappa)$ as a model of operations. They are inspired by what Luo et al. [2004] state informally as "continuously revised estimates" and "acceptable pacing." *Progression* states that inconsistency can always diminish with a single operation, and *continuity* limits the ability of such operation to have a relatively drastic effect.

More formally, *progression* states that, within the underlying repair system $\mathcal{R} = (O, \kappa)$, there is always a way to progress towards consistency, as we can find an operation o of $\mathcal{R}$ such that inconsistency reduces after applying o.

Progression: whenever $D \not\models \Sigma$, there is $o \in O$ such that $\mathcal{I}(\Sigma, o(D)) < I(\Sigma, D)$.

For illustration, let us restrict to the system $\mathcal{R}_\subseteq$ of subset repairs. Clearly, the measure $\mathcal{I}_{\mathsf{d}}$ violates progression. The measure $\mathcal{I}_{\mathcal{R}}$ satisfies progression, since we can always remove a fact from the minimum repair. The measure $\mathcal{I}_{\mathsf{MI}}$ satisfies progression, since we can always remove a fact f that participates in one of the minimal inconsistent subsets and, by doing so, eliminate all the subsets that include f. When we remove a fact f that appears in a minimal inconsistent subset, the measure $\mathcal{I}_{\mathsf{P}}$ decreases as well; hence, it satisfies progression. On the other hand, $\mathcal{I}_{\mathsf{MC}}$ may violate progression even for functional dependencies.

Proposition 3. *In the case of $\mathcal{I}_{\mathsf{MC}}$, progression can be violated already for the class $\mathbf{C}_{\mathsf{FD}}$ of FDs and the system $\mathcal{R}_\subseteq$ of subset repairs.*

Proof. Consider again the database D and the set Σ_2 from the proof of Proposition 2. As explained there, $\mathcal{I}_{\mathsf{MC}}(\Sigma_2, D) = 1$. The reader can easily verify that for every tuple deletion o, it is still the case that $\mathcal{I}_{\mathsf{MC}}(\Sigma_2, o(D)) = 1$. □

The last postulate we discuss is *continuity* that, as said above, limits the relative reduction of inconsistency in reaction to a single operation. More formally, this postulate is parameterized by a number $\delta \geq 1$ and it states that, for every two databases D_1 and D_2, and for each operation o_1 on D_1 we can find an operation o_2 on D_2 that is (almost) at least as impactful as o_1: it either eliminates all inconsistency or reduces inconsistency by at least $1/\delta$ of what o_1 does. More formally, we denote by $\Delta_{\mathcal{I},\Sigma}(o_1, D_1)$ the value $\mathcal{I}(\Sigma, D_1) - \mathcal{I}(\Sigma, o_1(D_1))$.

δ-continuity: For all Σ, D_1, D_2 and $o_1 \in O$, there exists $o_2 \in O$ such that either $o_2(D_2) \models \Sigma$ or $\Delta_{\mathcal{I},\Sigma}(o_2, D_2) \geq \Delta_{\mathcal{I},\Sigma}(o_1, D_1)/\delta$.

Note that the possibility of $o_2(D_2) \models \Sigma$ eliminates the case where a measure violates continuity just because of a situation where the database is only slightly inconsistent and a last step suffices to complete repairing.

This definition can be extended to the case where the inconsistency measure is aware of the cost of operations in the repair system $\mathcal{R}$. There, the change is relative to the cost of the operation. That is, we define the weighted version of δ-continuity in the following way.

Weighted δ-continuity: For all Σ, D_1, D_2 and $o_1 \in O$, there exists $o_2 \in O$ such that either $o_2(D_2) \models \Sigma$ or $\frac{\Delta_{\mathcal{I},\Sigma}(o_2,D_2)}{\kappa(o_2,D_2)} \geq \frac{\Delta_{\mathcal{I},\Sigma}(o_1,D_1)}{\delta \cdot \kappa(o_1,D_1)}$.

We say that a measure $\mathcal{I}$ has *bounded continuity*, if there exists $\delta > 0$ such that $\mathcal{I}$ satisfies δ-continuity. Clearly, none of the measures discussed so far, except for $\mathcal{I}_{\mathcal{R}}$, satisfies (unweighted) bounded continuity.

Proposition 4. *In the case of $\mathcal{I}_{\mathsf{d}}$, $\mathcal{I}_{\mathsf{MI}}$, $\mathcal{I}_{\mathsf{P}}$ and $\mathcal{I}_{\mathsf{MC}}$, bounded (unweighted) continuity can be violated already for the class $\mathbf{C}_{\mathsf{FD}}$ of FDs and the system $\mathcal{R}_{\subseteq}$ of subset repairs.*

Proof. Let $\Sigma = \{A \rightarrow B\}$ and let D be a database that contains the following facts over $R(A, B, C)$:

$$f_0 = R(0, 0, 0) \quad f_i = R(0, 1, i) \quad f_j^k = R(j, k, 0)$$

where $i, j \in \{1, n\}$ for some n and $k \in \{1, 2\}$. The fact f_0 violates the FD with every fact f_i, and for each j, the facts f_j^1 and f_j^2 jointly violate the FD. All the facts in the database participate in a violation of the FD; hence, $\mathcal{I}_{\mathsf{P}}(\Sigma, D) = 3n + 1$. In addition, it holds that $\mathcal{I}_{\mathsf{MI}}(\Sigma, D) = 2n$.

Let the operation o_1 be the deletion of f_0. Applying o_1, we significantly reduce inconsistency w.r.t. these two measures, since none of the facts f_i now participates in a violation; thus, $\mathcal{I}_{\mathsf{P}}(\Sigma, o_1(D)) = 2n$ and $\mathcal{I}_{\mathsf{MI}}(\Sigma, o_1(D)) = n$. However, every possible operation o_2 on the database $o_1(D)$ only slightly reduces inconsistency (by two in the case of $\mathcal{I}_{\mathsf{P}}$ and by one in the case of $\mathcal{I}_{\mathsf{MI}}$). Therefore, $\Delta_{\mathcal{I}_{\mathsf{MI}},\Sigma}(o_1, D) = n$ and $\Delta_{\mathcal{I}_{\mathsf{MI}},\Sigma}(o_2, o_1(D)) = 1$, and the ratio between these two values depends on $|D|$. Similarly, it holds that $\Delta_{\mathcal{I}_{\mathsf{P}},\Sigma}(o_1, D) = n+1$ and $\Delta_{\mathcal{I}_{\mathsf{P}},\Sigma}(o_2, o_1(D)) = 2$, and again the ratio between these two values depends on $|D|$.

As for $\mathcal{I}_{\mathsf{d}}$ and $\mathcal{I}_{\mathsf{MC}}$, we use Proposition 5 that we give later on. In the case of FDs, each of the two measures satisfies positivity but not progression (Proposition 3), and hence, they violate bounded continuity. □

Table 1: Satisfaction of rationality postulates for a system $\mathbf{C}$ of anti-monotonic constraints and the repairing system $\mathcal{R}_{\subseteq}$, and tractability for DCs ("PTime" column, assuming P $\neq$ NP)

	Pos.	Mono.	Prog.	B. Cont.	PTime
$\mathcal{I}_{\mathsf{d}}$	✓	✓	✗	✗	✓
$\mathcal{I}_{\mathsf{MI}}$	✓	✗	✓	✗	✓
$\mathcal{I}_{\mathsf{P}}$	✓	✗	✓	✗	✓
$\mathcal{I}_{\mathsf{MC}}$	✗	✗	✗	✗	✗
$\mathcal{I}_{\mathcal{R}}$	✓	✓	✓	✓	✗
$\mathcal{I}_{\mathcal{R}}^{\mathsf{lin}}$	✓	✓	✓	✓	✓

On the other hand, it is an easy observation that $\mathcal{I}_{\mathcal{R}}$ satisfies bounded continuity, and even bounded *weighted* continuity.

Table 1 summarizes the satisfaction of the postulates held by the different inconsistency measures we discussed here, for the case of a system $\mathbf{C}$ of anti-monotonic constraints and the repair system $\mathcal{R}_{\subseteq}$. The last column refers to computational complexity and the last row refers to another measure, $\mathcal{I}_{\mathcal{R}}^{\mathsf{lin}}$, both discussed in Section 5.

Note that there are some dependencies among the postulates, as shown in the following easy proposition.

Proposition 5. *Suppose that the class $\mathbf{C}$ is realizable by the repair system $\mathcal{R}$, and let $\mathcal{I}$ be an inconsistency measure.*

- *If $\mathcal{I}$ satisfies progression, then $\mathcal{I}$ satisfies positivity.*
- *If $\mathcal{I}$ satisfies positivity and bounded continuity, then $\mathcal{I}$ satisfies progression.*

The proof of Proposition 5 is in the Appendix.

5 Computational Complexity

We now discuss aspects of computational complexity, and begin with the complexity of measuring inconsistency according to the aforementioned example measures. We focus on the class of DCs and the special case of FDs. Moreover, we focus on *data complexity*, which means that the set Σ of constraints is fixed, and only the database D is given as input for the computation of $\mathcal{I}(\Sigma, D)$.

The measure $\mathcal{I}_{\mathsf{d}}$ boils down to testing consistency, which is doable in polynomial time (under data complexity). The measures $\mathcal{I}_{\mathsf{MI}}$ and $\mathcal{I}_{\mathsf{P}}$ can be computed by enumerating all the subsets of D of a bounded size, where this size is determined by Σ. Hence, $\mathcal{I}_{\mathsf{MI}}$ and $\mathcal{I}_{\mathsf{P}}$ can also be computed in polynomial time. Yet, the measures $\mathcal{I}_{\mathsf{MC}}$ and $\mathcal{I}_{\mathcal{R}}$ can be intractable to compute, already in the case of FDs, as we explain next.

When Σ is a set of FDs, $\mathcal{I}_{\mathsf{MC}}(\Sigma, D)$ is the number of maximal independent sets (minus one) of the *conflict graph* wherein the tuples of D are the nodes, and there is an edge between every two tuples that violate an FD. Counting maximal independent sets is generally #P-complete, with several tractable classes of graphs such as the *P_4-free* graphs—the graphs that do not contain any induced subgraph that is a path of length four. Under conventional complexity assumptions, the finite sets Σ of FDs for which $\mathcal{I}_{\mathsf{MC}}(\Sigma, D)$ is computable in polynomial time are *precisely* the sets Σ of FDs that entail a P_4-free conflict graph for every database D [Livshits and Kimelfeld, 2017].

For $\mathbf{C} = \mathbf{C}_{\mathsf{FD}}$ and $\mathcal{R} = \mathcal{R}_{\subseteq}$, the measure $\mathcal{I}_{\mathcal{R}}(\Sigma, D)$ is the size of the minimum *vertex cover* of the conflict graph.

Again, this is a hard (NP-hard) computational problem on general graphs. In a recent work, it has been shown that there is an efficient procedure that takes as input a set Σ of FDs and determines one of two outcomes: *(a)* $\mathcal{I}_{\mathcal{R}}(\Sigma, D)$ can be computed in polynomial time, *or (b)* $\mathcal{I}_{\mathcal{R}}(\Sigma, D)$ is NP-hard to compute (and even approximate beyond some constant) [Livshits *et al.*, 2018]. There, they have also studied the case where the repair system allows only to update cells (and not delete or insert tuples). In both repair systems it is the case that, if Σ consists of a single FD per relation (which is a commonly studied case, e.g., key constraints [Fuxman and Miller, 2007; Koutris and Wijsen, 2017]) then $\mathcal{I}_{\mathcal{R}}(\Sigma, D)$ can be computed in polynomial time. Unfortunately, this is no longer true (under conventional complexity assumptions) if we go beyond FDs to simple EGDs.

Example 1. *Consider the following four EGDs.*

$$\begin{aligned}
\sigma_1: &\quad \forall x, y, z[R(x, y), R(x, z) \Rightarrow (y = z)] \\
\sigma_2: &\quad \forall x, y, z[R(x, y), R(y, z) \Rightarrow (x = z)] \\
\sigma_3: &\quad \forall x, y, z[R(x, y), R(y, z) \Rightarrow (x = y)] \\
\sigma_4: &\quad \forall x, y, z[R(x, y), S(y, z) \Rightarrow (x = z)]
\end{aligned}$$

Observe that σ_1 is an FD whereas σ_2, σ_3 and σ_4 are not. The constraint σ_2 states that there are no paths of length two except for two-node cycles, and σ_3 states that there are no paths of length two except for single-node cycles. Computing $\mathcal{I}_{\mathcal{R}}(\Sigma, D)$ w.r.t. $\Sigma = \{\sigma_1\}$ or $\Sigma = \{\sigma_4\}$ can be done in polynomial time; however, the problem becomes NP-hard for $\Sigma = \{\sigma_2\}$ and $\Sigma = \{\sigma_3\}$. □

The following theorem gives a full classification of the complexity of computing $\mathcal{I}_{\mathcal{R}}(\Sigma, D)$ for Σ that consists of a single EGD with two binary atoms.

Theorem 1. *Let $\mathcal{R} = \mathcal{R}_{\subseteq}$, and let Σ be a set of constraints that contains a single EGD σ with two binary atoms. If σ is of the following form:*

$$\forall x_1, x_2, x_3[R(x_1, x_2), R(x_2, x_3) \Rightarrow (x_i = x_j)]$$

then computing $\mathcal{I}_{\mathcal{R}}(\Sigma, D)$ is NP-hard. In any other case, $\mathcal{I}_{\mathcal{R}}(\Sigma, D)$ can be computed in polynomial time.

The proof of hardness in Theorem 1 is by reduction from the problem of finding a maximum cut in a graph, which is known to be NP-hard. The full proof, as well as efficient algorithms for the tractable cases, are in the Appendix.

Note that the EGDs σ_2 and σ_3 from Example 1 satisfy the condition of Theorem 1; hence, computing $\mathcal{I}_{\mathcal{R}}(\Sigma, D)$ w.r.t. these EGDs is indeed NP-hard. The EGDs σ_1 and σ_4 do not satisfy the condition of the theorem; thus, computing $\mathcal{I}_{\mathcal{R}}(\Sigma, D)$ w.r.t. these EGDs can be done in polynomial time.

5.1 Rational and Tractable Measure

We now propose a new inconsistency measure that applies to the special case where $\mathbf{C}$ is the class $\mathbf{C}_{\mathsf{DC}}$ of DCs (denial constraints) and $\mathcal{R} = \mathcal{R}_{\subseteq}$. Recall that a DC has the form $\forall \vec{x} \neg[\varphi(\vec{x}) \wedge \psi(\vec{x})]$ where $\varphi(\vec{x})$ is a conjunction of atomic formulas over the schema, and $\psi(\vec{x})$ is a conjunction of comparisons over $\vec{x}$. Also recall that DCs generalize common classes of constraints such as FDs, conditional FDs, and EGDs.

$$\begin{aligned}
\text{Minimize}: &\quad \sum_{i \in ids(D)} x_i \cdot \kappa(\langle -i \rangle(\cdot), D) \text{ subj. to:} \\
\forall E \in \mathsf{MI}_{\Sigma}(D): &\quad \sum_{i \in ids(E)} x_i \geq 1 \qquad (1) \\
\forall i \in ids(D): &\quad x_i \in \{0, 1\} \qquad (2)
\end{aligned}$$

Figure 1: ILP for $\mathcal{I}_{\mathcal{R}}(\Sigma, D)$ under $\mathbf{C}_{\mathsf{DC}}$ and $\mathcal{R}_{\subseteq}$

Let D be a database and Σ a finite set of DCs. For $\mathcal{R} = (O, \kappa)$, the measure $\mathcal{I}_{\mathcal{R}}(\Sigma, D)$ is the result of the Integer Linear Program (ILP) of Figure 1 wherein each x_i, for $i \in ids(D)$, determines whether to delete the ith tuple ($x_i = 1$) or not ($x_i = 0$). Denote by $\mathcal{I}^{\mathsf{lin}}_{\mathcal{R}}(\Sigma, D)$ the solution of the *linear relaxation* of this ILP, which is the Linear Program (LP) obtained from the ILP by replacing the last constraint (i.e., Equation (2)) with:

$$\forall i \in ids(D): \quad 0 \leq x_i \leq 1$$

It is easy to see that the relative rankings of the inconsistency measure values of two databases under $\mathcal{I}^{\mathsf{lin}}_{\mathcal{R}}$ and $\mathcal{I}_{\mathcal{R}}$ are consistent with each other if they have sufficient separation under the first one. More formally, for two databases D_1, D_2 we have that $\mathcal{I}^{\mathsf{lin}}_{\mathcal{R}}(\Sigma, D_1) \geq \mu \cdot \mathcal{I}^{\mathsf{lin}}_{\mathcal{R}}(\Sigma, D_2)$ implies that $\mathcal{I}_{\mathcal{R}}(\Sigma, D_1) \geq \mathcal{I}_{\mathcal{R}}(\Sigma, D_2)$, where μ is the integrality gap of the LP relaxation. The maximum number of tuples involved in a violation of a constraint in Σ gives an upper bound on this integrality gap. In particular, for FDs (as well for the EGDs in Example 1), this number is 2; hence, $\mathcal{I}^{\mathsf{lin}}_{\mathcal{R}}(\Sigma, D_1) \geq 2 \cdot \mathcal{I}^{\mathsf{lin}}_{\mathcal{R}}(\Sigma, D_2)$ implies that $\mathcal{I}_{\mathcal{R}}(\Sigma, D_1) \geq \mathcal{I}_{\mathcal{R}}(\Sigma, D_2)$.

The following theorem shows that $\mathcal{I}^{\mathsf{lin}}_{\mathcal{R}}$ satisfies all four postulates and can be efficiently computed for the class $\mathbf{C}_{\mathsf{DC}}$ of denial constraints and the repair system $\mathcal{R}_{\subseteq}$.

Theorem 2. *The following hold for $\mathbf{C} = \mathbf{C}_{\mathsf{DC}}$ and $\mathcal{R} = \mathcal{R}_{\subseteq}$.*

1. *$\mathcal{I}^{\mathsf{lin}}_{\mathcal{R}}$ satisfies positivity, monotonicity, progression and constant weighted continuity.*
2. *$\mathcal{I}^{\mathsf{lin}}_{\mathcal{R}}$ can be computed in polynomial time (in data complexity).*

The proof of Theorem 2 is in the Appendix. It thus appears from Theorem 2 that, for tuple deletions and DCs, $\mathcal{I}^{\mathsf{lin}}_{\mathcal{R}}$ is a desirable inconsistency measure, as it satisfies the postulates we discussed in this paper and avoids the inherent hardness of $\mathcal{I}_{\mathcal{R}}$ (e.g., Theorem 1).

6 Concluding Remarks

We presented a framework for measuring database inconsistency from the viewpoint of progress estimation in database repairing. In particular, we have discussed four rationality postulates, where two (progression and continuity) are defined in the context of the underlying repair system. We have also used the postulates to reason about various instances of inconsistency measures. In particular, the combination of the postulates and the computational complexity shed a positive light on the linear relaxation of minimal repairing. In future work, we plan to explore other rationality postulates as well as *completeness* criteria for sets of postulates to determine sufficiency for progress indication.

References

[Afrati and Kolaitis, 2009] Foto N. Afrati and Phokion G. Kolaitis. Repair checking in inconsistent databases: algorithms and complexity. In *ICDT*, volume 361, pages 31–41. ACM, 2009.

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

[Beeri and Vardi, 1981] Catriel Beeri and Moshe Y. Vardi. The implication problem for data dependencies. In *ICALP*, volume 115, pages 73–85. Springer, 1981.

[Bertossi *et al.*, 2008] Leopoldo E. Bertossi, Loreto Bravo, Enrico Franconi, and Andrei Lopatenko. The complexity and approximation of fixing numerical attributes in databases under integrity constraints. *Inf. Syst.*, 33(4-5):407–434, 2008.

[Bertossi, 2018] Leopoldo E. Bertossi. Measuring and computing database inconsistency via repairs. *CoRR*, abs/1804.08834, 2018.

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

[Ebaid *et al.*, 2013] Amr Ebaid, Ahmed K. Elmagarmid, Ihab F. Ilyas, Mourad Ouzzani, Jorge-Arnulfo Quiané-Ruiz, Nan Tang, and Si Yin. NADEEF: A generalized data cleaning system. *PVLDB*, 6(12):1218–1221, 2013.

[Fuxman and Miller, 2007] Ariel Fuxman and Renée J. Miller. First-order query rewriting for inconsistent databases. *J. Comput. Syst. Sci.*, 73(4):610–635, 2007.

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

[Gardezi *et al.*, 2011] Jaffer Gardezi, Leopoldo E. Bertossi, and Iluju Kiringa. Matching dependencies with arbitrary attribute values: semantics, query answering and integrity constraints. In *LID*, pages 23–30, 2011.

[Geerts *et al.*, 2013] Floris Geerts, Giansalvatore Mecca, Paolo Papotti, and Donatello Santoro. The LLUNATIC data-cleaning framework. *PVLDB*, 6(9):625–636, 2013.

[Goldreich *et al.*, 1998] Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *J. ACM*, 45(4):653–750, 1998.

[Grant and Hunter, 2006] John Grant and Anthony Hunter. Measuring inconsistency in knowledgebases. *J. Intell. Inf. Syst.*, 27(2):159–184, 2006.

[Grant and Hunter, 2011] John Grant and Anthony Hunter. Measuring consistency gain and information loss in stepwise inconsistency resolution. In *ECSQARU*, volume 6717, pages 362–373. Springer, 2011.

[Grant and Hunter, 2017] John Grant and Anthony Hunter. Analysing inconsistent information using distance-based measures. *Int. J. Approx. Reasoning*, 89:3–26, 2017.

[Hunter and Konieczny, 2008] Anthony Hunter and Sébastien Konieczny. Measuring inconsistency through minimal inconsistent sets. In *KR*, pages 358–366. AAAI Press, 2008.

[Hunter and Konieczny, 2010] Anthony Hunter and Sébastien Konieczny. On the measure of conflicts: Shapley inconsistency values. *Artif. Intell.*, 174(14):1007–1026, 2010.

[Khachiyan, 1979] Leonid G Khachiyan. A polynomial algorithm in linear programming. In *Doklady Academii Nauk SSSR*, volume 244, pages 1093–1096, 1979.

[Knight, 2003] Kevin M. Knight. Two information measures for inconsistent sets. *Journal of Logic, Language and Information*, 12(2):227–248, 2003.

[Kolahi and Lakshmanan, 2009] Solmaz Kolahi and Laks V. S. Lakshmanan. On approximating optimum repairs for functional dependency violations. In *ICDT*, volume 361, pages 53–62. ACM, 2009.

[Konieczny *et al.*, 2003] Sébastien Konieczny, Jérôme Lang, and Pierre Marquis. Quantifying information and contradiction in propositional logic through test actions. In *IJCAI*, pages 106–111. Morgan Kaufmann, 2003.

[Koutris and Wijsen, 2017] Paraschos Koutris and Jef Wijsen. Consistent query answering for self-join-free conjunctive queries under primary key constraints. *ACM Trans. Database Syst.*, 42(2):9:1–9:45, 2017.

[Kruse *et al.*, 2015] Sebastian Kruse, Paolo Papotti, and Felix Naumann. Estimating data integration and cleaning effort. In *EDBT*, pages 61–72. OpenProceedings.org, 2015.

[Livshits and Kimelfeld, 2017] Ester Livshits and Benny Kimelfeld. Counting and enumerating (preferred) database repairs. In *PODS*, pages 289–301. ACM, 2017.

[Livshits *et al.*, 2018] Ester Livshits, Benny Kimelfeld, and Sudeepa Roy. Computing optimal repairs for functional dependencies. In *PODS*, pages 225–237. ACM, 2018.

[Lopatenko and Bertossi, 2007] Andrei Lopatenko and Leopoldo E. Bertossi. Complexity of consistent query answering in databases under cardinality-based and incremental repair semantics. In *ICDT*, pages 179–193, 2007.

[Luo *et al.*, 2004] Gang Luo, Jeffrey F. Naughton, Curt J. Ellmann, and Michael Watzke. Toward a progress indicator for database queries. In *SIGMOD*, pages 791–802, 2004.

[Rekatsinas *et al.*, 2017] Theodoros Rekatsinas, Xu Chu, Ihab F. Ilyas, and Christopher Ré. Holoclean: Holistic data repairs with probabilistic inference. *PVLDB*, 10(11):1190–1201, 2017.

[Thimm, 2017] Matthias Thimm. On the compliance of rationality postulates for inconsistency measures: A more or less complete picture. *KI*, 31(1):31–39, 2017.

[Yun *et al.*, 2018] Bruno Yun, Srdjan Vesic, Madalina Croitoru, and Pierre Bisquert. Inconsistency measures for repair semantics in OBDA. In *IJCAI*, pages 1977–1983. ijcai.org, 2018.

A Additional Proofs

A.1 Proof of Proposition 5

Proposition 5. *Suppose that the class* $\mathbf{C}$ *is realizable by the repair system* $\mathcal{R}$*, and let* $\mathcal{I}$ *be an inconsistency measure.*

- *If* $\mathcal{I}$ *satisfies progression, then* $\mathcal{I}$ *satisfies positivity.*
- *If* $\mathcal{I}$ *satisfies positivity and bounded continuity, then* $\mathcal{I}$ *satisfies progression.*

Proof. The first part of the proposition holds since if $\mathcal{I}$ satisfies progression, then for every inconsistent database D there exists an operation o such that $\mathcal{I}(\Sigma, o(D))$ is strictly lower than $\mathcal{I}(\Sigma, D)$, which implies that $\mathcal{I}(\Sigma, D) > 0$.

For the second part, let us assume, by way of contradiction, that $\mathcal{I}$ does not satisfy progression. Then, there exists a database D and a set of constraints Σ such that $D \not\models \Sigma$, and for every operation $o \in O$ on D it holds that $\mathcal{I}(\Sigma, o(D)) \geq \mathcal{I}(\Sigma, D)$. In addition, $\mathcal{I}$ satisfies positivity; thus, it holds that $\mathcal{I}(\Sigma, D) > 0$. Since $\mathbf{C}$ is realizable by $\mathcal{R}$, there exists a sequence of operations $o_1, \ldots, o_n$ from $\mathcal{R}$, such that $o_n \circ \cdots \circ o_1(D)$ is consistent (hence, $\mathcal{I}(\Sigma, o_n \circ \cdots \circ o_1(D)) = 0$). We conclude that there exists an operation o_j such that $\Delta_{\mathcal{I},\Sigma}(o_j, o_{j-1} \circ \cdots \circ o_1(D)) > 0$, but there is no such operation on D, which is a contradiction to the fact that $\mathcal{I}$ satisfies continuity, and that concludes our proof. □

A.2 Proof of Theorem 1

Theorem 1. *Let* $\mathcal{R} = \mathcal{R}_{\subseteq}$*, and let* Σ *be a set of constraints that contains a single EGD* σ *with two binary atoms. If* σ *is of the following form:*

$$\forall x_1, x_2, x_3[R(x_1, x_2), R(x_2, x_3) \Rightarrow (x_i = x_j)]$$

then computing $\mathcal{I}_{\mathcal{R}}(\Sigma, D)$ *is NP-hard. In any other case,* $\mathcal{I}_{\mathcal{R}}(\Sigma, D)$ *can be computed in polynomial time.*

We start by proving the negative side of the theorem. That is, we prove the following.

Lemma 1. *Let* $\mathcal{R} = \mathcal{R}_{\subseteq}$ *and let* $\Sigma = \{\sigma\}$ *where* σ *is an EGD of the form* $\forall x_1, x_2, x_3\,[R(x_1, x_2), R(x_2, x_3) \Rightarrow (x_i = x_j)]$*. Then, computing* $\mathcal{I}_{\mathcal{R}}(\Sigma, D)$ *is NP-hard.*

Proof. We build a reduction from the MaxCut problem to the problem of computing $\mathcal{I}_{\mathcal{R}}(\Sigma, D)$ for $\Sigma = \{\sigma\}$. The MaxCut problem is the problem of finding a cut in a graph (i.e., a partition of the vertices into two disjoint subsets), such that the number of edges crossing the cut is the highest among all possible cuts. This problem is known to be NP-hard.

Given a graph g, with n vertices and m edges, we construct an input to our problem (that is, a database D) as follows. For each vertex v_i we add the following two facts to the database:

$$R(1, v_i), R(v_i, 2)$$

Moreover, for each edge (v_i, v_j), we add the following two facts to the database:

$$R(v_j, v_i), R(v_i, v_j)$$

Note that for each vertex v_i, the facts $R(1, v_i)$ and $R(v_i, 2)$ violate the EGD together. Moreover, two facts of the form $R(1, v_i)$ and $R(v_i, v_j)$ jointly violate the EGD, and two facts of the form $R(v_i, 2)$ and $R(v_j, v_i)$ jointly violate the EGD. Finally, two facts of the form $R(v_i, v_j)$ and $R(v_j, v_k)$ violate the FD with each other. These are the only violations of the EGD in the database.

We set the cost $\kappa(o, D)$ to be 1 when the operation o is a deletion of a fact of the form $R(v_i, v_j)$, and we set $\kappa(o, D)$ to be $m + 1$ when the operation o is a deletion of a fact of the form $R(1, v_i)$ or $R(v_i, 2)$. We now prove that there is a cut of size at least k, if and only if

$$\mathcal{I}_{\mathcal{R}}(\Sigma, D) \leq (m + 1) \cdot n + 2(m - k) + k$$

First, assume that there exists a cut of size k in the graph, that partitions the vertices into two groups - S_1 and S_2. In this case, we can remove the following facts from D to obtain a consistent subset D'.

- $R(1, v_i)$ if $v_i \in S_2$,
- $R(v_i, 2)$ if $v_i \in S_1$,
- $R(v_j, v_i)$ if either $R(1, v_j)$ or $R(v_i, 2)$ have not been removed.

Each vertex v_i belongs to either S_1 or S_2; hence, we remove exactly one of the facts $R(1, v_i)$ and $R(v_i, 2)$ for each v_i, and resolve the conflict between these two facts. The cost of removing these n facts is $(m + 1) \cdot n$. Next, for each edge (v_i, v_j) such that both v_i and v_j belong to the same subset S_k, we remove both $R(v_j, v_i)$ and $R(v_i, v_j)$, since the first violates the EDG with $R(1, v_j)$ if $v_i, v_j \in S_1$ or with $R(v_i, 2)$ if $v_i, v_j \in S_2$, and the second violates the EDG with $R(1, v_i)$ if $v_i, v_j \in S_1$ or with $R(v_j, 2)$ if $v_i, v_j \in S_2$. The cost of removing these $2(m - k)$ facts is $2(m - k)$.

Finally, for each edge (v_i, v_j) that crosses the cut, we remove one of $R(v_j, v_i)$ or $R(v_i, v_j)$ from the database. If $v_i \in S_1$ and $v_j \in S_2$, then we have already removed the facts $R(1, v_j)$ and $R(v_i, 2)$ from the database; thus, the fact $R(v_j, v_i)$ does not violate the EGD with any other fact, and we only have to remove the fact $R(v_i, v_j)$ that violates the EGD with both $R(1, v_i)$ and $R(v_j, 2)$. Similarly, if $v_i \in S_2$ and $v_j \in S_1$, we only remove the fact $R(v_j, v_i)$. The cost of removing these k facts is k. Hence, the total cost of removing all these facts is $(m + 1) \cdot n + 2(m - k) + k$.

Clearly, the result is a consistent subset D' of D. As explained above, we have resolved the conflict between $R(1, v_i)$ and $R(v_i, 2)$ for each v_i, and we have resolved the conflict between every pair $\{R(v_i, v_j), R(1, v_i)\}$ of conflicting facts and every pair $\{R(v_i, v_j), R(v_j, 2)\}$ of conflicting facts. Finally, there are no conflicts among facts $R(v_i, v_j)$ and $R(v_j, v_k)$ in D' since v_j either belongs to S_1, in which case the fact $R(1, v_j)$ appears in D' and $R(v_j, v_k)$ has been removed from D' as a result, or it belongs to S_2, in which case the fact $R(v_j, 2)$ appears in D' and $R(v_i, v_j)$ has been removed from D' as a result. Thus, the minimal cost of obtaining a consistent subset of D (i.e., a repair) is at most $(m + 1) \cdot n + 2(m - k) + k$.

Next, we assume that $\mathcal{I}_{\mathcal{R}}(\Sigma, D) \leq (m + 1) \cdot n + 2(m - k) + k$, and we prove that there exists a cut of size at least

k in the graph. First, note that if there exists a consistent subset D' of D that can be obtained with cost at most $(m+1)\cdot n+2(m-k)+k$, such that both $R(1, v_i)$ and $R(v_i, 2)$ have been deleted, then we can obtain another consistent subset of D with a lower cost by removing only $R(1, v_i)$ and removing all the facts of the form $R(v_j, v_i)$ instead of removing $R(v_i, 2)$. There are at most m such facts (if v_i appears in every clause) and the cost of removing them is at most m, while the cost of removing $R(v_i, 2)$ is $m + 1$. Hence, from now on we assume that the subset D' contains exactly one fact from $\{R(1, v_i), R(v_i, 2)\}$ for each v_i.

Now, we construct a cut in the graph from D' in the following way. For each v_i, if the fact $R(1, v_i)$ belongs to D', then we put v_i in S_1, and if the fact $R(v_i, 2)$ belongs to D', then we put v_i in S_2. As mentioned above, exactly one of these two cases holds for each v_i. It is only left to show that the size of the cut is at least k. Since the cost of removing the facts of the form $R(1, v_i)$ and $R(v_i, 2)$ is $(m + 1) \cdot n$, and the cost of removing each fact of the form $R(v_i, v_j)$ is one, at most $2(m - k) + k$ facts of the form $R(v_i, v_j)$ have been removed from D to obtain D'. There are $2m$ facts of the form $R(v_i, v_j)$ in D; thus, at least k of them belong to D'.

For each fact $R(v_i, v_j)$ in D' it holds that both $R(1, v_i)$ and $R(v_j, 2)$ do not belong to D' (otherwise, D' is inconsistent). Hence, it holds that $v_i \in S_2$ and $v_j \in S_1$, and the corresponding edge (v_i, v_j) crosses the cut. We conclude that there are at least k edges that cross the cut. □

We now move on to the proof of the positive side of the theorem. We first consider the case where the EGD contains two different relations and show that in this case $\mathcal{I}_{\mathcal{R}}(\Sigma, D)$ can always be computed in polynomial time.

Lemma 2. *Let $\mathcal{R} = \mathcal{R}_{\subseteq}$ and let $\Sigma = \{\sigma\}$ where σ is an EGD of the form*

$$\forall x_1, x_2, x_3, x_4 \big[R(x_{i_1}, x_{i_2}), S(x_{j_1}, x_{j_2}) \Rightarrow (x_i = x_j)\big]$$

Then, computing $\mathcal{I}_{\mathcal{R}}(\Sigma, D)$ can be done in polynomial time.

Proof. First, if $x_{i_1} = x_{i_2}$, then facts of the form $R(\mathtt{a}, \mathtt{b})$ for $\mathtt{a} \neq \mathtt{b}$ will never participate in a violation; hence we can ignore these facts when computing a minimum repair and add them to the repair later. Similarly, if $x_{i_3} = x_{i_4}$, then we can ignore all the facts of the form $R(\mathtt{a}, \mathtt{b})$ for $\mathtt{a} \neq \mathtt{b}$. Thus, from now on we assume that the database only contains facts that may participate in a violation.

If $R(x_{i_1}, x_{i_2})$ and $S(x_{j_1}, x_{j_2})$ share variables (e.g., $x_{i_2} = x_{j_1}$), then two facts will violate the EGD only if they agree on the values of the shared variables. Hence, we first split the database into blocks of facts that agree on the values of the shared variables, and we solve the problem separately for each one of these blocks, since there are no violations among facts in different blocks. Then, a minimum repair for the entire database will be the disjoint union of the minimum repairs for each one of the blocks. For example, if the only shared variable is $x_{i_2} = x_{j_1}$, then each block will contain facts of the form $R(\cdot, \mathtt{a})$, $S(\mathtt{a}, \cdot)$ for some value $\mathtt{a}$. Note that if the two atoms do not share any variables, then we will have one block that contains all the facts in the database.

We start by considering the case where x_i and x_j both appear in either $R(x_{i_1}, x_{i_2})$ or in $S(x_{j_1}, x_{j_2})$. We assume, without the loss of generality, that they both appear in $R(x_{i_1}, x_{i_2})$ (that is, $x_i = x_{i_1}$ and $x_j = x_{i_2}$). This means, that as long as there is at least one fact in S that belongs to the current block, all the facts in R that belong to the block should be of the form $R(\mathtt{a}, \mathtt{a})$ for some value $\mathtt{a}$. Hence, we have two possible ways to obtain a consistent subset of a block: *(a)* remove all the facts from S, or *(b)* remove from R all the facts $R(\mathtt{a}, \mathtt{b})$ such that $\mathtt{a} \neq \mathtt{b}$. We can compute the cost of each one of these options and choose the one with the lower cost.

In the case where x_i appears only in $R(x_{i_1}, x_{i_2})$ and x_j appears only in $S(x_{j_1}, x_{j_2})$, we have three possible ways to obtain a consistent subset of a block: *(a)* remove all the facts from R, *(b)* remove all the facts from S, or *(c)* choose a single value that will appear in all the attributes corresponding to the variables x_i and x_j, and remove from R and S all the facts that use different values in these attributes. We can again compute the cost of each one of these options and choose the one with the lowest cost. □

Next, we consider EGDs that use a single relation, such that the two atoms in the EGD do not share any variables.

Lemma 3. *Let $\mathcal{R} = \mathcal{R}_{\subseteq}$ and let $\Sigma = \{\sigma\}$ where σ is an EGD of the form*

$$\forall x_1, x_2, x_3, x_4 \big[R(x_1, x_2), R(x_3, x_4) \Rightarrow (x_i = x_j)\big]$$

Then, computing $\mathcal{I}_{\mathcal{R}}(\Sigma, D)$ can be done in polynomial time.

Proof. If x_i and x_j are both from either $\{x_1, x_2\}$ or $\{x_3, x_4\}$, then each fact of the form $R(\mathtt{a}, \mathtt{b})$ such that $\mathtt{a} \neq \mathtt{b}$ violates the EGD by itself. Thus, we have to remove from the database all such facts, and we can compute the cost of removing these facts in polynomial time. Otherwise, x_i is one of x_1, x_2 and x_j is one of x_3, x_4, in which case we have to choose some value $\mathtt{a}$ and remove from the database all the facts that do not use this value in both the attributes corresponding to x_i, x_j.

If $x_i = x_1$ and $x_j = x_3$, then all the facts in the database should agree on the value of the first attribute, and we have to choose the value that entails the lowest cost (that is, the cost of removing all the facts that use a different value in the first attribute is the lowest among all possible values). The case where $x_i = x_2$ and $x_j = x_4$ is symmetric to this case.

If $x_i = x_2$ and $x_j = x_3$, then again each fact will violate the EGD by itself unless it is of the form $R(\mathtt{a}, \mathtt{a})$. In addition, two facts $R(\mathtt{a}, \mathtt{a})$ and $R(\mathtt{b}, \mathtt{b})$ for $\mathtt{a} \neq \mathtt{b}$ will jointly violate the EGD. Thus, only one fact of the form $R(\mathtt{a}, \mathtt{a})$ will not be removed from the database, and we again have to choose the one that entails the lowest cost. The case where $x_i = x_1$ and $x_j = x_3$ is symmetric to this case. Hence, computing $\mathcal{I}_{\mathcal{R}}(\Sigma, D)$ can again be done in polynomial time. □

So far, we have covered the cases where the EGD either considers two different relations or considers only one relation, but there are no shared variables among the atoms. In both cases, computing $\mathcal{I}_{\mathcal{R}}(\Sigma, D)$ can always be done in polynomial time. Next, we consider the cases where the EGD uses a single relation and the two atoms do share variables, but the EGD does not satisfy the condition of the theorem.

Lemma 4. *Let $\mathcal{R} = \mathcal{R}_{\subseteq}$ and let $\Sigma = \{\sigma\}$ where σ is an EGD of one of the following forms:*

1. $\forall x_1, x_2 \big[R(x_1, x_2), R(x_1, x_2) \Rightarrow (x_1 = x_2)\big]$
2. $\forall x_1, x_2, x_3 \big[R(x_1, x_2), R(x_1, x_3) \Rightarrow (x_i = x_j)\big]$
3. $\forall x_1, x_2 \big[R(x_1, x_2), R(x_2, x_1) \Rightarrow (x_1 = x_2)\big]$

Then, computing $\mathcal{I}_{\mathcal{R}}(\Sigma, D)$ can be done in polynomial time.

Proof. It is straightforward that computing $\mathcal{I}_{\mathcal{R}}(\Sigma, D)$ w.r.t. an EGD of the first form can be done in polynomial time, as each fact in R violates the EGD by itself, unless it uses the same value in both attributes. Thus, we have to remove from the database all the facts of the form $R(\mathtt{a}, \mathtt{b})$ for $\mathtt{a} \neq \mathtt{b}$, and we can compute the cost of removing these facts in polynomial time.

For the second case, if $x_i = x_2$ and $x_j = x_3$, then an EGD of this form is an FD, and $\mathcal{I}_{\mathcal{R}}(\Sigma, D)$ can be computed in polynomial time [Livshits *et al.*, 2018]. If $x_i, x_j \in \{x_1, x_2\}$ or $x_i, x_j \in \{x_1, x_3\}$, then we are again in the case where only facts of the form $R(\mathtt{a}, \mathtt{a})$ are allowed, and we have to remove the rest of the facts from the database.

Finally, for EGDs of the third form, it holds that only pairs of facts $\{R(\mathtt{a}, \mathtt{b}), R(\mathtt{b}, \mathtt{a})\}$ violate the EGD; hence, for each such pair we have to remove one of facts in the pair, and we will remove the one that entails the lower cost. □

A.3 Proof of Theorem 2

Theorem 2. *The following hold for $\mathbf{C} = \mathbf{C}_{\mathsf{DC}}$ and $\mathcal{R} = \mathcal{R}_{\subseteq}$.*

1. *$\mathcal{I}^{\mathsf{lin}}_{\mathcal{R}}$ satisfies positivity, monotonicity, progression and constant weighted continuity.*
2. *$\mathcal{I}^{\mathsf{lin}}_{\mathcal{R}}$ can be computed in polynomial time (in data complexity).*

Proof. The tractability of $\mathcal{I}^{\mathsf{lin}}_{\mathcal{R}}$ is simply due to the fact that we can enumerate $\mathsf{MI}_{\Sigma}(D)$ in polynomial time, hence construct the LP, and then use any polynomial-time LP solver [Khachiyan, 1979]. So, in the remainder of the proof, we prove that the four postulates are satisfied. By "the LP program" we refer to the linear relaxation of the ILP program of Figure 1, where the bottom condition is replaced with $\forall i \in ids(D) : 0 \leq x_i \leq 1$.

Proving *positivity* is straightforward. In particular, if D violates Σ, then $\mathsf{MI}_{\Sigma}(D)$ is nonempty; hence, the zero assignment (i.e., assigning zero to every x_i) is infeasible due to the first constraint, and the resulting value of the objective function is strictly positive.

For *monotonicity*, let D be a database, and let Σ and Σ' be two sets of DCs such that $\Sigma' \models \Sigma$. To prove that $\mathcal{I}^{\mathsf{lin}}_{\mathcal{R}}(\Sigma, D) \leq \mathcal{I}^{\mathsf{lin}}_{\mathcal{R}}(\Sigma', D)$ it suffices to prove that every feasible solution w.r.t. Σ' is also a feasible solution w.r.t. Σ. More specifically, if an assignment to the x_i's satisfies the set (1) of inequalities for $\mathsf{MI}_{\Sigma'}(D)$, then it also satisfies (1) for $\mathsf{MI}_{\Sigma}(D)$. For that, it suffices to show that for every $E \in \mathsf{MI}_{\Sigma}(D)$ there is $E' \in \mathsf{MI}_{\Sigma'}(D)$ such that $E' \subseteq E$. This is straightforward from our assumption that $\Sigma' \models \Sigma$: if $E \in \mathsf{MI}_{\Sigma}(D)$, then $E \not\models \Sigma$; hence, $E \not\models \Sigma'$, and E contains a minimal inconsistent subset $E' \in \mathsf{MI}_{\Sigma'}(D)$.

As for *progression*, let D be a database, and let Σ be a set of DCs such that D violates Σ. Let μ be an assignment to the x_i's of the LP such that μ realizes $\mathcal{I}^{\mathsf{lin}}_{\mathcal{R}}(\Sigma, D)$. Take any $j \in ids(D)$ such that $x_j > 0$. Let D' be the database obtained from D by removing $D[j]$, and let μ' be the assignment obtained by restricting μ to $ids(D) \setminus \{j\}$. Then, μ' is a feasible solution, and its objective value is smaller than that of μ. We conclue that $\mathcal{I}^{\mathsf{lin}}_{\mathcal{R}}(\Sigma, D') < \mathcal{I}^{\mathsf{lin}}_{\mathcal{R}}(\Sigma, D)$.

Finally, we prove constant weighted *continuity*. Let Σ be a set of DCs, let D_1 and D_2 be two inconsistent databases, and let $o_1 = \langle -i \rangle(\cdot)$ be an operation on D_1. We need to find an operation $o_2 = \langle -j \rangle(\cdot)$ on D_2 such that $\Delta_{\mathcal{I}^{\mathsf{lin}}_{\mathcal{R}}, \Sigma}(o_1, D_1) \leq \delta \cdot \Delta_{\mathcal{I}^{\mathsf{lin}}_{\mathcal{R}}, \Sigma}(o_2, D_2)$ for some constant δ that depends only on Σ. Let μ_1 be an assignment for the LP that realizes $\mathcal{I}^{\mathsf{lin}}_{\mathcal{R}}(\Sigma, D_1)$, and let μ'_1 be an assignment for the LP that realizes $\mathcal{I}^{\mathsf{lin}}_{\mathcal{R}}(\Sigma, o_1(D_1))$. Let μ''_1 be the assignment that is obtained by extending μ'_1 with $\mu''_1(x_i) = 1$; that is, $\mu''_1(x_j) = \mu'_1(x_j)$ for $j \in ids(D_1) \setminus \{i\}$, and $\mu''_1(x_j) = 1$ for $j = i$. Then the objective value for μ''_1 is greater than the objective value for μ'_1 by at most $\kappa(o_1, D_1)$. Moreover, μ''_1 is a feasible solution w.r.t. D_1. We conclude that

$$\Delta_{\mathcal{I}^{\mathsf{lin}}_{\mathcal{R}}, \Sigma}(o_1, D_1) \leq \kappa(o_1, D_1)\,. \tag{3}$$

Now, let μ_2 be an assignment that realizes $\mathcal{I}^{\mathsf{lin}}_{\mathcal{R}}(\Sigma, D_2)$, and let E be any set in $\mathsf{MI}_{\Sigma}(D_2)$. Then, from the definition of the LP it follows that there exists a tuple identifier j such that $D_2[j] \in E$ and $\mu_2(x_j) \geq 1/|E|$. Therefore, by removing $D_2[j]$ we get a feasible solution w.r.t. D_2 such that the reduction in inconsistency is at least $\kappa(o_2, D_2)/|E|$ for $o_2 = \langle -j \rangle(\cdot)$. We conclude that

$$\Delta_{\mathcal{I}^{\mathsf{lin}}_{\mathcal{R}}, \Sigma}(o_2, D_2) \geq \frac{\kappa(o_2, D_2)}{|E|}\,. \tag{4}$$

Combining (3) and (4), we conclude that

$$\Delta_{\mathcal{I}^{\mathsf{lin}}_{\mathcal{R}}, \Sigma}(o_1, D_1) \leq |E| \cdot \frac{\kappa(o_1, D_1)}{\kappa(o_2, D_2)} \cdot \Delta_{\mathcal{I}^{\mathsf{lin}}_{\mathcal{R}}, \Sigma}(o_2, D_2)\,.$$

Finally, we observe that the cardinality $|E|$ is bounded by the maximal number of atoms in any DC of Σ. Denoting this maximal number by d_{Σ}, we conclude constant weighted continuity by taking $\delta = d_{\Sigma}$. □