The Shapley Value of Inconsistency Measures for Functional Dependencies

Ester Livshits
Technion - Israel Institute of Technology, Haifa, Israel

Benny Kimelfeld
Technion - Israel Institute of Technology, Haifa, Israel

Abstract

Quantifying the inconsistency of a database is motivated by various goals including reliability estimation for new datasets and progress indication in data cleaning. Another goal is to attribute to individual tuples a level of responsibility to the overall inconsistency, and thereby prioritize tuples in the explanation or inspection of dirt. Therefore, inconsistency quantification and attribution have been a subject of much research in Knowledge Representation and, more recently, in Databases. As in many other fields, a conventional responsibility sharing mechanism is the Shapley value from cooperative game theory. In this paper, we carry out a systematic investigation of the complexity of the Shapley value in common inconsistency measures for functional-dependency (FD) violations. For several measures we establish a full classification of the FD sets into tractable and intractable classes with respect to Shapley-value computation. We also study the complexity of approximation in intractable cases.

2012 ACM Subject Classification Theory of computation → Incomplete, inconsistent, and uncertain databases; Information systems → Data cleaning

Keywords and phrases Shapley value, inconsistent databases, functional dependencies, database repairs

Digital Object Identifier 10.4230/LIPIcs.ICDT.2021.15

Related Version *Full Version*: `https://arxiv.org/abs/2009.13819`

Funding This work was supported by the Israel Science Foundation (ISF), Grant 768/19, and the German Research Foundation (DFG) Project 412400621 (DIP program).

1 Introduction

Inconsistency measures for knowledge bases have received considerable attention from the Knowledge Representation (KR) and Logic communities [9,13,15–17,19,20,37]. More recently, inconsistency measures have also been studied from the database viewpoint [2,24]. Such measures quantify the extent to which the database violates a set of integrity constraints. There are multiple reasons why one might be using such measures. For one, the measure can be used for estimating the usefulness or reliability of new datasets for data-centric applications such as business intelligence [6]. Inconsistency measures have also been proposed as the basis of progress indicators for data-cleaning systems [24]. Finally, the measure can be used for attributing to individual tuples a level of responsibility to the overall inconsistency [30,36], thereby prioritize tuples in the explanation/inspection/correction of errors.

A conventional approach to dividing the responsibility for a quantitative property (here an inconsistency measure) among entities (here the database tuples) is the *Shapley value* [35], which is a game-theoretic formula for wealth distribution in a cooperative game. The Shapley value has been applied in a plethora of domains, including economics [14], law [32], environmental science [22,33], social network analysis [31], physical network analysis [29], and advertisement [5]. In data management, the Shapley value has been used for determining the relative contribution of features in machine-learning predictions [21,28], the responsibility of tuples to database queries [4,23,34], and the reliability of data sources [6].

The Shapley value has also been studied in a context similar to the one we adopt in this paper – assigning a level of inconsistency to statements in an inconsistent knowledge base [17,30,36,39]. Hunter and Konieczny [15–17] use the maximal Shapley value of one inconsistency measure in order to define a new inconsistency measure. Grant and Hunter [12] considered information systems distributed along data sources of different reliabilities, and apply the Shapley value to determine the expected blame of each statement to the overall inconsistency. Yet, with all the investigation that has been conducted on the Shapley value of inconsistency, we are not aware of any results or efforts regarding the computational complexity of calculating this value.

In this work, we embark on a systematic analysis of the complexity of the Shapley value of database tuples relative to inconsistency measures, where the goal is to calculate the contribution of a tuple to inconsistency. Our main results are summarized in Table 1. We consider inconsistent databases with respect to functional dependencies (FDs), and basic measures of inconsistency following Bertossi [3] and Livshits, Ilyas, Kimelfeld and Roy [24]. We note that these measures are all adopted from the measures studied in the aforementioned KR research. In our setting, an individual tuple affects the inconsistency of only its containing relation, since the constraints are FDs. Hence, we focus on databases with a single relation. While most of our results extend to multiple relations, some extensions require a more subtle proof. We discuss the extension beyond a single relation in Section 8.

More formally, we investigate the following computational problem for any fixed combination of a relational signature, a set of FDs, and an inconsistency measure: given a database and a tuple, compute the Shapley value of the tuple with respect to the inconsistency measure. As Table 1 shows, two of these measures are computable in polynomial time: $\mathcal{I}_{\mathsf{MI}}$ (number of FD violations) and $\mathcal{I}_{\mathsf{P}}$ (number of problematic facts that participate in violations). For two other measures, we establish a full dichotomy in the complexity of the Shapley value: $\mathcal{I}_{\mathsf{d}}$ (the drastic measure – 0 for consistency and 1 for inconsistency) and $\mathcal{I}_{\mathsf{MC}}$ (number of maximal consistent subsets, a.k.a. repairs). The dichotomy in both cases is the same: when the FD set has, up to equivalence, an lhs chain (i.e., the left-hand sides form a chain w.r.t. inclusion [25]), the Shapley value can be computed in polynomial time; in any other case, it is $\mathrm{FP}^{\#\mathrm{P}}$-hard. In the case of $\mathcal{I}_{\mathsf{R}}$ (the minimal number of tuples to delete for consistency), the problem is solvable in polynomial time in the case of an lhs chain, and NP-hard whenever it is intractable to find a cardinality repair [27]; however, the problem is open for every FD set in between, for example, the bipartite matching constraint $\{A \rightarrow B, B \rightarrow A\}$.

We also study the complexity of approximating the Shapley value and show the following (as described in Table 1). First, in the case of $\mathcal{I}_{\mathsf{d}}$, there is a (multiplicative) fully polynomial-time approximation scheme (FPRAS) for every set of FDs. In the case of $\mathcal{I}_{\mathsf{MC}}$, approximating the Shapley value of *any* intractable (non-lhs-chain) FD set is at least as hard as approximating the number of maximal matchings of a bipartite graph – a long standing open problem [18]. In the case of $\mathcal{I}_{\mathsf{R}}$, we establish a full dichotomy, namely FPRAS vs. hardness of approximation, that has the same separation as the problem of finding a cardinality repair

The rest of this paper is organized as follows. After presenting the basic notation and terminology in Section 2, we formally define the studied problem and give initial observations in Section 3. In Section 4, we describe polynomial-time algorithms for $\mathcal{I}_{\mathsf{MI}}$ and $\mathcal{I}_{\mathsf{P}}$. Then, we explore the measures $\mathcal{I}_{\mathsf{d}}$, $\mathcal{I}_{\mathsf{R}}$ and $\mathcal{I}_{\mathsf{MC}}$ in Sections 5, 6 and 7, respectively. We conclude and discuss future directions in Section 8. Some proofs appear only in the archive version [26] and will be given in the full version of the paper.

Table 1 Complexity of the (exact ; approximate) Shapley value of inconsistency measures.

	lhs chain	**no lhs chain, PTime c-repair**	**other**
$\mathcal{I}_{\mathsf{d}}$	PTime	$\mathrm{FP}^{\#\mathrm{P}}$-complete ; FPRAS	
$\mathcal{I}_{\mathsf{MI}}$	PTime		
$\mathcal{I}_{\mathsf{P}}$	PTime		
$\mathcal{I}_{\mathsf{R}}$	PTime	**?** ; FPRAS	NP-hard [27] ; no FPRAS
$\mathcal{I}_{\mathsf{MC}}$	PTime	$\mathrm{FP}^{\#\mathrm{P}}$-complete [25] ; **?**	

2 Preliminaries

Database concepts. By a *relational schema* we refer to a sequence $(A_1, \dots, A_n)$ of attributes. A database D over $(A_1, \dots, A_n)$ is a finite set of tuples, or *facts*, of the form $(c_1, \dots, c_n)$, where each c_i is a constant from a countably infinite domain. For a fact f and an attribute A_i, we denote by $f[A_i]$ the value associated by f with the attribute A_i (that is, $f[A_i] = c_i$). Similarly, for a sequence $X = (A_{j_1}, \dots, A_{j_m})$ of attributes, we denote by $f[X]$ the tuple $(f[A_{j_1}], \dots, f[A_{j_m}])$. Generally, we use letters from the beginning of the English alphabet (i.e., $A, B, C, \dots$) to denote single attributes and letters from the end of the alphabet (i.e., $X, Y, Z, \dots$) to denote sets of attributes. We may omit stating the relational schema of a database D when it is clear from the context or irrelevant.

A *Functional Dependency* (FD for short) over $(A_1, \dots, A_n)$ is an expression of the form $X \rightarrow Y$, where $X, Y \subseteq \{A_1, \dots, A_m\}$. We may also write the attribute sets X and Y by concatenating the attributes (e.g., $AB \rightarrow C$ instead of $\{A, B\} \rightarrow \{C\}$). A database D satisfies $X \rightarrow Y$ if every two facts $f, g \in D$ that agree on the values of the attributes of X also agree on the values of the attributes of Y (that is, if $f[X] = g[X]$ then $f[Y] = g[Y]$). A database D *satisfies* a set Δ of FDs, denoted by $D \models \Delta$, if D satisfies every FD of Δ. Otherwise, D *violates* Δ (denoted by $D \not\models \Delta$). Two FD sets over the same relational schema are *equivalent* if every database that satisfies one of them also satisfies the other.

Let Δ be a set of FDs and D a database (which may violate Δ). A *repair* (*of D w.r.t.* Δ) is a maximal consistent subset of D; that is, $E \subseteq D$ is a repair if $E \models \Delta$ but $E' \not\models \Delta$ for every $E \subsetneq E'$. A *cardinality repair* (or *c-repair* for short) is a repair of maximum cardinality; that is, it is a repair E such that $|E| \geq |E'|$ for every repair E'.

▶ **Example 1.** Figure 1 depicts an inconsistent database over a relational schema that stores a train schedule. For example, the fact f_1 states that train number 16 will depart from the New York Penn Station at time 1030 and arrive to the Boston Back Bay Station after 315 minutes. The FD set Δ consists of the two FDs:

- train time $\rightarrow$ departs
- train time duration $\rightarrow$ arrives

The first FD states that the departure station is determined by the train number and departure time, and the second FD states that the arrival station is determined by the train number, the departure time, and the duration of the ride.

Observe that the database of Figure 1 violates the FDs as all the facts refer to the same train number and departure time, but there is no agreement on the departure station. Moreover, the facts f_6 and f_7 also agree on the duration, but disagree on the arrival station. The database has five repairs: *(a)* $\{f_1, f_2\}$, *(b)* $\{f_3, f_4, f_5\}$, *(c)* $\{f_6, f_8\}$, *(d)* $\{f_7, f_8\}$, and *(e)* $\{f_9\}$; only the second one is a cardinality repair. $\Diamond$

fact	train	departs	arrives	time	duration
f_1	16	NYP	BBY	1030	315
f_2	16	NYP	PVD	1030	250
f_3	16	PHL	WIL	1030	20
f_4	16	PHL	BAL	1030	70
f_5	16	PHL	WAS	1030	120
f_6	16	BBY	PHL	1030	260
f_7	16	BBY	NYP	1030	260
f_8	16	BBY	WAS	1030	420
f_9	16	WAS	PVD	1030	390

Figure 1 The inconsistent database of our running example.

Shapley value. A *cooperative game* of a set A of players is a function $v : \mathcal{P}(A) \to \mathbb{R}$, where $\mathcal{P}(A)$ is the power set of A, such that $v(\emptyset) = 0$. The value $v(B)$ should be thought of as the joint wealth obtained by the players of B when they cooperate. The *Shapley value* of a player $a \in A$ measures the contribution of a to the total wealth $v(A)$ of the game [35], and is formally defined by

$$\text{Shapley}(A, v, a) \stackrel{\text{def}}{=} \frac{1}{|A|!} \sum_{\sigma \in \Pi_A} (v(\sigma_a \cup \{a\}) - v(\sigma_a))$$

where Π_A is the set of all permutations over the players of A and σ_a is the set of players that appear before a in the permutation σ. Intuitively, the Shapley value of a player a is the expected contribution of a to a subset constructed by drawing players randomly one by one (without replacement), where the contribution of a is the change to the value of v caused by the addition of a. An alternative formula for the Shapley value, that we will use in this paper, is the following.

$$\text{Shapley}(A, v, a) \stackrel{\text{def}}{=} \sum_{B \subseteq A \setminus \{a\}} \frac{|B|! \cdot (|A| - |B| - 1)!}{|A|!} \Big(v(B \cup \{a\}) - v(B)\Big)$$

Observe that $|B|! \cdot (|A| - |B| - 1)!$ is the number of permutations where the players of B appear first, then a, and then the rest of the players.

Approximation schemes. We discuss both exact and approximate algorithms for computing Shapley values. Recall that a *Fully-Polynomial Randomized Approximation Scheme* (FPRAS, for short) for a function f is a randomized algorithm $A(x, \epsilon, \delta)$ that returns an ϵ-approximation of $f(x)$ with probability at least $1 - \delta$, given an input x for f and $\epsilon, \delta \in (0, 1)$, in time polynomial in x, $1/\epsilon$, and $\log(1/\delta)$. Formally, an FPRAS, satisfies:

$$\Pr\left[f(x)/(1 + \epsilon) \le A(x, \epsilon, \delta) \le (1 + \epsilon) f(x)\right] \ge 1 - \delta\,.$$

Note that this notion of FPRAS refers to a *multiplicative* approximation, and we adopt this notion implicitly unless stated otherwise. We may also write "multiplicative" explicitly for stress. In cases where the function f has a bounded range, it also makes sense to discuss an *additive* FPRAS where $\Pr\left[f(x) - \epsilon \le A(x, \epsilon, \delta) \le f(x) + \epsilon\right] \ge 1 - \delta$. We refer to an additive FPRAS, and explicitly state so, in cases where the Shapley value is in the range $[0, 1]$.

3 The Shapley Value of Inconsistency Measures

In this paper, we study the Shapley value of facts with respect to measures of database inconsistency. More precisely, the cooperative game that we consider here is determined by an inconsistency measure $\mathcal{I}$, and the facts of the database take the role of the players. In turn, an *inconsistency measure* $\mathcal{I}$ is a function that maps pairs (D, Δ) of a database D and a set Δ of FDs to a number $\mathcal{I}(D, \Delta) \in [0, \infty)$. Intuitively, the higher the value $\mathcal{I}(D, \Delta)$ is, the more inconsistent (or, the less consistent) the database D is w.r.t. Δ. The Shapley value of a fact f of a database D w.r.t. an FD set Δ and inconsistency measure $\mathcal{I}$ is then defined as follows.

$$\mathrm{Shapley}(D, \Delta, f, \mathcal{I}) \stackrel{\text{def}}{=} \sum_{E \subseteq (D \setminus \{f\})} \frac{|E|! \cdot (|D| - |E| - 1)!}{|D|!} \Big(\mathcal{I}(E \cup \{f\}, \Delta) - \mathcal{I}(E, \Delta) \Big) \tag{1}$$

We note that the definition of the Shapley value requires the cooperative game to be zero on the empty set [35] and this is indeed the case for all of the inconsistency measures $\mathcal{I}$ that we consider in this work. Next, we introduce each of these measures.

- $\mathcal{I}_{\mathsf{d}}$ is the *drastic measure* that takes the value 1 if the database is inconsistent and the value 0 otherwise [37].
- $\mathcal{I}_{\mathsf{MI}}$ counts the *minimal inconsistent subsets* [16, 17]; in the case of FDs, these subsets are simply the pairs of tuples that jointly violate an FD.
- $\mathcal{I}_{\mathsf{P}}$ is the number of *problematic facts*, where a fact is problematic if it belongs to a minimal inconsistent subset [10]; in the case of FDs, a fact is problematic if and only if it participates in a pair of facts that jointly violate Δ.
- $\mathcal{I}_{\mathsf{R}}$ is the minimal number of facts that we need to delete from the database for Δ to be satisfied (similarly to the concept of a cardinality repair and proximity in Property Testing) [3, 8, 11].
- $\mathcal{I}_{\mathsf{MC}}$ is the number of *maximal consistent subsets* (i.e., repairs) [10, 13].

Table 1 summarizes the complexity results for the different measures. The first column (lhs chain) refers to FD sets that have a left-hand-side chain – a notion that was introduced by Livshits et al. [25], and we recall in the next section. The second column (no lhs chain, PTime c-repair) refers to FD sets that do not have a left-hand-side chain, but entail a polynomial-time cardinality repair computation according to the dichotomy of Livshits et al. [27] that we discuss in more details in Section 6.

▶ **Example 2.** Consider again the database of our running example. Since the database is inconsistent w.r.t. the FD set defined in Example 1, we have that $\mathcal{I}_{\mathsf{d}}(D, \Delta) = 1$. As for the measure $\mathcal{I}_{\mathsf{MI}}$, the reader can easily verify that there are twenty eight pairs of tuples that jointly violate the FDs; hence, we have that $\mathcal{I}_{\mathsf{MI}}(D, \Delta) = 28$. Since each tuple participates in at least one violation of the FDs, it holds that $\mathcal{I}_{\mathsf{P}} = 9$. Finally, as we have already seen in Example 1, the database has five repairs and a single cardinality repair obtained by deleting six facts. Thus, $\mathcal{I}_{\mathsf{R}}(D, \Delta) = 6$ and $\mathcal{I}_{\mathsf{MC}}(D, \Delta) = 5$. In the next sections, we discuss the computation of the Shapley value for each one of these measures. ◊

Preliminary analysis. We study the *data complexity* of computing $\mathrm{Shapley}(D, \Delta, f, \mathcal{I})$ for different inconsistency measures $\mathcal{I}$. To this end, we give here two important observations that we will use throughout the paper. The first observation is that the computation of $\mathrm{Shapley}(D, \Delta, f, \mathcal{I})$ can be easily reduced to the computation of the expected value of the inconsistency measure over all subsets of the database of a given size. In the following proposition, we denote by $\mathbb{E}_{D'\sim U_m(D\setminus\{f\})}\big(\mathcal{I}(D'\cup\{f\},\Delta)\big)$ the expected value of $\mathcal{I}(D'\cup\{f\},\Delta)$ over all subsets D' of $D\setminus\{f\}$ of a given size m, assuming a uniform distribution. Similarly, $\mathbb{E}_{D'\sim U_m(D\setminus\{f\})}\big(\mathcal{I}(D',\Delta)\big)$ is the expected value of $\mathcal{I}(D',\Delta)$ over all such subsets D'.

▶ **Proposition 3.** *Let $\mathcal{I}$ be an inconsistency measure. The following holds.*

$$\mathrm{Shapley}(D,\Delta,f,\mathcal{I})=\frac{1}{|D|}\sum_{m=0}^{|D|-1}\left[\mathbb{E}_{D'\sim U_m(D\setminus\{f\})}\big(\mathcal{I}(D'\cup\{f\},\Delta)\big)-\mathbb{E}_{D'\sim U_m(D\setminus\{f\})}\big(\mathcal{I}(D',\Delta)\big)\right]$$

Proposition 3 implies that to compute the Shapley value of f, it suffices to compute the expectations of the amount of inconsistency over subsets D' and $D'\cup\{f\}$, where D' is drawn uniformly from the space of subsets of size m, for every m. More precisely, the computation of the Shapley value is Cook reducible[1] to the computation of these expectations. Our algorithms will, indeed, compute these expectations instead of the Shapley value.

The second observation is the following. One of the basic properties of the Shapley value is *efficiency* – the sum of the Shapley values over all the players equals the total wealth [35]. This property implies that $\sum_{f\in D}\mathrm{Shapley}(D,\Delta,f,\mathcal{I})=\mathcal{I}(D,\Delta)$. Thus, whenever the measure itself is computationally hard, so is the Shapley value of facts.

▶ **Fact 1.** *Let $\mathcal{I}$ be an inconsistency measure. The computation of $\mathcal{I}$ is Cook reducible to the computation of the Shapley value of facts under $\mathcal{I}$.*

This observation can be used for showing lower bounds on the complexity of the Shapley value, as we will see in the next sections.

4 Tractable Measures: $\mathcal{I}_{\mathsf{MI}}$ and $\mathcal{I}_{\mathsf{P}}$

We start by discussing two tractable measures. The first measure is $\mathcal{I}_{\mathsf{MI}}$, that counts the *minimal inconsistent subsets* (i.e., pairs of facts that jointly violate at least one FD). An easy observation is that a fact f increases the value of the measure $\mathcal{I}_{\mathsf{MI}}$ by i in a permutation σ if and only if σ_f contains exactly i facts that are in conflict with f. Hence, assuming that D contains N_f facts that conflict with f, the Shapley value for this measure can be computed in the following way:

$$\begin{aligned}\mathrm{Shapley}(D,\Delta,f,\mathcal{I}_{\mathsf{MI}})&=\sum_{E\subseteq(D\setminus\{f\})}\frac{|E|!\cdot(|D|-|E|-1)!}{|D|!}\Big(\mathcal{I}(E\cup\{f\},\Delta)-\mathcal{I}(E,\Delta)\Big)\\&=\frac{1}{|D|!}\sum_{i=1}^{N_f}\sum_{\substack{E\subseteq(D\setminus\{f\})\\|E\cap N_f|=i}}|E|!\cdot(|D|-|E|-1)!\cdot i\\&=\frac{1}{|D|!}\sum_{i=1}^{N_f}\sum_{m=i}^{|D|-1}\sum_{\substack{E\subseteq(D\setminus\{f\})\\|E|=m\\|E\cap N_f|=i}}m!\cdot(|D|-m-1)!\cdot i\\&=\frac{1}{|D|!}\sum_{i=1}^{N_f}\sum_{m=i}^{|D|-1}\binom{N_f}{i}\binom{|D|-N_f-1}{m-i}\cdot m!\cdot(|D|-m-1)!\cdot i\end{aligned}$$

[1] Recall that a *Cook reduction* from a function F to a function G is a polynomial-time *Turing reduction* from F to G, that is, an algorithm that computes F with an oracle to a solver of G.

Therefore, we immediately obtain the following result.

▶ **Theorem 4.** *Let* Δ *be a set of FDs. Computing* $\mathrm{Shapley}(D,\Delta,f,\mathcal{I}_{\mathsf{MI}})$ *can be done in polynomial time, given* D *and* f*.*

The second measure that we consider here is $\mathcal{I}_{\mathsf{P}}$ that counts the "problematic" facts; that is, facts that participate in a violation of Δ. Here, a fact f increases the measure by i in a permutation σ if and only if σ_f contains precisely $i-1$ facts that are in conflict with f, but not in conflict with any other fact of σ_f (hence, all these facts and f itself are added to the group of problematic facts). We prove the following.

▶ **Theorem 5.** *Let* Δ *be a set of FDs. Computing* $\mathrm{Shapley}(D,f,\Delta,\mathcal{I}_{\mathsf{P}})$ *can be done in polynomial time, given* D *and* f*.*

Proof. We now show how the expected values of Proposition 3 can be computed in polynomial time. We start with $\mathbb{E}_{D'\sim U_m(D\setminus\{f\})}\big(\mathcal{I}_{\mathsf{P}}(D',\Delta)\big)$. We denote by X_m the random variable holding the number of problematic facts in a subset of size m of $D\setminus\{f\}$. We denote by X_m^g the random variable that holds 1 if the fact g is selected and participates in a violation of the FDs in such a subset, and 0 otherwise. In addition, we denote the expectations of these variables by $\mathbb{E}(X_m)$ and $\mathbb{E}(X_m^g)$, respectively (without explicitly stating the distribution $D'\sim U_m(D\setminus\{f\})$ in the subscript). Due to the linearity of expectation we have:

$$\mathbb{E}_{D'\sim U_m(D\setminus\{f\})}\big(\mathcal{I}_{\mathsf{P}}(D',\Delta)\big)=\mathbb{E}(X_m)=\mathbb{E}(\sum_{g\in D\setminus\{f\}}X_m^g)=\sum_{g\in D\setminus\{f\}}\mathbb{E}(X_m^g)$$

Hence, the computation of $\mathbb{E}_{D'\sim U_m(D\setminus\{f\})}\big(\mathcal{I}_{\mathsf{P}}(D',\Delta)\big)$ reduces to the computation of $\mathbb{E}(X_m^g)$, and this value can be computed as follows.

$$\begin{aligned}\mathbb{E}(X_m^g)&=\Pr\left[g\text{ is selected in a subset of size }m\right]\times\\&\qquad\Pr\left[\text{a conflicting fact is selected in a subset of size }m\mid g\text{ is selected in the subset}\right]\\&=\frac{\binom{|D|-2}{m-1}}{\binom{|D|-1}{m}}\cdot\frac{\sum_{k=1}^{N_g}\binom{N_g}{k}\cdot\binom{|D|-1-N_g}{m-k-1}}{\binom{|D|-2}{m-1}}=\frac{\sum_{k=1}^{N_g}\binom{N_g}{k}\cdot\binom{|D|-1-N_g}{m-k-1}}{\binom{|D|-1}{m}}\end{aligned}$$

where N_g is the number of facts in $D\setminus\{f\}$ that are in conflict with g.

We can similarly show that $\mathbb{E}_{D'\sim U_m(D\setminus\{f\})}\big(\mathcal{I}_{\mathsf{P}}(D'\cup\{f\},\Delta)\big)=\sum_{g\in D\setminus\{f\}}\mathbb{E}(X_{m,f}^g)$, where $X_{m,f}^g$ is a random variable that holds 1 if g is selected in a subset E of size m of $D\setminus\{f\}$ and participates in a violation of the FDs with the rest of the facts of $E\cup\{f\}$, and 0 otherwise. For a fact g that is not in conflict with f we have that: $\mathbb{E}(X_{m,f}^g)=\mathbb{E}(X_m^g)$, while for a fact g that is in conflict with f it holds that:

$$\mathbb{E}(X_{m,f}^g)=\Pr\left[g\text{ is selected in a subset of size }m\right]=\binom{|D|-2}{m-1}\Big/\binom{|D|-1}{m}$$

and that concludes our proof. ◀

5 The Drastic Measure $\mathcal{I}_{\mathsf{d}}$

In this section, we consider the drastic measure $\mathcal{I}_{\mathsf{d}}$. While the measure itself is extremely simple and, in particular, computable in polynomial time (testing whether Δ is satisfied), it might be intractable to compute the Shapley value of a fact. In particular, we prove a dichotomy for this measure, classifying FD sets into ones where the Shapley value can be computed in polynomial time and the rest where the problem is $\mathrm{FP}^{\#\mathrm{P}}$-complete.[2] Before giving our dichotomy, we recall the definition of a *left-hand-side chain* (lhs chain, for short), introduced by Livshits et al. [25].

▶ **Definition 6.** [25] *An FD set Δ has a left-hand-side chain if for every two FDs $X \to Y$ and $X' \to Y'$ in Δ, either $X \subseteq X'$ or $X' \subseteq X$.*

▶ **Example 7.** The FD set of our running example (Example 1) has an lhs chain. We could also define Δ with redundancy by adding the following FD: train time arrives $\to$ departs. The resulting FD set does not have an lhs chain, but it is *equivalent* to an FD set with an lhs chain. An example of an FD set that does not have an lhs chain, not even up to equivalence, is $\{\text{train time} \to \text{departs}, \text{train departs} \to \text{time}\}$. ◊

We prove the following.

▶ **Theorem 8.** *Let Δ be a set of FDs. If Δ is equivalent to an FD set with an lhs chain, then* $\mathrm{Shapley}(D, f, \Delta, \mathcal{I}_{\mathsf{d}})$ *can be computed in polynomial time, given D and f. Otherwise, the problem is* $\mathrm{FP}^{\#\mathrm{P}}$*-complete.*

Interestingly, this is the exact same dichotomy that we obtained in prior work [25] for the problem of counting subset repairs. We also showed that this tractability criterion is decidable in polynomial time by computing a minimal cover: if Δ is equivalent to an FD set with an lhs chain, then every minimal cover of Δ has an lhs chain.

Proof of Theorem 8. The proof of the hardness side has two steps. We first show hardness for the matching constraint $\{A \to B, B \to A\}$ over the schema (A, B), and this proof is similar to the proof of Livshits et al. [23] for the problem of computing the Shapley contribution of facts to the result of the query $q() \text{ :- } R(x), S(x, y), T(y)$. Then, from this case to the remaining cases we apply the *fact-wise reductions* that we established in our prior work [25]. So, in the remainder of this section we will focus on the tractability side.

As stated in Proposition 3, the computation of $\mathrm{Shapley}(D, f, \Delta, \mathcal{I}_{\mathsf{d}})$ reduces to the computation of the expected value of the measure over all subsets of the database of a given size m. In this case, $\mathbb{E}_{D' \sim U_m(D \setminus \{f\})}\big(\mathcal{I}_{\mathsf{d}}(D' \cup \{f\}, \Delta)\big)$ and $\mathbb{E}_{D' \sim U_m(D \setminus \{f\})}\big(\mathcal{I}_{\mathsf{d}}(D', \Delta)\big)$ are the probabilities that a uniformly chosen $D' \subseteq D \setminus \{f\}$ of size m is such that $(D' \cup \{f\}) \not\models \Delta$ and $D' \not\models \Delta$, respectively. Due to the structure of FD sets with an lhs chain, we can compute these probabilities efficiently, as we explain next.

Our main observation is that for an FD $X \to Y$, if we group the facts of D by X (i.e., split D into maximal subsets of facts that agree on the values of all attributes in X), then this FD and the FDs that appear later in the chain may be violated only among facts from the same group. Moreover, when we group by XY (i.e., further split each group of X into maximal subsets of facts that agree on the values of all attributes in Y), facts from different groups always violate this FD, and hence, violate Δ. We refer to the former groups as *blocks* and the latter groups as *subblocks*. This special structure allows us to split the problem into smaller problems, solve each one of them separately, and then combine the solutions via dynamic programming.

We define a data structure T where each vertex v is associated with a subset of D that we denote by $D[v]$. The root r is associated with D itself, that is, $D[r] = D$. At the first level, each child c of r is associated with a block of $D[r]$ w.r.t. $X_1 \to Y_1$, and each child c' of c is

[2] Recall that $\mathrm{FP}^{\#\mathrm{P}}$ is the class of polynomial-time functions with an oracle to a problem in #P (e.g., count the satisfying assignments of a propositional formula).

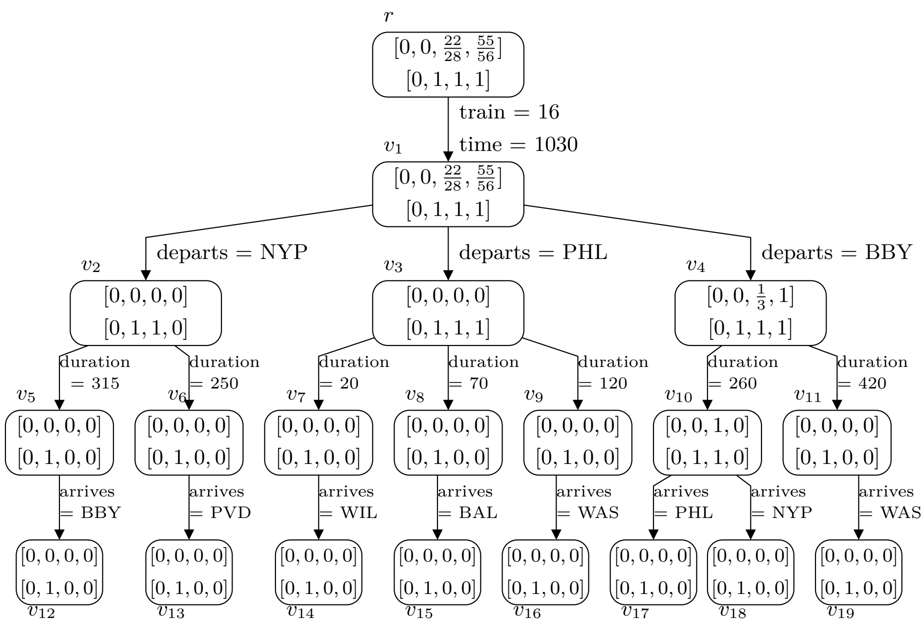

Figure 2 The data structure T of our running example.

associated with a subblock of $D[c]$ w.r.t. $X_1 \to Y_1$. At the second level, each child c'' of c' is associated with a block of $D[c']$ w.r.t. $X_2 \to Y_2$, and each child c''' of c'' is associated with a subblock of $D[c'']$ w.r.t. $X_2 \to Y_2$. This continues all the way to the nth FD, where at the ith level, each child u of an $(i-1)$th level subblock vertex v is associated with a block of $D[v]$ w.r.t. $X_i \to Y_i$ and each child u' of u is associated with a subblock of $D[u]$ w.r.t. $X_i \to Y_i$.

We assume that the data structure T is constructed in a preprocessing phase. Clearly, the number of vertices in T is polynomial in $|D|$ and n (recall that n is the number of FDs in Δ) as the height of the tree is $2n$, and each level contains at most $|D|$ vertices; hence, this preprocessing phase requires polynomial time (even under combined complexity). Then, we compute both $\mathbb{E}_{D' \sim U_m(D\setminus\{f\})}\big(\mathcal{I}_{\mathsf{d}}(D', \Delta)\big)$ and $\mathbb{E}_{D' \sim U_m(D\setminus\{f\})}\big(\mathcal{I}_{\mathsf{d}}(D' \cup \{f\}, \Delta)\big)$ by going over the vertices of T from bottom to top, as we will explain later. Note that for the computation of these values, we construct T from the database $D \setminus \{f\}$. Figure 2 depicts the data structure T used for the computation of $\mathrm{Shapley}(D, f_9, \Delta, \mathcal{I}_{\mathsf{d}})$ for the database D and fact f_9 of our running example. Next, we explain the meaning of the values stored in each vertex.

Each vertex v in T stores an array $v.\mathsf{val}$ with $|D[v]| + 1$ entries (that is initialized with zeros) such that $v.\mathsf{val}[j] = \mathbb{E}_{D' \sim U_j(D[v])}\big(\mathcal{I}_{\mathsf{d}}(D', \Delta)\big)$ for all $j \in \{0, \dots, |D[v]|\}$ at the end of the execution. For this measure, we have that:

$$v.\mathsf{val}[j] \overset{\text{def}}{=} \Pr\left[\text{a random subset of size } j \text{ of } D[v] \text{ violates } \Delta\right]$$

Our final goal is to compute $r.\mathsf{val}[m]$, where r is the root of T. For that purpose, in the algorithm DrasticShapley, depicted in Figure 3, we go over the vertices of T in a bottom-up order and compute the values of $v.\mathsf{val}$ for every vertex v in the UpdateProb subroutine. Observe that we only need one execution of DrasticShapley with $m = |D| - 1$ to compute the required values for all $m \in \{1, \dots, |D| - 1\}$, as we calculate all these values in our intermediate computations.

To compute $v.\mathsf{val}$ for a subblock vertex v, we iterate over its children in T (which are the $(i+1)$th level blocks) according to an arbitrary order defined in the construction of T. For a child c of v, we denote by $\mathsf{prev}(c)$ the set of children of v that occur before c in that order,

Algorithm 1 DrasticShapley(D, Δ, m, T).

```
1: for all vertices $v$ of $T$ in a bottom-up order do
2:     UpdateProb$(v, m)$
3: return $r.\mathsf{val}[m]$
```

Subroutine 1 UpdateProb(v, m).

```
1: for all children $c$ of $v$ in $T$ do
2:     for $j \in \{m, \dots, 1\}$ do
3:         $v.\mathsf{val}[j] = \sum_{\substack{j_1+j_2=j \\ 0\leq j_1\leq|D[c]| \\ 0\leq j_2\leq|D[\mathsf{prev}(c)]|}} \big(c.\mathsf{val}[j_1] + (1 - c.\mathsf{val}[j_1]) \cdot v.\mathsf{val}[j_2]\big) \cdot \frac{\binom{|D[c]|}{j_1}\cdot\binom{|D[\mathsf{prev}(c)]|}{j_2}}{\binom{|D[\mathsf{prev}(c)]|+|D[c]|}{j}}$
4:         if $v$ is a block node then
5:             $v.\mathsf{val}[j] \mathrel{+}= \sum_{\substack{j_1+j_2=j \\ 0<j_1\leq|D[c]| \\ 0<j_2\leq|D[\mathsf{prev}(c)]|}} \big((1 - c.\mathsf{val}[j_1]) \cdot (1 - v.\mathsf{val}[j_2])\big) \cdot \frac{\binom{|D[c]|}{j_1}\cdot\binom{|D[\mathsf{prev}(c)]|}{j_2}}{\binom{|D[\mathsf{prev}(c)]|+|D[c]|}{j}}$
```

Figure 3 An algorithm for computing $\mathbb{E}_{D'\sim U_m(D\setminus\{f\})}\big(\mathcal{I}_{\mathsf{d}}(D', \Delta)\big)$ for Δ with an lhs chain.

and by $D[\mathsf{prev}(c)]$ the database $\bigcup_{c'\in\mathsf{prev}(c)} D[c']$. When considering c in the for loop of line 1, we compute the expected value of the measure on a subset of $D[\mathsf{prev}(c)] \cup D[c]$. Hence, when we consider the last child of v in the for loop of line 1, we compute the expected value of the measure on a subset of the entire database $D[v]$.

For a child c of v, there are $N_1 = \binom{|D[\mathsf{prev}(c)]|+|D[c]|}{j}$ subsets of size j of all the children of v considered so far (including c itself). Each such subset consists of j_1 facts of the current c (there are $N_2 = \binom{|D[c]|}{j_1}$ possibilities) and j_2 facts of the previously considered children (there are $N_3 = \binom{|D[\mathsf{prev}(c)]|}{j_2}$ possibilities), for some j_1, j_2 such that $j_1 + j_2 = j$, with probability N_2N_3/N_1. Moreover, such a subset violates Δ if either the facts of the current c violate Δ (with probability $c.\mathsf{val}[j_1]$ that was computed in a previous iteration) or these facts satisfy Δ, but the facts of the previous children violate Δ (with probability $(1 - c.\mathsf{val}[j_1]) \cdot v.\mathsf{val}[j_2]$). Observe that since we go over the values j in reverse order in the for loop of line 2 (i.e., from m to 1), at each iteration of this loop, we have that $v.\mathsf{val}[j_2]$ (for all considered $j_2 \leq j$) still holds the expected value of $\mathcal{I}_{\mathsf{d}}$ over subsets of size j_2 of the previous children of v, which is indeed the value that we need for our computation.

This computation of $v.\mathsf{val}$ also applies to the block vertices. However, the addition of line 5 only applies to blocks. Since the children of a block belong to different subblocks, and two facts from the same ith level block but different ith level subbblocks always jointly violate $X_i \to Y_i$, a subset of size j of a block also violates the constraints if we select a non-empty subset of the current child c and a non-empty subset of the previous children, even if each of these subsets by itself is consistent w.r.t. Δ. Hence, we add this probability in line 5. Note that all the three cases that we consider are disjoint, so we sum the probabilities. Observe also that the leaves of T have no children and we do not update their probabilities, and, indeed the probability to select a subset from a leaf v that violates the constraints is zero, as all the facts of $D[v]$ agree on the values of all the attributes that occur in Δ.

To compute $\mathbb{E}_{D'\sim U_m(D\setminus\{f\})}\big(\mathcal{I}_{\mathsf{d}}(D'\cup\{f\}, \Delta)\big)$, we use the algorithm DrasticShapleyF, given in [26]. There, we distinguish between several types of vertices w.r.t. f, and show how this expectation can be computed for each one of these types.

▶ **Example 9.** We now illustrate the computation of $\mathbb{E}_{D'\sim U_m(D\setminus\{f_9\})}\big(\mathcal{I}_{\mathsf{d}}(D',\Delta)\big)$ on the database D and the fact f_9 of our running example for $m=3$. Inside each node of the data structure T of Figure 2, we show the values $[v.\mathsf{val}[0], v.\mathsf{val}[1], v.\mathsf{val}[2], v.\mathsf{val}[3]]$ used for this computation. Below them, we present the corresponding values used in the computation of $\mathbb{E}_{D'\sim U_m(D\setminus\{f_9\})}\big(\mathcal{I}_{\mathsf{d}}(D'\cup\{f\},\Delta)\big)$. For the leaves v and each vertex $v\in\{v_5,\dots,v_9,v_{11}\}$, we have that $v.\mathsf{val}[j]=0$ for every $j\in\{0,1,2,3\}$, as $D[v]$ has a single fact. As for v_{10}, when we consider its first child v_{17} in the for loop of line 1 of UpdateProb, all the values in $v_{10}.\mathsf{val}$ remain zero (since $v_{17}.\mathsf{val}[j_1]=v_{10}.\mathsf{val}[j_2]=0$ for any j_1,j_2, and $|D[\mathsf{prev}(c)]|=0$). However, when we consider its second child v_{18}, while the computation of line 3 again has no impact on $v_{10}.\mathsf{val}$, after the computation of line 5 we have that $v_{10}.\mathsf{val}[2]=1$. And, indeed, there is a single subset of size two of $D[v_{10}]$, which is $\{f_7,f_8\}$, and it violates the FD train time duration $\rightarrow$ arrives. This also affects the values of $v_4.\mathsf{val}$. In particular, when we consider the first child v_{10} of v_4, we have that $v_4.\mathsf{val}[j]=1$ for $j=2$ and $v_4.\mathsf{val}[j]=0$ for any other j. Then, when we consider the second child v_{11} of v_4, it holds that $v_4.\mathsf{val}[2]=\frac{1}{3}$ (as the only subset of size two of $D[v_4]$ that violates the FDs is $\{f_6,f_7\}$, and there are three subsets in total) and $v_4.\mathsf{val}[3]=1$ (as every subset of size three contains both f_6 and f_7). Finally, we have that $\mathbb{E}_{D'\sim U_3(D\setminus\{f_9\})}\big(\mathcal{I}_{\mathsf{d}}(D',\Delta)\big)=\frac{55}{56}$ and $\mathbb{E}_{D'\sim U_3(D\setminus\{f_9\})}\big(\mathcal{I}_{\mathsf{d}}(D'\cup\{f_9\},\Delta)\big)=1$. ◊

Approximation. We now consider an approximate computation of the Shapley value. Using the Chernoff-Hoeffding bound, we can easily obtain an additive FPRAS of the value $\mathrm{Shapley}(D,f,\Delta,\mathcal{I}_{\mathsf{d}})$, by sampling $O(\log(1/\delta)/\epsilon^2)$ permutations and computing the average contribution of f in a permutation. As observed by Livshits et al. [23], a multiplicative FPRAS can be obtained using the same algorithm (possibly with a different number of samples) if the "gap" property holds: nonzero Shapley values are guaranteed to be large enough compared to the utility value (which is at most 1 in the case of the drastic measure). This is indeed the case here, as we now prove the following gap property of $\mathrm{Shapley}(D,f,\Delta,\mathcal{I}_{\mathsf{d}})$.

▶ **Proposition 10.** *There is a polynomial p such that for all databases D and facts f of D the value $\mathrm{Shapley}(D,f,\Delta,\mathcal{I}_{\mathsf{d}})$ is either zero or at least $1/(p(|D|))$.*

Proof. If no fact of D is in conflict with f, then $\mathrm{Shapley}(D,f,\Delta,\mathcal{I}_{\mathsf{d}})=0$. Otherwise, let g be a fact that violates an FD of Δ jointly with f. Clearly, it holds that $\{g\}\models\Delta$, while $\{g,f\}\not\models\Delta$. The probability to choose a permutation σ, such that σ_f is exactly $\{g\}$ is $\frac{(|D|-2)!}{|D|!}=\frac{1}{|D|\cdot(|D|-1)}$ (recall that σ_f is the set of facts that appear before f in σ). Therefore, we have that $\mathrm{Shapley}(D,f,\Delta,\mathcal{I}_{\mathsf{d}})\geq\frac{1}{|D|\cdot(|D|-1)}$, and that concludes our proof. ◀

We conclude that the following holds.

▶ **Corollary 11.** $\mathrm{Shapley}(D,f,\Delta,\mathcal{I}_{\mathsf{d}})$ *has both an additive and a multiplicative FPRAS.*

6 The Cost of a Cardinality Repair $\mathcal{I}_{\mathsf{R}}$

In this section, we study the measure $\mathcal{I}_{\mathsf{R}}$, based on the cost of a *cardinality repair* of D. Here, we refer to the number of facts that are removed from D to obtain a repair E as the cost of E. This is the first measure that we consider that is not always computable in polynomial time. Livshits et al. [27] established a dichotomy for the problem of computing a cardinality repair, classifying FD sets into those for which the problem is solvable in polynomial time, and those for which it is NP-hard. They presented a polynomial-time algorithm, which we refer to as Simplify, that takes as input an FD set Δ, finds a *removable* pair (X,Y) of attribute sets (if such a pair exists), and removes every attribute of $X\cup Y$ from every FD in

Algorithm 2 Simplify(Δ).

```
1: Remove trivial FDs from $\Delta$
2: if $\Delta$ is not empty then
3:     find a removable pair $(X, Y)$ of attribute sets
4:     $\Delta := \Delta - XY$
   return $\Delta$
```

Figure 4 A simplification algorithm used for deciding whether a cardinality repair w.r.t. Δ can be computed in polynomial time [27].

Δ (we denote the result by $\Delta - XY$). A pair (X, Y) of attribute sets is considered removable if it satisfies the following three conditions:

- $\text{Closure}_\Delta(X) = \text{Closure}_\Delta(Y)$,
- XY is nonempty,
- every FD in Δ contains either X or Y on the left-hand side.

Note that it may be the case that $X = Y$, and then the conditions imply that every FD of Δ contains X on the left-hand side. The algorithm is depicted in Figure 4.

Livshits et al. [27] have shown that if it is possible to transform Δ to an empty set by repeatedly applying Simplify(Δ), then a cardinality repair can be computed in polynomial time. Otherwise, the problem is NP-hard (and, in fact, APX-complete).

Fact 1 implies that computing $\text{Shapley}(D, f, \Delta, \mathcal{I}_\mathsf{R})$ is hard whenever computing $\mathcal{I}_\mathsf{R}(D, \Delta)$ is hard. Hence, we immediately obtain the following.

▶ **Theorem 12.** *Let Δ be a set of FDs. If Δ cannot be emptied by repeatedly applying* Simplify(Δ)*, then computing* $\text{Shapley}(D, f, \Delta, \mathcal{I}_\mathsf{R})$ *is NP-hard.*

In the remainder of this section, we focus on the tractable cases of the dichotomy of Livshits et al. [27]. In particular, we start by proving that the Shapley value can again be computed in polynomial time for an FD set that has an lhs chain.

▶ **Theorem 13.** *Let Δ be a set of FDs. If Δ is equivalent to an FD set with an lhs chain, then computing* $\text{Shapley}(D, f, \Delta, \mathcal{I}_\mathsf{R})$ *can be done in polynomial time, given D and f.*

Our polynomial-time algorithm RShapley, depicted in Figure 5, is very similar in structure to DrasticShapley. However, to compute the expected value of $\mathcal{I}_\mathsf{R}$, we take the reduction of Proposition 3 a step further, and show, that the problem of computing the expected value of the measure over subsets of size m can be reduced to the problem of computing the number of subsets of size m of D that have a cardinality repair of cost k, given m and k. In the subroutine UpdateCount, we compute this number.

For each vertex v in T, we define:

$$v.\mathsf{val}[j, t] \overset{\text{def}}{=} \text{number of subsets of size } j \text{ of } D[v] \text{ with a cardinality repair of cost } t$$

For the leaves v of T, we set $v.\mathsf{val}[j, 0] = \binom{|D[v]|}{j}$ for $0 \leq j \leq |D[v]|$, as every subset of $D[v]$ is consistent, and the cost of a cardinality repair is zero. We also set $v.\mathsf{val}[0, 0] = 1$ for each v in T for the same reason. Since the size of the cardinality repair is bounded by the size of the database, in UpdateCount(v, m), we compute the value $v.\mathsf{val}[j, t]$ for every $1 \leq j \leq m$ and $0 \leq t \leq j$. To compute this number, we again go over the children of v, one by one. When we consider a child c in the for loop of line 1, the value $v.\mathsf{val}[j, t]$ is the number of subsets of size j of $D[\mathsf{prev}(c)] \cup D[c]$ that have a cardinality repair of cost t.

Algorithm 3 $\mathsf{RShapley}(D, \Delta, m, T)$.

```
1: for all vertices $v$ of $T$ in a bottom-up order do
2:     $\mathsf{UpdateCount}(v, m)$
3: return $\sum_{k=0}^{m} \frac{k}{\binom{|D|-1}{m}} \cdot r.\mathsf{val}[m, k]$
```

Subroutine 2 $\mathsf{UpdateCount}(v, m)$.

```
1: $v.\mathsf{val}[0, 0] = 1$
2: if $v$ is a leaf then $v.\mathsf{val}[j, 0] = \binom{|D[v]|}{j}$ for all $j \in \{1, \dots, |D[v]|\}$
3: for all children $c$ of $v$ in $T$ do
4:     for $j \in \{m, \dots, 1\}$ do
5:         for $t \in \{j, \dots, 0\}$ do
6:             if $v$ is a block vertex then
7:                 $v.\mathsf{val}[j, t] = \sum_{\substack{j_1+j_2=j \\ 0 \le j_1 \le |D[c]| \\ t-j_1 \le j_2 \le \min\{t, |D[\mathsf{prev}(c)]|\}}} \sum_{t-j_1 \le w_2 \le j_2} \big(c.\mathsf{val}[j_1, t-j_2] \cdot v.\mathsf{val}[j_2, w_2]\big)$
8:                 $v.\mathsf{val}[j, t] \mathrel{+}= \sum_{\substack{j_1+j_2=j \\ t-j_2 \le j_1 \le \min\{t, |D[c]|\} \\ 0 \le j_2 \le |D[\mathsf{prev}(c)]|}} \sum_{t-j_2 < w_1 \le j_1} \big(c.\mathsf{val}[j_1, w_1] \cdot v.\mathsf{val}[j_2, t-j_1]\big)$
9:             else
10:                $v.\mathsf{val}[j, t] = \sum_{\substack{j_1+j_2=j \\ 0 \le j_1 \le |D[c]| \\ 0 \le j_2 \le |D[\mathsf{prev}(c)]|}} \sum_{\substack{t_1+t_2=t \\ 0 \le t_1 \le j_1 \\ 0 \le t_2 \le j_2}} \big(c.\mathsf{val}[j_1, t_1] \cdot v.\mathsf{val}[j_2, t_2]\big)$
```

Figure 5 An algorithm for computing $\mathbb{E}_{D' \sim U_m(D \setminus \{f\})}\big(\mathcal{I}_{\mathsf{R}}(D', \Delta)\big)$ for Δ with an lhs chain.

The children of a block v are subblocks that jointly violate an FD of Δ; hence, when we consider a child c of v, a cardinality repair of a subset E of $D[\mathsf{prev}(c)] \cup D[c]$ is either a cardinality repair of $E \cap D[c]$ (in which case we remove every fact of $E \cap D[\mathsf{prev}(c)]$) or a cardinality repair of $E \cap D[\mathsf{prev}(c)]$ (in which case we remove every fact of $E \cap D[c]$). The decision regarding which of these cases holds is based on the following four parameters: *(1)* the number j_1 of facts in $E \cap D[c]$, *(2)* the number j_2 of facts in $E \cap D[\mathsf{prev}(c)]$, *(3)* the cost w_1 of a cardinality repair of $E \cap D[c]$, and *(4)* the cost w_2 of a cardinality repair of $E \cap D[\mathsf{prev}(c)]$. In particular:

- If $w_1 + j_2 \le w_2 + j_1$, then a cardinality repair of $E \cap D[c]$ is preferred over a cardinality repair of $E \cap D[\mathsf{prev}(c)]$, as it requires removing less facts from the database.
- If $w_1 + j_2 > w_2 + j_1$, then a cardinality repair of $E \cap D[\mathsf{prev}(c)]$ is preferred over a cardinality repair of $E \cap D[c]$.

In fact, since we fix t in the computation of $v.\mathsf{val}[j, t]$, we do not need to go over all w_1 and w_2. In the first case, we have that $w_1 = t - j_2$ (hence, the total number of removed facts is $t - j_2 + j_2 = t$), and in the second case we have that $w_2 = t - j_1$ for the same reason. Hence, in line 7 we consider the first case where $t \le w_2 + j_1$, and in line 8 we consider the second case where $w_1 + j_2 > t$. To avoid negative costs, we add a lower bound of $t - j_1$ on j_2 and w_2 in line 7, and, similarly, a lower bound of $t - j_2$ on j_1 and w_1 in line 8.

For a subblock vertex v, a cardinality repair of $D[v]$ is the union of cardinality repairs of the children of v, as facts corresponding to different children of v do not jointly violate any FD. Therefore, for such vertices, in line 10, we compute $v.\mathsf{val}$ by going over all j_1, j_2 such that $j_1 + j_2 = j$ and all t_1, t_2 such that $t_1 + t_2 = t$ and multiply the number of subsets of size j_1 of the current child for which the cost of a cardinality repair is t_1 by the number of subsets of size j_2 of the previously considered children for which the cost of a cardinality repair is t_2.

We present a polynomial-time algorithm for computing $\mathbb{E}_{D' \sim U_m(D \setminus \{f\})}\big(\mathcal{I}_{\mathsf{R}}(D' \cup \{f\}, \Delta)\big)$ in [26].

Approximation. In cases where a cardinality repair can be computed in polynomial time, we can obtain an additive FPRAS in the same way as the drastic measure. (Note that this Shapley value is also in $[0,1]$.) Moreover, we can again obtain a multiplicative FPRAS using the same technique due to the following gap property (proved similarly to Proposition 10).

▶ **Proposition 14.** *There is a polynomial p such that for all databases D and facts f of D the value* $\text{Shapley}(D, f, \Delta, \mathcal{I}_{\mathsf{R}})$ *is either zero or at least* $1/(p(|D|))$.

As aforementioned, Livshits et al. [27] showed that the hard cases of their dichotomy for the problem of computing a cardinality repair are, in fact, APX-complete; hence, there is a polynomial-time constant-ratio approximation, but for some $\epsilon > 1$ there is no (randomized) ϵ-approximation or else P = NP (NP $\subseteq$ BPP). Since the Shapley value of every fact w.r.t. $\mathcal{I}_{\mathsf{R}}$ is positive, the existence of a multiplicative FPRAS for $\text{Shapley}(D, f, \Delta, \mathcal{I}_{\mathsf{R}})$ would imply the existence of a multiplicative FPRAS for $\mathcal{I}_{\mathsf{R}}(D, \Delta)$ (due to Fact 1), which is a contradiction to the APX-hardness. We conclude the following.

▶ **Proposition 15.** *Let Δ be a set of FDs. If Δ can be emptied by repeatedly applying* $\mathsf{Simplify}(\Delta)$*, then* $\text{Shapley}(D, f, \Delta, \mathcal{I}_{\mathsf{R}})$ *has both an additive and a multiplicative FPRAS. Otherwise, it has neither multiplicative nor additive FPRAS, unless NP $\subseteq$ BPP.*

Unsolved cases for $\mathcal{I}_{\mathsf{R}}$. Unlike the other inconsistency measures considered in this paper, we do not have a full dichotomy for the measure $\mathcal{I}_{\mathsf{R}}$. In particular, a basic open problem is the computation of $\text{Shapley}(D, f, \Delta, \mathcal{I}_{\mathsf{R}})$ for $\Delta = \{A \to B, B \to A\}$. On the one hand, Proposition 15 shows that this case belongs to the tractable side if an approximation is allowed. On the other hand, our algorithm for exact $\text{Shapley}(D, f, \Delta, \mathcal{I}_{\mathsf{R}})$ is via counting the subsets of size m that have a cardinality repair of cost k. This approach will not work here:

▶ **Proposition 16.** *Let $\Delta = \{A \to B, B \to A\}$ be an FD set over (A, B). Counting the subsets of size m of a given database that have a cardinality repair of cost k is #P-hard.*

Proof. The proof is by a reduction from the problem of computing the number of perfect matchings in a bipartite graph, known to be #P-complete [38]. Given a bipartite graph $g = (A \cup B, E)$ (where $|A| = |B|$), we construct a database D over (A, B) by adding a fact (a, b) for every edge $(a, b) \in E$. We then define $m = |A|$ and $k = 0$. It is rather straightforward that the perfect matchings of g correspond exactly to the subsets D' of size $|A|$ of D such that D' itself is a cardinality repair. ◀

Observe that the cooperative game for $\Delta = \{A \to B, B \to A\}$ can be seen as a game on bipartite graphs where the vertices on the left-hand side represent the values of attribute A, the vertices on the right-hand side correspond to the values that occur in attribute B, and the edges represent the tuples of the database (hence, the players of the game). This game is different from the well-known matching game [1] where the players are the *vertices* of the graph (and the value of the game is determined by the maximum weight matching of the subgraph induced by the coalition). In contrast, in our case the players correspond to the *edges* of the graph. It is not clear what is the connection between the two games and whether or how we can use known results on matching games to derive results for the game that we consider here.

Algorithm 4 MCShapley(D, Δ, m, T).

```
1: for all vertices $v$ of $T$ in a bottom-up order do
2:     UpdateExpected$(v, m)$
3: return $r.\mathsf{val}[m]$
```

Subroutine 3 UpdateExpected(v, m).

```
1: $v.\mathsf{val}[0] = 1$
2: if $v$ is a leaf then $v.\mathsf{val}[j] = 1$ for all $j \in \{1, \dots, |D[v]|\}$
3: for all children $c$ of $v$ in $T$ do
4:     for $j \in \{m, \dots, 1\}$ do
5:         if $v$ is a block vertex then
6:             $v.\mathsf{val}[j] = \sum_{\substack{j_1+j_2=j \\ 0\le j_1\le |D[c]| \\ 0\le j_2\le |D[\mathsf{prev}(c)]|}} \big(c.\mathsf{val}[j_1] + v.\mathsf{val}[j_2]\big)$
7:         else
8:             $v.\mathsf{val}[j] = \sum_{\substack{j_1+j_2=j \\ 0\le j_1\le |D[c]| \\ 0\le j_2\le |D[\mathsf{prev}(c)]|}} \big(c.\mathsf{val}[j_1] \cdot v.\mathsf{val}[j_2]\big)$
```

Figure 6 An algorithm for computing $\mathbb{E}_{D'\sim U_m(D\setminus\{f\})}\big(\mathcal{I}_{\mathsf{MC}}(D', \Delta)\big)$ for Δ with an lhs chain.

7 The Number of Repairs $\mathcal{I}_{\mathsf{MC}}$

The final measure that we consider counts the repairs of the database. A dichotomy result from our previous work [25] states that the problem of counting repairs can be solved in polynomial time for FD sets with an lhs chain (up to equivalence), and is #P-complete for any other FD set. The hardness side, along with Fact 1, implies that computing Shapley$(D, f, \Delta, \mathcal{I}_{\mathsf{MC}})$ is FP$^{\#\mathrm{P}}$-hard whenever the FD set is not equivalent to an FD set with an lhs chain. Hence, an lhs chain is a necessary condition for tractability. We show here that it is also sufficient: if the FD set has an lhs chain, then the problem can be solved in polynomial time. Consequently, we obtain the following dichotomy.

▶ **Theorem 17.** *Let Δ be a set of FDs. If Δ is equivalent to an FD set with an lhs chain, then computing* Shapley$(D, f, \Delta, \mathcal{I}_{\mathsf{MC}})$ *can be done in polynomial time, given D and f. Otherwise, the problem is* FP$^{\#\mathrm{P}}$*-complete.*

The algorithm MCShapley, depicted in Figure 6, for computing Shapley$(D, f, \Delta, \mathcal{I}_{\mathsf{MC}})$, has the same structure as DrasticShapley, with the only difference being the computations in the subroutine UpdateExpected (that replaces UpdateProb).

For a vertex v in T we define:

$$v.\mathsf{val}[j] = \mathbb{E}\,[\text{number of repairs of a random subset of size } j \text{ of } D[v]]$$

As the number of repairs of a consistent database D is one (D itself is a repair), we set $v.\mathsf{val}[0] = 1$ for every vertex v and $v.\mathsf{val}[j] = 1$ for $0 \le j \le |D[v]|$ for every leaf v. Now, consider a block vertex v and a child c of v. Since the children of v are subblocks, each repair consists of facts of a single child. Hence, the total number of repairs is the sum of repairs of the children of v. Moreover, since our choice of facts from different subblocks is independent, we have the following (where $\mathsf{MC}(D,\Delta)$ is the set of repairs of D w.r.t. Δ).

$$\mathbb{E}_{D'\sim U_j(D[\mathsf{prev}(c)]\cup D[c])}\big(\mathcal{I}_{\mathsf{MC}}(D',\Delta)\big) = \sum_{\substack{D'\subseteq D[\mathsf{prev}(c)]\cup D[c]\\ |D'|=j}} \Pr[D']\cdot|MC(D',\Delta)| =$$
$$\sum_{\substack{j_1+j_2=j\\ 0\le j_1\le |D[c]|\\ 0\le j_2\le D[\mathsf{prev}(c)]}} \sum_{\substack{E_1\subseteq D[c]\\ |E_1|=j_1}} \sum_{\substack{E_2\subseteq D[\mathsf{prev}(c)]\\ |E_2|=j_2}} \Pr[E_1]\Pr[E_2]\big(|MC(E_1,\Delta)|+|MC(E_2,\Delta)|\big)$$

Using standard mathematical manipulations, we obtain the following result:

$$\mathbb{E}_{D'\sim U_j(D[\mathsf{prev}(c)]\cup D[c])}\big(\mathcal{I}_{\mathsf{MC}}(D',\Delta)\big) =$$
$$\sum_{\substack{j_1+j_2=j\\ 0\le j_1\le |D[c]|\\ 0\le j_2\le D[\mathsf{prev}(c)]}} \left[\mathbb{E}_{D'\sim U_{j_1}(D[c])}\big(\mathcal{I}_{\mathsf{MC}}(D',\Delta)\big) + \mathbb{E}_{D'\sim U_{j_2}(D[\mathsf{prev}(c)]}\big(\mathcal{I}_{\mathsf{MC}}(D',\Delta)\big)\right]$$

This calculation is reflected in line 6 of the UpdateExpected subroutine.

We can similarly show that for a subblock vertex v, it holds that:

$$\mathbb{E}_{D'\sim U_j(D[\mathsf{prev}(c)]\cup D[c])}\big(\mathcal{I}_{\mathsf{MC}}(D',\Delta)\big) =$$
$$\sum_{\substack{j_1+j_2=j\\ 0\le j_1\le |D[c]|\\ 0\le j_2\le D[\mathsf{prev}(c)]}} \left[\mathbb{E}_{D'\sim U_{j_1}(D[c])}\big(\mathcal{I}_{\mathsf{MC}}(D',\Delta)\big) \times \mathbb{E}_{D'\sim U_{j_2}(D[\mathsf{prev}(c)]}\big(\mathcal{I}_{\mathsf{MC}}(D',\Delta)\big)\right]$$

We use this result for the calculation of line 8. The difference between the calculations lies in the observation that the children of a subblock are blocks that never jointly violate Δ; hence, the number of repairs is obtained by multiplying the number of repairs of each child of v.

An algorithm for computing $\mathbb{E}_{D'\sim U_m(D\setminus\{f\})}\big(\mathcal{I}_{\mathsf{MC}}(D'\cup\{f\},\Delta)\big)$ is given in [26].

Approximation. Repair counting for $\Delta=\{A\to B, B\to A\}$ is the problem of counting the maximal matchings of a bipartite graph. As the values Shapley$(D,f,\Delta,\mathcal{I}_{\mathsf{MC}})$ are nonnegative and sum up to the number of repairs, we conclude that an FPRAS for Shapley implies an FPRAS for the number of maximal matchings. To the best of our knowledge, existence of the latter is a long-standing open problem [18]. This is also the case for any Δ' that is not equivalent to an FD set with an lhs chain, since there is a fact-wise reduction from Δ to such Δ' [25].

8 Conclusions

We studied the complexity of calculating the Shapley value of database facts for basic inconsistency measures, focusing on FD constraints. We showed that two of them are computable in polynomial time: the number of violations ($\mathcal{I}_{\mathsf{MI}}$) and the number of problematic facts ($\mathcal{I}_{\mathsf{P}}$). In contrast, each of the drastic measure ($\mathcal{I}_{\mathsf{d}}$) and the number of repairs ($\mathcal{I}_{\mathsf{MC}}$) features a dichotomy in complexity, where the tractability condition is the possession of an lhs chain (up to equivalence). For the cost of a cardinality repair ($\mathcal{I}_{\mathsf{R}}$) we showed a tractable fragment and an intractable fragment, but a gap remains on certain FD sets – the ones that do not have an lhs chain, and yet, a cardinality repair can be computed in polynomial time. We also studied the approximability of the Shapley value and showed, among other things, an FPRAS for $\mathcal{I}_{\mathsf{d}}$ and a dichotomy in the existence of an FPRAS for $\mathcal{I}_{\mathsf{R}}$.

This work has been restricted to schemas with a single relation. Some of the results immediately generalize to schemas with multiple relations. For instance, it is easy to show that all of our lower bounds generalize. It is also easy to show that our upper bounds generalize when considering the additive inconsistency measures $\mathcal{I}_{\mathsf{MI}}$, $\mathcal{I}_{\mathsf{P}}$ and $\mathcal{I}_{\mathsf{R}}$ (where the value of the measure on the entire database is the sum of the values over the individual relations of the schema). This is due to the linearity property of the Shapley value [35]: $\mathrm{Shapley}(D, f, \Delta, a \cdot \alpha + b \cdot \beta) = a \cdot \mathrm{Shapley}(D, f, \Delta, \alpha) + b \cdot \mathrm{Shapley}(D, f, \Delta, \beta)$. We believe that our upper bounds can also be generalized to multiple relation schemas for the measures $\mathcal{I}_{\mathsf{d}}$ and $\mathcal{I}_{\mathsf{MC}}$, but this requires a more subtle proof that we leave for future work.

Many other directions are left open for future research. First, the picture is incomplete for the measure $\mathcal{I}_{\mathsf{R}}$. In particular, the complexity of the exact computation is open for the bipartite matching constraint $\{A \rightarrow B, B \rightarrow A\}$ that, unlike the known FD sets in the intractable fragment, has an FPRAS. In general, we would like to complete the picture of $\mathcal{I}_{\mathsf{R}}$ towards a full dichotomy. Moreover, for the schemas where there is no FPRAS for $\mathcal{I}_{\mathsf{R}}$, our results neither imply nor refute the existence of a constant-ratio approximation (for *some* constant). Second, the problems are immediately extendible to any type of constraints other than functional dependencies, such as denial constraints, tuple generating dependencies, and so on. Third, it would be interesting to see how the results extend to wealth distribution functions other than Shapley, e.g., the Banzhaff Power Index [7]. The tractable cases remain tractable for the Banzhaff Power Index, but it is not clear how (and whether) our proofs for the lower bounds generalize to this function. Finally, there is the practical question of implementation: while our algorithms terminate in polynomial time, we believe that they are hardly scalable without further optimization and heuristics ad-hoc to the use case; developing those is an important challenge for future research.

References

1 Haris Aziz and Bart de Keijzer. Shapley meets Shapley. In *STACS*, volume 25 of *LIPIcs*, pages 99–111. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2014.

2 Leopoldo E. Bertossi. Measuring and computing database inconsistency via repairs. In *SUM*, volume 11142 of *Lecture Notes in Computer Science*, pages 368–372. Springer, 2018.

3 Leopoldo E. Bertossi. Repair-based degrees of database inconsistency. In *LPNMR*, volume 11481 of *Lecture Notes in Computer Science*, pages 195–209. Springer, 2019.

4 Leopoldo E. Bertossi and Floris Geerts. Data quality and explainable AI. *J. Data and Information Quality*, 12(2):11:1–11:9, 2020.

5 Omar Besbes, Antoine Désir, Vineet Goyal, Garud Iyengar, and Raghav Singal. Shapley meets uniform: An axiomatic framework for attribution in online advertising. In *WWW*, pages 1713–1723. ACM, 2019.

6 Laurence Cholvy, Laurent Perrussel, William Raynaut, and Jean - Marc Th ’e venin. Towards consistency-based reliability assessment. In *AAMAS*, pages 1643–1644. ACM, 2015.

7 Pradeep Dubey and Lloyd S. Shapley. Mathematical properties of the banzhaf power index. *Mathematics of Operations Research*, 4(2):99–131, 1979.

8 Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *J. ACM*, 45(4):653–750, 1998. `doi:10.1145/285055.285060`.

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

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

11 John Grant and Anthony Hunter. Distance-based measures of inconsistency. In *ECSQARU*, volume 7958 of *Lecture Notes in Computer Science*, pages 230–241. Springer, 2013.

12 John Grant and Anthony Hunter. Using Shapley inconsistency values for distributed information systems with uncertainty. In *ECSQARU*, volume 9161 of *Lecture Notes in Computer Science*, pages 235–245. Springer, 2015.

13 John Grant and Anthony Hunter. Analysing inconsistent information using distance-based measures. *Int. J. Approx. Reasoning*, 89:3–26, 2017. `doi:10.1016/j.ijar.2016.04.004`.

14 Faruk Gul. Bargaining foundations of Shapley value. *Econometrica: Journal of the Econometric Society*, pages 81–95, 1989.

15 Anthony Hunter and Sébastien Konieczny. Shapley inconsistency values. In *KR*, pages 249–259. AAAI Press, 2006.

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

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

18 Yifan Jing and Akbar Rafiey. Counting maximal near perfect matchings in quasirandom and dense graphs. *CoRR*, abs/1807.04803, 2018.

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

20 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.

21 Christophe Labreuche and Simon Fossier. Explaining multi-criteria decision aiding models with an extended Shapley value. In *IJCAI*, pages 331–339. ijcai.org, 2018.

22 Zhenliang Liao, Xiaolong Zhu, and Jiaorong Shi. Case study on initial allocation of shanghai carbon emission trading based on Shapley value. *Journal of Cleaner Production*, 103:338–344, 2015.

23 Ester Livshits, Leopoldo E. Bertossi, Benny Kimelfeld, and Moshe Sebag. The Shapley value of tuples in query answering. In *ICDT*, volume 155 of *LIPIcs*, pages 20: 1–20: 19. Schloss Dagstuhl, 2020.

24 Ester Livshits, Ihab F. Ilyas, Benny Kimelfeld, and Sudeepa Roy. Principles of progress indicators for database repairing. *CoRR*, abs/1904.06492, 2019.

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

26 Ester Livshits and Benny Kimelfeld. The Shapley value of inconsistency measures for functional dependencies. *CoRR*, abs/2009.13819, 2020.

27 Ester Livshits, Benny Kimelfeld, and Sudeepa Roy. Computing optimal repairs for functional dependencies. *ACM Trans. Database Syst.*, 45(1):4: 1–4: 46, 2020.

28 Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *NIPS*, pages 4765–4774, 2017.

29 Richard TB Ma, Dah Ming Chiu, John Lui, Vishal Misra, and Dan Rubenstein. Internet economics: The use of Shapley value for isp settlement. *IEEE/ACM Transactions on Networking (TON)*, 18(3):775–787, 2010.

30 Kedian Mu, Weiru Liu, and Zhi Jin. Measuring the blame of each formula for inconsistent prioritized knowledge bases. *Journal of Logic and Computation*, 22(3):481–516, February 2011. `arXiv:https://academic.oup.com/logcom/article-pdf/22/3/481/3177718/exr002.pdf`.

31 Ramasuri Narayanam and Yadati Narahari. A Shapley value-based approach to discover influential nodes in social networks. *IEEE Transactions on Automation Science and Engineering*, 8(1):130–147, 2011.

32 Tatiana Nenova. The value of corporate voting rights and control: A cross-country analysis. *Journal of financial economics*, 68(3):325–351, 2003.

33 Leon Petrosjan and Georges Zaccour. Time-consistent Shapley value allocation of pollution cost reduction. *Journal of economic dynamics and control*, 27(3):381–398, 2003.

34 Alon Reshef, Benny Kimelfeld, and Ester Livshits. The impact of negation on the complexity of the Shapley value in conjunctive queries. In *PODS*, pages 285–297. ACM, 2020.

35 Lloyd S Shapley. A value for n-person games. In Harold W. Kuhn and Albert W. Tucker, editors, *Contributions to the Theory of Games II*, pages 307–317. Princeton University Press, Princeton, 1953.

36 Matthias Thimm. Measuring inconsistency in probabilistic knowledge bases. In *UAI*, pages 530–537. AUAI Press, 2009.

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

38 L.G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8(2):189–201, 1979. doi:10.1016/0304-3975(79)90044-6.

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