24th European Conference on Artificial Intelligence - ECAI 2020
Santiago de Compostela, Spain

On Measuring Inconsistency in Relational Databases with Denial Constraints

Francesco Parisi [1] and **John Grant** [2]

Abstract. Real-world databases are often inconsistent. Although there has been an extensive body of work on handling inconsistency, little work has been done on measuring inconsistency in databases. In this paper, building on work done on measuring inconsistency in propositional knowledge bases, we explore inconsistency measures (IMs) for databases with denial constraints. We first introduce new database IMs that are inspired by well-established methods to quantify inconsistency in propositional knowledge bases, but are tailored to the relational database context where data are generally the reason for inconsistency, not the integrity constraints. Then, we analyze the compliance of the database IMs with rationality postulates, and investigate the complexity of the inconsistency measurement problem as well as of the problems of deciding whether the inconsistency is lower than, greater than, or equal to a given threshold.

1 Introduction

There is a growing number of applications where inconsistent information arises, often because data are obtained from multiple sources [32]. This has led to an extensive body of work on handling inconsistent data, and in particular inconsistent databases. Approaches for dealing with inconsistent databases include consistent query answering frameworks [1, 7], inconsistency management policies [35, 36], interactive data repairing and cleaning systems [17, 18, 28, 29], as well as interactive data exploration tools [6, 20]. However, little work has been done on measuring inconsistency in databases, a problem which in contrast has been extensively investigated for propositional knowledge bases [22, 23, 24, 30, 31].

Measuring the amount of inconsistency in a database can help in understanding the primary sources of such conflicts as well as devising ways to deal with them. In fact, inconsistency measurement has recently been explored in [16] to quantify and understand inconsistency in bank holding company data with respect to four consistency rules (i.e. integrity constraints) specifically asserted for the considered domain. In general, quantifying and monitoring the amount of inconsistency helps to get information on the health status of data, whose quality is more and more important nowadays—the global market of data quality tools is expected to grow from USD 610.2 Million in 2017 to USD 1,376.7 Million by 2022 [34]. Furthermore, measuring inconsistency makes it possible to compare the amount of inconsistency between various chunks of information. Indeed, inconsistency measures (IMs for short) are important for inconsistency-tolerant integrity checking [14], where one wants to accept an update only if the measure of inconsistency of the old state does not increase in the new state.

Although the idea of measuring inconsistency was introduced more than 40 years ago, in [21], at that time it did not seem to be an important issue. The problem became more noticeable in the 1990s when it became possible to store large amounts of information. It was only in the early 2000s when several AI researchers started to investigate this issue systematically [31]. The bulk of this work since then has been for propositional knowledge bases, that is, where the information is presented as a set of formulas in propositional logic. In the last couple of years the work has been extended to other frameworks. [24] surveys what has been done so far and gives some extensions.

There are few works addressing the problem of measuring inconsistency in relational databases [27]. [37] first developed single-dependency axioms for dirtiness functions quantifying inconsistency w.r.t. one functional dependency—a simple type of denial constraint—considered in isolation, and proposed an IM that satisfies these axioms. Then a single axiom for dirtiness functions that handle multiple functional dependencies was proposed, although such functions are supposed to be built on top of a dirtiness function for single dependencies. The approach in [11, 15, 12] deals with relational databases from the point of view of first-order logic, as in logic programming. Its purpose is to show how database IMs can be applied to integrity checking [13, 14], relaxing repairs, and repair checking, which are applications that perfectly fit within our framework. However, the degrees of inconsistency defined in [12] form a partially ordered set; hence it is not always possible to compare the inconsistency of different databases. Finally, [3] proposes an IM based on an abstract repair semantics, where the degree of inconsistency depends on the distance between the database instance and the set of possible repairs under a given repair semantics, and an instantiation for cardinality-repairs that can be computed via answer-set programs [4]. Thus, previous work has not investigated the problem of tailoring propositional IMs to relational data, as well as analysing postulates compliance and the complexity of the resulting IMs, which is the focus of our work.

Contribution. In this paper, we introduce new IMs for relational databases with denial constraints. These measures are inspired by IMs for propositional logic. However, we do not apply well-known methods for propositional logic directly; rather, we use these methods as inspiration to define (by analogy) new IMs that are applicable to databases. In particular, we make the following contributions.

- We first introduce the database counterpart, namely $\mathscr{I}_x$ with $x \in \{B, M, \#, P, A, H, C, \eta\}$ of several propositional IMs I_x (Section 3). Every *database IM* $\mathscr{I}_x$ measures the inconsistency by blaming database tuples only. This is different from measuring the inconsistency of a set of formulas, all of which have the same status, as

[1] DIMES Department, University of Calabria, Italy, email: fparisi@dimes.unical.it

[2] University of Maryland at College Park, USA, email: grant@cs.umd.edu

Table 1. Postulates satisfaction for database inconsistency measures.

	Database Inconsistency Measures							
	$\mathscr{I}_B$	$\mathscr{I}_M$	$\mathscr{I}_\#$	$\mathscr{I}_P$	$\mathscr{I}_A$	$\mathscr{I}_H$	$\mathscr{I}_C$	$\mathscr{I}_\eta$
Free-Tuple Independence	✓	✓	✓	✓	✓	✓	✓⃝	✓
Penalty	✗	✓	✓	✓	✗	✗	✗	✗
Super-Additivity	✗	✓	✓	✓	✓⃝	✓	✓⃝	✗
MI-Separability	✗	✓	✓	✗	✗	✗	✗	✗
MI-Normalization	✓	✓	✗	✗	✗	✓	✓⃝	✗
Equal Conflict	✓	✓	✓	✓	✓	✓	✓⃝	✓

✓: satisfied for database measures (and satisfied for the corresponding propositional measure in the knowledge base setting).
✓⃝: satisfied for database measures but not for the corresponding propositional measure in the knowledge base setting.
✗: not satisfied for database measures (and not satisfied for propositional measures).

Table 2. Complexity of Lower Value (**LV**), Upper Value (**UV**), Exact Value (**EV**), and Inconsistency Measurement (**IM**) problems.

Inconsistency Measure(s)	$\mathbf{LV}_{\mathscr{I}}(D,v)$	$\mathbf{UV}_{\mathscr{I}}(D,v)$	$\mathbf{EV}_{\mathscr{I}}(D,v)$	$\mathbf{IM}_{\mathscr{I}}(D)$
$\mathscr{I}_B$,$\mathscr{I}_M$,$\mathscr{I}_\#$,$\mathscr{I}_P$	P	P	P	FP
$\mathscr{I}_A$	CP	CP	CP	#P-complete*
$\mathscr{I}_H$, $\mathscr{I}_C$	$coNP$-complete	NP-complete	D^p-complete	$FP^{NP[log\ n]}$-complete
$\mathscr{I}_\eta$	$coNP$-complete	NP-complete	D^p	FP^{NP}

*: #P-hardness follows from a result in [33], which also implies that **IM**, as well as **LV**, **UV**, and **EV**, are polynomial for *chain* FD schemas.

is the case for propositional knowledge bases. This leads to some substantial differences between the two cases: for instance, some measures that are distinct for propositional knowledge bases become identical in our setting (namely $\mathscr{I}_C$ and $\mathscr{I}_H$).

- We also introduce the database counterparts of several rationality postulates defined for propositional knowledge bases and check for compliance by the database IMs. It turns out that some postulates violated in the propositional case become satisfied in the database case (Section 4). Table 1 summarizes the results obtained, and allows us to compare the IMs in terms of the satisfied postulates. Intuitively, the behaviour of IMs that satisfy the same set of postulates can be considered as equivalent in those practical situations that are regulated by the selected postulates.
- Finally, we investigate the data complexity of the problems of deciding whether a given value is lower than (**LV**), greater than (**UV**), or equal to (**EV**) the inconsistency measured using a given database inconsistency measure $\mathscr{I}_x$ (Section 5). A summary of the results obtained for these problems, as well as for the problem of computing the actual value of an inconsistency measure (**IM** problem), is reported in Table 2. Interestingly, while the measures $\mathscr{I}_B$, $\mathscr{I}_M$, $\mathscr{I}_\#$, $\mathscr{I}_P$ become tractable in the database setting and the complexity of $\mathscr{I}_A$ decreases, the measures $\mathscr{I}_H$ and $\mathscr{I}_\eta$ remain hard as in the propositional case [44] even under data complexity.

Notably, $\mathscr{I}_M$ can computed in polynomial time and satisfies all the postulates in Table 1. One step down, we find $\mathscr{I}_\#$ which is also tractable and satisfies all the postulates but one. Next, $\mathscr{I}_P$ is still tractable and satisfies all the postulates but two. Finally, $\mathscr{I}_H$ (which as said above coincides with $\mathscr{I}_C$ in our setting) also satisfies all the postulates but two, though it turns out to be computationally hard.

2 Preliminaries

This section starts with our notation for relational databases and denial constraints and reviews some basic notions about complexity.

2.1 Relational Databases

We assume the existence of two finite (disjoint) sets: *Rel*, the set of relation names, and *Att*, the set of attribute names. We also assume a countably infinite database domain *Dom*, consisting of uninterpreted constants; elements of the domain with different names are different elements. Given a relation name $R \in Rel$, a *relation scheme* for it is a sorted list $(A_1,\dots,A_n)$ of attributes $A_1,\dots,A_n \in Att$, where n is said to be the *arity* of R and each attribute A_i (with $i \in [1..n]$) has associated a domain $DOM(A_i) \subseteq Dom$. We use $R(A_1,\dots,A_n)$ to denote a relation scheme. A *database scheme* $\mathscr{DS}$ is a nonempty finite set of relation schemes. A *tuple* over $R(A_1,\dots,A_n)$ is a mapping assigning to each attribute A_i of R a value $v_i \in DOM(A_i)$. Given a tuple $\vec{t} = \langle v_1,\dots,v_n\rangle$, we use $\vec{t}[A_i]$ to denote the value v_i of attribute A_i of $\vec{t}$. For an ordered list of attributes $\langle A_{i_1},\dots,A_{i_j}\rangle$, we use $\vec{t}[A_{i_1},\dots,A_{i_j}]$ to denote $\langle \vec{t}[A_{i_1}],\dots,\vec{t}[A_{i_j}]\rangle$. A *relation instance* (or simply relation) is a finite set of tuples over a given relation scheme, and a *database instance* (database) is a set of relations over a given database scheme. A *database instance* can be viewed as a finite Herbrand interpretation for a (function-free) first-order language using constant symbols in *Dom* and predicate symbols in *Rel*. Hence, we write $R(v_1,\dots,v_n)$ or $R(\vec{t})$ for denoting the (ground) atom corresponding to the tuple $\vec{t} = \langle v_1,\dots,v_n\rangle$ over R.

Integrity constraints are first-order sentences expressing properties that are supposed to be satisfied by the database instance. To define constraints, we extend the alphabet of the above-mentioned language to allow variables from a set *Var* of variables names (disjoint from *Rel* and *Att*). A *term* is either a constant in *Dom* or a variable in *Var*. An atom over a database scheme $\mathscr{DS}$ is an expression of the form $R(\tau_1,\dots,\tau_n)$ where R is a relation scheme in $\mathscr{DS}$ having arity n and $\tau_1,\dots,\tau_n$ are terms.

A *denial constraint* over $\mathscr{DS}$ is a first-order sentence of the form: $\forall \vec{x}_1,\dots,\vec{x}_k\ [\neg R_1(\vec{x}_1) \vee \dots \vee \neg R_k(\vec{x}_k) \vee \varphi(\vec{x}_1,\dots,\vec{x}_k)]$ where: (*i*) $\forall i \in [1..k]$, $\vec{x}_i$ are tuples of variables and $R_i(\vec{x}_i)$ are atoms over $\mathscr{DS}$; and (*ii*) φ is a disjunction of built-in predicates of the form $\tau_i \circ \tau_j$ where τ_i and τ_j are variables in $\vec{x}_1, \ldots, \vec{x}_k$ or constants, and $\circ \in \{=, \neq, >, <, \geq, \leq\}$. In the following, for the sake of readability, we will omit the prefix of universal quantifiers and write $[\neg R_1(\vec{x}_1) \vee \cdots \vee \neg R_k(\vec{x}_k) \vee \varphi(\vec{x}_1, \ldots, \vec{x}_k)]$ for a denial constraint. k is said to be the *arity* of the constraint. Denial constraints of arity 2 (resp. 3) are called *binary* (resp. *ternary*) constraints.

A *ground* constraint for $[\neg R_1(\vec{x}_1) \vee \cdots \vee \neg R_k(\vec{x}_k) \vee \varphi(\vec{x}_1, \ldots, \vec{x}_k)]$ is a formula $(\neg R_1(\vec{t}_1)) \vee \cdots \vee \neg(R_k(\vec{t}_k))$ such that $\varphi(\vec{t}_1, \ldots, \vec{t}_k)$ is true.

A *functional dependency* (FD) is a denial constraint of the form: $[\neg R(\vec{x}, y, \vec{z}) \vee \neg R(\vec{x}, u, \vec{w}) \vee (y = u)]$ where $\vec{x}, \vec{z}, \vec{w}$ are tuples of variables. It is usually written as $R : X \to Y$ (or simply $X \to Y$ if R is understood from the context), where X is the set of attributes of R corresponding to $\vec{x}$ and Y is the attribute corresponding to y (and u).

For a database scheme $\mathscr{DS}$ and a set $\mathscr{C}$ of integrity constraints over $\mathscr{DS}$, an instance D of $\mathscr{DS}$ is said to be *consistent* w.r.t. $\mathscr{C}$ iff $D \models \mathscr{C}$ in the standard model-theoretic sense.

2.2 Complexity

We briefly review some complexity classes used to characterize the complexity of decision and function problems (we refer the reader to [38] for further details). *P* (resp. *NP*) is the class of the decision problems that can be solved by a deterministic (resp. nondeterministic) Turing machine in polynomial time w.r.t. the size of the input of the problem. *coNP* is the class of the decision problems whose complement is in *NP*. Under the standard complexity-theoretic assumptions, it is conjectured that $P \subset NP$, $P \subset coNP$, and $NP \neq coNP$. A decision problem is in D^p iff it is the conjunction of a decision problem in *NP* and a decision problem in *coNP*.

We will also use the class *CP* [41], whose canonical complete problem is deciding whether a SAT expression has at least k many satisfying assignments. More formally, a problem is in *CP* if it is deciding whether $|\{y : \mathscr{P}(x,y) \text{ is true}\}| \geq k$, where $\mathscr{P}(x,y)$ is a predicate (with free variables x and y) such that (i) the size of y is polynomially bounded by the size of x, (ii) checking whether $\mathscr{P}(x,y)$ is true is in *P*, and (iii) the number k of solutions is a polynomial-time computable bound. The class *CP* coincides with the class *PP* of the decision problems that can be solved in polynomial time by a probabilistic Turing machine [41, 47]. The relationships between the classes are as follows: $NP \subseteq CP$ and $coNP \subseteq CP$. Differently from *NP*, the class *CP* is closed under complement, and, like *NP*, *CP* is closed under union and intersection.

Finally, we recall some complexity classes for function problems. *FP* is the class of the function problems that can be solved by a deterministic Turing machine in polynomial time (w.r.t. the size of the input of the problem). *#P* is the complexity class of the functions f such that f counts the number of accepting paths of a nondeterministic polynomial-time Turing machine [45]. For a complexity class $\mathscr{C}$, $FP^{\mathscr{C}}$ is the class of functions computable by a deterministic polynomial-time Turing machine using a $\mathscr{C}$-oracle. Thus, FP^{NP} is the class of problems that can be solved by a polynomial-time Turing machine that can ask a *polynomial* number of queries to an *NP* oracle. If a logarithmic number of queries is asked by the machine, then we have the class $FP^{NP[\log n]}$.

3 Database Inconsistency Measures

The idea of an inconsistency measure (IM) is to assign a nonnegative number to a knowledge base that measures its inconsistency [43]. Actually, there are two main types of IMs: an *absolute* measure measures the total amount of inconsistency; a *relative* measure is a ratio of the amount of inconsistency with respect to the size of the knowledge base. In the literature there is sometimes confusion between these two types; we will not deal with relative measures as these have not been studied in detail (as recently discussed in [5]).

In this section, we develop IMs for relational databases in analogy with IMs for propositional knowledge bases. In the following, we use $\mathbf{D}$ to denote the set of all database instances over a fixed but arbitrary database scheme $\mathscr{DS}$. We say that a database instance D is consistent, meaning that D is consistent w.r.t. a fixed but arbitrary set $\mathscr{C}$ of integrity constraints. In general, we will omit the database scheme and the set of integrity constraints in the terminology.

Now we are ready to define the inconsistency measure concept.

Definition 1 (Inconsistency Measure) *A function* $\mathscr{I} : \mathbf{D} \to \mathbb{R}^{\geq 0}_{\infty}$ *is an* **inconsistency measure** *if the following two conditions hold for all* $D, D' \in \mathbf{D}$*:*

Consistency $\mathscr{I}(D) = 0$ *iff D is consistent.*
Monotony *If* $D \subseteq D'$*, then* $\mathscr{I}(D) \leq \mathscr{I}(D')$*.*

Consistency and Monotony are called (rationality) postulates. Postulates are desirable properties for IMs and we will present additional ones later. However, we require that a function on databases must at least satisfy these two postulates in order to be called an IM. Consistency means that all and only consistent databases get measure 0. Monotony means that the enlargement of a database cannot decrease its measure. Monotony is not appropriate for relative measures where the ratio of inconsistency may decrease with the addition of consistent information; however, it is appropriate for absolute measures.

We now give some basic definitions needed to define IMs and their properties in this context.

A **minimal inconsistent subset** of D is a set of tuples $X \subseteq D$ such that X is inconsistent (with respect to $\mathscr{C}$) and no proper subset of X is inconsistent. We denote by $\mathsf{MI}(D)$ the set of minimal inconsistent subsets of D. Similarly, a **maximal consistent subset** is a set of tuples Y that is consistent and no proper superset of Y is consistent. We write $\mathsf{MC}(D)$ for the set of maximal consistent subsets (of D).

Any tuple that occurs in a minimal inconsistent subset is **problematic**; otherwise it is **free**. We use $\mathsf{Problematic}(D)$ and $\mathsf{Free}(D)$ to denote the sets of problematic and free tuples of D. A tuple t is **contradictory** if $\{t\}$ is inconsistent w.r.t. $\mathscr{C}$. We write $\mathsf{Contradictory}(D)$ for the set of contradictory tuples.

Example 1 (Running example) *Consider the database scheme* $\mathscr{DS}_{ex}$ *consisting of the relation scheme MealTicket(Number, Value, Holder, Date) whose instance contains the number, the value, the holder, and the issue date of meal tickets (one for each tuple) provided by a company to the employees. The set* $\mathscr{C}_{ex}$ *of integrity constraints consists of the following denial constraints:*

- $c_1 = [\neg \mathit{MealTicket}(x_1, x_2, x_3, x_4) \vee x_2 > 0]$*, stating that the value (i.e., the amount of the ticket) of every tuple of MealTicket must be a positive number.*
- $c_2 = [\neg \mathit{MealTicket}(x_1, x_2, x_3, x_4) \vee \neg \mathit{MealTicket}(x_1, x_5, x_6, x_7) \vee x_2 = x_5]$*, i.e., the FD Number→Value, stating that there cannot be two distinct tickets with the same number and different values.*
- $c_3 = [\neg \mathit{MealTicket}(x_1, x_2, x_3, x_4) \vee \neg \mathit{MealTicket}(x_1, x_5, x_6, x_7) \vee x_3 = x_6]$*, i.e., Number→Holder.*
- $c_4 = [\neg \mathit{MealTicket}(x_1, x_2, x_3, x_4) \vee \neg \mathit{MealTicket}(x_5, x_6, x_3, x_4) \vee \neg \mathit{MealTicket}(x_7, x_8, x_3, x_4) \vee x_1 = x_5 \vee x_1 = x_7 \vee x_5 = x_7]$*, stating*

the numerical dependency (see [25, 26]) Holder, Date $\to^2$*Number whose meaning is that for every holder and date there can be at most* 2 *meal ticket numbers.*

An instance D_{ex} *of* $\mathscr{D}\mathscr{S}_{ex}$ *consisting of* 7 *tuples* $\vec{t}_1,\ldots,\vec{t}_7$ *is shown in Table 3. Going through the integrity constraints we find that*
1) $\vec{t}_5$ *is inconsistent with* c_1*: it is a contradictory tuple;*
2) no pair of tuples violates c_2*;*
3) the two pairs of tuples, $\vec{t}_1$*,* $\vec{t}_3$*, and* $\vec{t}_2$*,* $\vec{t}_3$*, are inconsistent with* c_3*;*
4) the three tuples $\vec{t}_5$*,* $\vec{t}_6$*, and* $\vec{t}_7$ *together are inconsistent with* c_4*.*
Hence, $\mathsf{MI}(D_{ex}) = \{\{\vec{t}_5\},\{\vec{t}_1,\vec{t}_3\},\{\vec{t}_2,\vec{t}_3\}\}$*. Note that* $\{\vec{t}_5,\vec{t}_6,\vec{t}_7\}$ *is not included because it contains* $\{\vec{t}_5\}$*. Thus there are* 4 *problematic tuples in* D_{ex}*, and* 3 *free-tuples. Also,* $\mathsf{MC}(D) = \{\{\vec{t}_1,\vec{t}_2,\vec{t}_4,\vec{t}_6,\vec{t}_7\},\{\vec{t}_3,\vec{t}_4,\vec{t}_6,\vec{t}_7\}\}$*.*

Table 3. Database D_{ex}, which consists of the instance of *MealTicket*.

Number	*Value*	*Holder*	*Date*	Tuple
1001	15	Matthew	2018-12-13	$\vec{t}_1$
1001	15	Matthew	2018-12-18	$\vec{t}_2$
1001	15	Sophia	2018-12-17	$\vec{t}_3$
1004	20	Sophia	2018-12-17	$\vec{t}_4$
1005	0	Alex	2018-12-18	$\vec{t}_5$
1006	10	Alex	2018-12-18	$\vec{t}_6$
1007	20	Alex	2018-12-18	$\vec{t}_7$

3.1 Measures using Minimal Inconsistent Subsets

We start with the measures that rely on minimal inconsistent subsets and the related concepts defined above. We give the rationale for each measure below. The next definition is obtained by substituting the database D for the knowledge base K in each corresponding propositional IM using the concepts of minimal inconsistent and maximal consistent subsets as well as the sets of problematic and contradictory tuples defined earlier for our setting. In particular, the concept of contradictory tuples used to define the measure $\mathscr{I}_A$ has the same role as that of self-contradictions used to define the corresponding measure I_A for knowledge bases.

Definition 2 (Database Inconsistency Measures) *For a database D, the IMs* $\mathscr{I}_B$*,* $\mathscr{I}_M$*,* $\mathscr{I}_\#$*,* $\mathscr{I}_P$*,* $\mathscr{I}_A$*, and* $\mathscr{I}_H$ *are such that*

- $\mathscr{I}_B(D) = 1$ *if D is inconsistent and* $\mathscr{I}_B(D) = 0$ *if D is consistent.*
- $\mathscr{I}_M(D) = |\mathsf{MI}(D)|$.
- $\mathscr{I}_\#(D) = \begin{cases} 0 & \textit{if D is consistent,} \\ \sum\limits_{X\in\mathsf{MI}(D)} \frac{1}{|X|} & \textit{otherwise.} \end{cases}$
- $\mathscr{I}_P(D) = |\mathsf{Problematic}(D)|$.
- $\mathscr{I}_A(D) = (|\mathsf{MC}(D)| + |\mathsf{Contradictory}(D)|) - 1$.
- $\mathscr{I}_H(D) = \min\{|X| \; s.t.\; X \subseteq D \textit{ and } \forall M \in \mathsf{MI}(D),\; X \cap M \neq \emptyset\}$.

We explain the measures as follows. $\mathscr{I}_B$ is also called the drastic measure [30]: 0 means consistent; 1 means inconsistent. So this measure simply distinguishes between consistent and inconsistent databases. $\mathscr{I}_M$ counts the number of minimal inconsistent subsets [30]. The rationale is that a minimal inconsistent subset represents a minimal inconsistency for a set of database tuples; hence this measure counts the number of such inconsistencies. $\mathscr{I}_\#$ also counts the number of minimal inconsistent subsets, but it gives larger sets a smaller weight. The reason for $I_\#$ in the propositional logic setting is that when a minimal inconsistent set contains more formulas than another minimal inconsistent set, the former is intuitively less inconsistent than the latter [30]. For instance, we can say that intuitively $\{a,\neg a\}$ is more inconsistent than $\{a, a\to b, b\to c, c\to d, \neg d\}$. In the database setting, this measure gives a higher value to the violations of constraints having smaller arity, that is, to violations due to a smaller sets of tuples. It is a way to differentiate between constraints. For instance, in Example 1 this measure gives a violation of c_1 the highest value, a violation of c_4 the lowest value, and a violation of c_2 or c_3 a middle value. For databases, $\mathscr{I}_P$ counts the number of tuples that are in one or more minimal inconsistencies [22]. $\mathscr{I}_A$ uses the cardinality of the set of maximal consistent subsets [22], i.e. of the set of database repairs [1]. Intuitively, the larger this set, the larger is the space of different ways to get consistency, the higher the degree of inconsistency. Contradictory tuples are added as they do not appear in any way in a maximal consistent set; then 1 must be subtracted to obtain $\mathscr{I}_A(D) = 0$ for a consistent D because every consistent database has a maximal consistent subset, namely D itself. $\mathscr{I}_H$ counts the minimal number of tuples whose deletion makes the database consistent [23]. Hence $\mathscr{I}_H$ can be written as $\mathscr{I}_H = \min\{|X| \; s.t.\; D\setminus X$ is consistent $\}$. In fact, both $\mathscr{I}_A$ and $\mathscr{I}_H$ link the inconsistency measurement to the ways of restoring consistency, an idea recently explored in [3, 4] where the degree of inconsistency depends on the distance between the database instance and the set of possible repairs under a given repair semantics.

Example 2 *Now we calculate the values of the inconsistency measures for the database of our running example.*

- $\mathscr{I}_B(D_{ex}) = 1$ *as the database is inconsistent.*
- $\mathscr{I}_M(D_{ex}) = 3$ *as there are 3 minimal inconsistent subsets as given above.*
- $\mathscr{I}_\#(D_{ex}) = 1 + \frac{1}{2} + \frac{1}{2} = 2$ *as there is one minimal inconsistent subsets of size* 1 *and two minimal inconsistent subsets of size* 2*.*
- $\mathscr{I}_P(D_{ex}) = 4$ *as there are* 4 *distinct tuples in* $\mathsf{MI}(D_{ex})$*.*
- $\mathscr{I}_A(D_{ex}) = 2$ *because there are* 2 *maximal consistent subsets.*
- $\mathscr{I}_H(D_{ex}) = 2$ *as the set* $\{\vec{t}_3,\vec{t}_5\}$ *intersects with each minimal inconsistent subset.*

3.2 A Measure using Three-valued Logic

The next IM we consider is the Contension measure $\mathscr{I}_C$ [22], which uses a three-valued (3VL) logic. In our setting, a *3VL-interpretation* is a function i that assigns to each atom $R(\vec{t})$ in D one of the three truth values: T (*true*), F (*false*), or B (*both*). A interpretation i that assigns only the values in $\{T,F\}$ is said to be *classical*. The logical connectives are extended to 3VL interpretations as shown in Table 4, using Priest's three-valued logic. In classical two-valued logic, an interpretation is a *model* for a set of formulas if no formula gets the value F. The same condition is used for 3VL, but now, in addition to T the value B is also allowed. Given a database D with a set $\mathscr{C}$ of integrity constraints, a 3VL interpretation is a 3VL model iff all the constraints are satisfied and no atom $R(\vec{t}) \in D$ is assigned F. We use $\mathsf{Models}(D)$ to denote the set of 3VL models for a database D (with the integrity constraints in the background). Also, for a 3VL interpretation i we define $\mathsf{Conflictbase}(i) = \{R(\vec{t}) \mid i(R(\vec{t})) = B\}$, the tuples that have truth value B.

Definition 3 (Contension measure $\mathscr{I}_C$**)** *For a database D,* $\mathscr{I}_C$ *is such that* $\mathscr{I}_C(D) = \min\{|\mathsf{Conflictbase}(i)| \mid i \in \mathsf{Models}(D)\}$*.*

Table 4. Truth table for three valued logic (3VL). This semantics extends the classical semantics with a third truth value, B, denoting "inconsistency". Truth values on columns 1, 3, 7, and 9, give the classical semantics, and the other columns give the extended semantics.

Formula	Truth value								
ϕ	T	T	T	B	B	B	F	F	F
ψ	T	B	F	T	B	F	T	B	F
$\phi \vee \psi$	T	T	T	T	B	B	T	B	F
$\phi \wedge \psi$	T	B	F	B	B	F	F	F	F
$\neg\phi$	F	F	F	B	B	B	T	T	T

In the database setting $\mathscr{I}_C$ counts the minimal number of tuples that if we could consider them both true and false would resolve all inconsistencies. For the database of our running example, we have that $\mathscr{I}_C(D_{ex}) = 2$ as the minimal number of B values for an interpretation occur when $i(\vec{t}_1) = i(\vec{t}_2) = i(\vec{t}_4) = i(\vec{t}_6) = i(\vec{t}_7) = T$ and $i(\vec{t}_3) = i(\vec{t}_5) = B$.

3.3 A Probabilistic Measure

Finally, we define the database counterpart of the probabilistic measure I_η that uses the **PSAT (probabilistic satisfiability)** concept [31]. A PSAT instance is a set, $\Gamma = \{P(\phi_i) \geq p_i \mid 1 \leq i \leq m\}$, that assigns probability lower bounds to a set $\{\phi_1, \ldots, \phi_m\}$ of formulas; therefore $0 \leq p_i \leq 1$ for $1 \leq i \leq m$. A *probability function* over a set X is a function $\pi : X \to [0,1]$ such that $\sum_{x \in X} \pi(x) = 1$. Let Int be the set of all classical interpretations (of the set of formulas) and π a probability function over Int. The probability of a formula ϕ according to π is the sum of the probabilities assigned to the interpretations assigning T to ϕ, that is, $P_\pi(\phi) = \sum_{i \in Int, i(\phi)=T} \pi(i)$ for every formula $\phi \in K$. A PSAT instance is *satisfiable* if there is a probability function π over Int such that $P_\pi(\phi_i) \geq p_i$ for all $1 \leq i \leq m$.

I_η finds the maximum probability lower bound η that one can consistently assign to all formulas in a knowledge base; if η is equal to 1 then the knowledge base is consistent. In our setting, it means interpreting a database as a PSAT instance, where every atom $R(\vec{t})$ in the database is assigned probability η, and every (ground) integrity constraint is assigned a probability 1. Thus, given a database D, a set of integrity constraints $\mathscr{C}$, and a probability threshold $\eta \in [0,1]$, we define the PSAT instance $\Gamma_{\mathscr{C},\eta}(D) = \{P(R(\vec{t})) \geq \eta \mid R(\vec{t}) \in D\} \cup \{P(g(c)) = 1 \mid g(c)$ is a ground constraint for $c \in \mathscr{C}\}$, which enables the following definition.

Definition 4 (Probabilistic measure $\mathscr{I}_\eta$) *Given a database D and a set of integrity constraints $\mathscr{C}$, the inconsistency measure $\mathscr{I}_\eta$ is such that $\mathscr{I}_\eta(D) = 1 - \max\{\eta \in [0,1] \mid \Gamma_{\mathscr{C},\eta}(D)$ is satisfiable$\}$.*

Thus, $\mathscr{I}_\eta(D)$ is one minus the maximum probability lower bound one can consistently assign to all tuples in D. For the database of our running example we have that $\mathscr{I}_\eta(D_{ex}) = 1$ because the maximum probability that can be assigned to (the contradictory) tuple $\vec{t}_5$ is zero.

In the propositional case, of the measures considered no 2 measures give the same result for all knowledge bases. But because of the special structure of databases, for database IMs there is an equality that does hold: $\mathscr{I}_C(D) = \mathscr{I}_H(D)$. In fact, $\mathscr{I}_C(D)$ counts the minimum number of tuples that need to be assigned B in order to get a 3VL model in $\mathsf{Models}(D)$. This is the same as the cardinality of the set $X \subseteq D$ having a non-empty intersection with every minimal inconsistent subset of D.

Proposition 1 *For any database D, $\mathscr{I}_C(D) = \mathscr{I}_H(D)$.*

As will be noted after Theorem 1, no other pair of measures considered in this section is identical.

4 Rationality Postulate Satisfaction

In addition to devising many ways of measuring inconsistency, researchers have also investigated properties that a good IM should possess. These are called (rationality) postulates and we already gave two of them: Consistency and Monotony, that all (absolute) IMs should satisfy. [43] lists 16 additional postulates but some of them are oriented toward relative measures or deal with logically equivalent formulas and so are not relevant for relational databases. [3] As we did with the database IMs in Definition 2, we now give the database counterparts to a list of propositional rationality postulates. These postulates were originally introduced based on some intuition about how an IM should behave, but there is no consensus about which are essential or even which are really desirable. However, we think that in general an IM behaves more in line with our intuition if it satisfies more postulates.

Definition 5 (Postulates for Database Inconsistency Measures) *Let D, D' be databases, $R(\vec{t})$ a tuple of D, and $\mathscr{I}$ an IM. The postulates for database IMs are as follows:*

Free-Tuple Independence *If $R(\vec{t}) \in \mathsf{Free}(D)$, then $\mathscr{I}(D) = \mathscr{I}(D \setminus \{R(\vec{t})\})$.*
Penalty *If $R(\vec{t}) \in \mathsf{Problematic}(D)$, then $\mathscr{I}(D) > \mathscr{I}(D \setminus \{R(\vec{t})\})$.*
Super-Additivity *If $D \cap D' = \varnothing$, then $\mathscr{I}(D \cup D') \geq \mathscr{I}(D) + \mathscr{I}(D')$.*
MI-Separability *If $\mathsf{MI}(D \cup D') = \mathsf{MI}(D) \cup \mathsf{MI}(D')$ and $\mathsf{MI}(D) \cap \mathsf{MI}(D') = \varnothing$, then $\mathscr{I}(D \cup D') = \mathscr{I}(D) + \mathscr{I}(D')$.*
MI-Normalization *If $M \in \mathsf{MI}(D)$, then $\mathscr{I}(M) = 1$.*
Equal Conflict *If $M, M' \in \mathsf{MI}(D)$ and $|M| = |M'|$, then $\mathscr{I}(M) = \mathscr{I}(M')$.*

Independence means that free tuples do not change the inconsistency measure. Penalty states that deleting a problematic tuple decreases the measure. Thus, measures satisfying these postulates allow us to check if deleting a given tuple makes a database less inconsistent or not, in the sense that the inconsistency value decreases iff the deleted tuple occurs is a minimal inconsistent subset. So for an IM that satisfies both Independence and Penalty, e.g. $\mathscr{I}_M$, we can tell from the change or no change of the IM after a deletion if the formula was free or problematic. But for an IM that violates Penalty, e.g. $\mathscr{I}_H$, that is not the case. For the database D_{ex} of our running example, the fact that $\vec{t}_1$ is problematic is signaled by $\mathscr{I}_M$ since $\mathscr{I}_M(D_{ex} \setminus \{\vec{t}_1\}) = 2 < \mathscr{I}_M(D_{ex}) = 3$, but not by $\mathscr{I}_H$ which gives $\mathscr{I}_H(D_{ex} \setminus \{\vec{t}_1\}) = \mathscr{I}_H(D_{ex}) = 2$. This would negatively affect applications of IMs where, for instance, the inconsistency value is used to detect if updates involve inconsistent tuples.

Super-Additivity and MI-Separability give information about the union of two databases under certain conditions. Super-Additivity deals with the case where the databases are disjoint in which case the measure of the union is at least as great as the sum of the measures of the two databases. MI-Separability requires that the minimal

[3] Propositional postulates that are not relevant or applicable to databases are for instance *Dominance* and *Exchange*. Indeed, the database version of Dominance would be: if $R(\vec{t})$ is not contradictory and $R(\vec{t})$ implies $R'(\vec{t}')$ then $\mathscr{I}(D \cup \{R(\vec{t})\}) \geq \mathscr{I}(D \cup \{R'(\vec{t}')\})$. But a database tuple cannot imply any other database tuple, and thus the postulate would be trivially satisfied. Analogously, the database version of Exchange would be: if D_1 is consistent and D_1 is logically equivalent to D_2 then $\mathscr{I}(D \cup \{D_1\}) = \mathscr{I}(D \cup \{D_2\})$. This is also useless as two different databases cannot be logically equivalent.

inconsistent sets of the two databases partition the minimal inconsistent sets of the union, in which case the measure of the union is the sum of the measures of the two databases. Hence, a measure satisfying Super-Additivity is able to separately take into account the inconsistency arising from databases containing different information when they are merged. A violation to Super-Additivity means that the act of merging loses some portion of the IM. The satisfaction of MI-Separability guarantees that inconsistencies originating from different minimal sets of tuples are counted individually, even if the databases have a non-empty intersection. A violation of MI-Separability means that even though the two databases have completely different minimal inconsistent subsets, the merger causes (usually) additional inconsistencies.

MI-Normalization and Equal Conflict deal specifically with minimal inconsistent sets. MI-Normalization requires every minimal inconsistent set to have measure 1: it is the essence of $\mathscr{I}_M$. Equal Conflict requires minimal inconsistent sets of the same size to have the same measure. This means that violations originated from the same constraint, and thus having the same size, should be counted the same. Clearly, MI-Normalization implies Equal Conflict but the converse does not hold.

It turns out that the satisfaction of the postulates for database IMs is very similar to but not identical to the satisfaction of the corresponding postulates for the propositional IMs.

Theorem 1 *The satisfaction of postulates for database inconsistency measures is as given in Table 1.*

As shown in Table 1 there are 5 cases where the satisfaction results are positive for database IMs but negative for the propositional case. For each case we give a propositional example where the postulate is violated while it is satisfied for databases. We first recall the propositional versions, namely I_A and I_C, of the two database IMs $\mathscr{I}_A$ and $\mathscr{I}_C$. Given a propositional knowledge base K, $I_A(K)$ gives the number of maximal consistent subsets of K plus the number of self-contradictions of K minus 1. Measure I_C counts the minimal number of atoms in K that must be assigned the truth-value B in the three-valued logic by an interpretation that satisfies every formula in K. The following examples show that the violation of postulates is due to the power of propositional logic; for databases only a simpler class of formulas is allowed.

- Free-Tuple Independence for I_C. Let $K_0 = \{a \wedge \neg a \wedge b, \neg b\}$ where $\neg b$ is free because it does not belong to the unique minimal inconsistent subset of K_0, consisting of the self-contradictory formula $a \wedge \neg a \wedge b$. It is violated since $I_C(K_0) = 2$, while $I_C(K_0 \setminus \{\neg b\}) = 1$.
- Super-Additivity for I_A. Let $K_1 = \{a, \neg a\}$, $K_2 = \{a \wedge a, \neg a \wedge \neg a\}$. Since both K_1 and K_2, as well their union, contain 2 maximal consistent subsets and no self-contradictions, then $I_A(K_1) = I_A(K_2) = I_A(K_1 \cup K_2) = 1$, which violates the postulate.
- Super-Additivity for I_C. Using again K_1 and K_2, we have that $I_C(K_1) = I_C(K_2) = I_C(K_1 \cup K_2) = 1$, as assigning B to atom a suffices to get a 3VL interpretation that satisfies every formula.
- MI-Normalization for I_C. Let $K_3 = \{a \wedge \neg a \wedge b \wedge \neg b\}$. Then $I_C(K_3) = 2$, as both atoms should be assigned the truth-value B.
- Equal Conflict for I_C. Let $K_4 = \{a \wedge \neg a\}$. Although K_3 and K_4 are two minimal inconsistent subsets of formulas of the same cardinality 1, $I_C(K_3) = 2$ is different from $I_C(K_4) = 1$.

As a consequence of Theorem 1, we have that, except for $\mathscr{I}_C$ and $\mathscr{I}_H$, no other pair of measures in Table 1 are identical since they do not satisfy exactly the same set of postulates.

5 Complexity of Database Inconsistency Measures

We investigate the data-complexity [8, 46] of the following three decision problems, which intuitively ask if a given rational value v is, respectively, lower than, greater than, or equal to the value returned by a given IM when applied to a given database. Observe that every IM returns a rational number, including I_η (as shown in [31]).

Definition 6 (Lower (LV), Upper (UV), and Exact Value (EV)) *Let $\mathscr{I}$ be an IM. Given a database D over a fixed database scheme with a fixed set of constraints, and a positive value $v \in \mathbb{Q}^{>0}$,*

- ***$LV_{\mathscr{I}}$**(D, v) is the problem of deciding whether $\mathscr{I}(D) \geq v$.*

Given D and a non-negative value $v' \in \mathbb{Q}^{\geq 0}$,

- ***$UV_{\mathscr{I}}$**(D, v') is the problem of deciding whether $\mathscr{I}(D) \leq v'$, and*
- ***$EV_{\mathscr{I}}$**(D, v') is the problem of deciding whether $\mathscr{I}(D) = v'$.*

We also consider the problem of determining the IM value.

Definition 7 (Inconsistency Measurement (IM) problem) *Let $\mathscr{I}$ be an inconsistency measure. Given a database D over a fixed database scheme with a fixed set of constraints, **$IM_{\mathscr{I}}$**(D) is the problem of computing the value of $\mathscr{I}(D)$.*

The following proposition states that the measure $\mathscr{I}_M$ is tractable in the presence of denial constraints. The result follows from the fact that the number of minimal inconsistent subsets is polynomial (in the number of tuples in the database instance). In fact, since under data complexity the database schema and the set of constraints are considered fixed, and only the database instance is considered as part of the input, the arity of the constraints is considered to be a constant, which in turn bounds the size of each minimal inconsistent subset.

Proposition 2 ***$IM_{\mathscr{I}_M}$**(D) is in FP.*

The result of Proposition 2 implies that $\mathbf{LV}_{\mathscr{I}_M}(D,v)$, $\mathbf{UV}_{\mathscr{I}_M}(D,v)$, and $\mathbf{EV}_{\mathscr{I}_M}(D,v)$ are in P. Moreover, the following corollary states that $\mathscr{I}_B, \mathscr{I}_P$ and $\mathscr{I}_\#$ are tractable as well. In fact, $\mathscr{I}_B(D) = 1$ iff $\mathscr{I}_M(D) \geq 1$, and since minimal inconsistent subsets can be enumerated in polynomial time, the computation of the number of problematic tuples (i.e., $\mathscr{I}_P$) and the weighted sum of the minimal inconsistent subsets (i.e., $\mathscr{I}_\#$) is feasible as well.

Corollary 1 *For $\mathscr{I} \in \{\mathscr{I}_B, \mathscr{I}_P, \mathscr{I}_\#\}$, **$IM_{\mathscr{I}}$**$(D)$ is in FP and **$LV_{\mathscr{I}}$**(D,v), **$UV_{\mathscr{I}}$**(D,v), and **$EV_{\mathscr{I}}$**(D,v) are in P.*

Complexity of the measure $\mathscr{I}_A$

We now consider the measure $\mathscr{I}_A$ which counts the number of maximal consistent subsets. For denial constraints, the set of conflicts among tuples can be naturally represented as a *conflict hypergraph* $\mathscr{H}(D)$ whose nodes are the tuples of D and the set of hyperedges is $\mathsf{MI}(D)$ [9, 10, 19]. Maximal consistent subsets one-to-one correspond to maximal independent sets of $\mathscr{H}(D)$ (a maximal independent set for $\mathscr{H}(D)$ is a maximal set of nodes that contains no hyperedge from $\mathsf{MI}(D)$). Hence, computing $\mathbf{IM}_{\mathscr{I}_A}(D)$ can be reduced to computing the number of maximal independent sets of $\mathscr{H}(D)$. Since counting the maximal independent sets of a *graph* is #P-complete [40], it follows that $\mathbf{IM}_{\mathscr{I}_A}(D)$ is in #P for *binary* denial constraints (for which the conflict hypergraph is a graph). However, the membership in #P of $\mathbf{IM}_{\mathscr{I}_A}$ also holds for general denial constraints. In fact, computing $\mathscr{I}_A(D)$ means counting the number of accepting paths of a nondeterministic polynomial-time Turing machine whose accepting paths are the elements in $\mathsf{MC}(D)$ (we can check in polynomial time if a guessed set $S \subseteq D$ of tuples belongs to $\mathsf{MC}(D)$, as it suffices to verify that for each $t \in D \setminus S$, $S \cup \{t\}$ is inconsistent).

If we focus on FDs only, a result stronger than that above follows from the dichotomy result established in [33] where it is shown that computing the number of subset repairs [1] (i.e., maximal consistent subsets) is either #P-complete or tractable for FDs. The latter case holds if, for a given set $\mathscr{F}$ of FDs, there is a minimal cover $\mathscr{F}_m$ of $\mathscr{F}$ which is a *chain*, that is, for every pair of FDs $R : X_1 \rightarrow Y_1$ and $R : X_2 \rightarrow Y_2$ in $\mathscr{F}_m$, either $X_1 \subseteq X_2$ or $X_2 \subseteq X_1$—in such case we say that $\mathscr{F}$ is (equivalent to) a chain FD schema.

Proposition 3 *$\mathbf{IM}_{\mathscr{I}_A}(D)$ is in #P. Moreover, if the set of constraints consists of FDs only and is equivalent to a chain FD schema, then $\mathbf{IM}_{\mathscr{I}_A}(D)$ is in FP; otherwise it is #P-complete.*

The following theorem states that the three decision problems for $\mathscr{I}_A$ are in the class CP, which coincides with PP [41, 47]. It is worth noting that these problems are unlikely to belong to classes contained in CP (under the standard complexity assumption $P \neq NP$). Specifically, $\mathbf{UV}_{\mathscr{I}_A}(D,v)$ (resp., $\mathbf{LV}_{\mathscr{I}_A}(D,v)$, $\mathbf{EV}_{\mathscr{I}_A}(D,v)$) is unlikely to be in NP (resp., $coNP$, D^P) as checking whether a given set of sets is a set of maximal consistent subsets cannot be accomplished in polynomial time due to the exponential size of $|\mathsf{MC}(D)|$ w.r.t. D.

Theorem 2 *$\mathbf{LV}_{\mathscr{I}_A}(D,v)$, $\mathbf{UV}_{\mathscr{I}_A}(D,v)$, and $\mathbf{EV}_{\mathscr{I}_A}(D,v)$ are in CP.*

Complexity of the measure $\mathscr{I}_H$

The following theorem characterizes the complexity of the Upper Value problem for $\mathscr{I}_H$ (and thus for $\mathscr{I}_C$, see Proposition 1), showing that it is in NP and NP-hard in the presence of either (*i*) just one ternary denial constraint, or (*ii*) two FDs such that the right-hand side of the first one coincides with the left-hand side of the second one, or (*iii*) two FDs without common attributes. The hardness for case (*i*) can be proved by showing a reduction from the VERTEX COVER problem, for which we can define a construction that can be exploited in the proof of Theorems 4 and 5 for characterizing the complexity of the Exact Value problem as well as that of computing the value of $\mathscr{I}_H$. The hardness for cases (*i*) and (*ii*) can be proved, respectively, by reduction from 3SAT and a problem investigated in [2] about the size of subset repairs.

Theorem 3 *$\mathbf{UV}_{\mathscr{I}_H}(D,v)$ is NP-complete. Specifically, let $\mathscr{C}$ be the set of integrity constraints. $\mathbf{UV}_{\mathscr{I}_H}(D,v)$ is NP-hard if one of the following cases holds:*

1. *$\mathscr{C}$ consists of a single ternary denial constraint;*
2. *$\mathscr{C}$ consists of 2 FDs of the form $R : A \rightarrow B$ and $R : B \rightarrow E$;*
3. *$\mathscr{C}$ consists of 2 FDs of the form $R : A \rightarrow B$ and $R : C \rightarrow D$.*

Since $\mathscr{I}_H(D) \geq v$ iff $\mathscr{I}_H(D) \nleq v-1$, i.e., $\mathbf{LV}_{\mathscr{I}_H}(D,v)$ is true iff $\mathbf{UV}_{\mathscr{I}_H}(D,v-1)$ is false, using the result of Theorem 3, we have that $\mathbf{LV}_{\mathscr{I}_H}(D,v)$ is $coNP$-complete. Moreover, since $\mathbf{EV}_{\mathscr{I}_H}(D,v)$ is the intersection of $\mathbf{LV}_{\mathscr{I}_H}(D,v)$ and $\mathbf{UV}_{\mathscr{I}_H}(D,v)$, it follows that $\mathbf{EV}_{\mathscr{I}_H}(D,v)$ is in D^p.

Corollary 2 *$\mathbf{LV}_{\mathscr{I}_H}(D,v)$ is coNP-complete. $\mathbf{EV}_{\mathscr{I}_H}(D,v)$ is in D^p.*

The characterization of the complexity of $\mathbf{EV}_{\mathscr{I}_H}(D,v)$ is strengthened by Theorem 4 that can be proved by showing a reduction to our problem from the D^P-hard problem EXACT VERTEX COVER [39]. Therefore, it follows that $\mathbf{EV}_{\mathscr{I}_H}(D,v)$ is complete for D^p.

Theorem 4 *$\mathbf{EV}_{\mathscr{I}_H}(D,v)$ is D^p-hard.*

Finally, the following theorem provides a tight characterization of the complexity of computing the value of $\mathscr{I}_H$.

Theorem 5 *$\mathbf{IM}_{\mathscr{I}_H}(D)$ is $FP^{NP[log\ n]}$-complete.*

The membership in $FP^{NP[log\ n]}$ follows from the fact that the value of $\mathscr{I}_H(D)$ can be computed by performing a binary search in the interval $[0,|D|]$, that is by asking a logarithmic number of queries to an NP-oracle solving $\mathbf{UV}_{\mathscr{I}_H}(D,v)$, where v is chosen as usual in binary search. The hardness result can be proved by showing a reduction from the $FP^{NP[log\ n]}$-complete problem MIN-VERTEX COVER [38].

Complexity of the measure $\mathscr{I}_\eta$

We now characterize the complexity of the probabilistic IM $\mathscr{I}_\eta$. It turns out that the Upper Value problem is NP-complete. The membership in NP follows from the fact that the problem belongs to NP for general propositional knowledge bases [44]. On the other hand, our NP-hardness result strengthens the hardness result previously shown for general knowledge bases, as it is shown that the problem is hard even for restricted forms of knowledge bases encoding relational databases with denial constraints.

Theorem 6 *$\mathbf{UV}_{\mathscr{I}_\eta}(D,v)$ is NP-complete. Specifically, let $\mathscr{C}$ be the set of integrity constraints. $\mathbf{UV}_{\mathscr{I}_\eta}(D,v)$ is NP-hard if one of the following cases holds:*

1. *$\mathscr{C}$ consists of 2 FDs of the form $R : A \rightarrow B$ and $R : BC \rightarrow A$;*
2. *$\mathscr{C}$ consists of a single ternary denial constraint.*

The result above can be proved by showing a reduction from 3-COLORABILITY to $\mathbf{UV}_{\mathscr{I}_\eta}(D,v)$. From Theorem 6, the following corollary follows.

Corollary 3 *$\mathbf{LV}_{\mathscr{I}_\eta}(D,v)$ is coNP-complete. $\mathbf{EV}_{\mathscr{I}_\eta}(D,v)$ is in D^p.*

Finally, since it can be shown that the number of distinct values that $\mathscr{I}_\eta(D)$ can take is polynomial [42], it can be computed by a polynomial number of queries to an NP oracle deciding $\mathbf{UV}_{\mathscr{I}_\eta}$.

Corollary 4 *$\mathbf{IM}_{\mathscr{I}_\eta}(D)$ is in FP^{NP}.*

6 Conclusions and Future Work

In this paper, we have introduced a formal framework for measuring inconsistency in databases. We believe that the definition and investigation of IMs for databases benefit from our systematic approach to the problem, which stems from what has been done in the past by the AI community. The results summarized in Tables 1 and 2 give indications on the behavior and complexity of IMs for databases, helping to figure out which measure is more appropriate for specific applications. Our analysis shows that $\mathscr{I}_M$ satisfies all the postulates and can be computed in polynomial time, suggesting that it can be used in practice. In fact, it is easy to see that $\mathscr{I}_M$, as well as $\mathscr{I}_\#$ and $\mathscr{I}_P$, can be evaluated by standard SQL, meaning that measuring inconsistency using these measures scales as far as the database can answer queries in reasonable time. Moreover, compared with their propositional versions, we found that the complexity of $\mathscr{I}_A$ decreases while that of $\mathscr{I}_H$ and $\mathscr{I}_\eta$ remains the same. Also, additional postulates become satisfied for $\mathscr{I}_A$ and $\mathscr{I}_C$, and $\mathscr{I}_C$ becomes identical to $\mathscr{I}_H$.

Many interesting issues concerning IMs in databases remain unexplored. We have dealt with denial constraints, a common type of integrity constraint which can express for instance equality generating dependencies (e.g., functional dependencies). We plan to extend our work to other types of integrity constraints, and in particular to inclusion dependencies. Also, we plan to identify tractable cases for the hard measures, possibly exploiting connections with work done on inconsistent databases (as shown for $\mathscr{I}_A$), and devise efficient algorithms for evaluating IMs. The IMs we have considered work at the tuple-level, without distinguishing inconsistency arising from different attributes, which is another issues we want to address in the future. Finally, another interesting direction for future work is considering incomplete information (e.g., databases with null values).

REFERENCES

[1] M. Arenas, L. E. Bertossi, and J. Chomicki, 'Consistent query answers in inconsistent databases', in *Proc. of Symposium on Principles of Database Systems (PODS)*, pp. 68–79, (1999).

[2] M. Arenas, L. E. Bertossi, J. Chomicki, X. He, V. Raghavan, and J. P. Spinrad, 'Scalar aggregation in inconsistent databases', *Theor. Comput. Sci.*, **296**(3), 405–434, (2003).

[3] L. E. Bertossi, 'Measuring and computing database inconsistency via repairs', in *Proc. of International Conference on Scalable Uncertainty Management (SUM)*, pp. 368–372, (2018).

[4] L. E. Bertossi, 'Repair-based degrees of database inconsistency', in *Proc. of Logic Programming and Nonmonotonic Reasoning (LPNMR)*, pp. 195–209, (2019).

[5] P. Besnard and J. Grant, 'Relative inconsistency measures', *Artif. Intell.*, **280**, 103231, (2020).

[6] T. Bleifuß, L. Bornemann, D. V. Kalashnikov, F. Naumann, and D. Srivastava, 'Dbchex: Interactive exploration of data and schema change', in *Proc. of Conf. on Innovative Data Systems Research (CIDR)*, (2019).

[7] M. Calautti, L. Libkin, and A. Pieris, 'An operational approach to consistent query answering', in *Proc. of ACM Symposium on Principles of Database Systems (PODS)*, pp. 239–251, (2018).

[8] A. K. Chandra and D. Harel, 'Computable queries for relational data bases', *J. Comput. Syst. Sci.*, **21**(2), 156–178, (1980).

[9] J. Chomicki and J. Marcinkowski, 'Minimal-change integrity maintenance using tuple deletions', *Inf. Comput.*, **197**(1-2), 90–121, (2005).

[10] J. Chomicki, J. Marcinkowski, and S. Staworko, 'Computing consistent query answers using conflict hypergraphs', in *Proc. of Int. Conf. on Information and Knowledge Management (CIKM)*, pp. 417–426, (2004).

[11] H. Decker, 'Inconsistency-tolerant database repairs and simplified repair checking by measure-based integrity checking', *T. Large-Scale Data- and Knowledge-Centered Systems*, **34**, 153–183, (2017).

[12] H. Decker, 'Measuring database inconsistency', in *Measuring Inconsistency in Information*, eds., John. Grant and Maria Vanina Martinez, 271–311, College Publications, (2018).

[13] H. Decker and D. Martinenghi, 'Classifying integrity checking methods with regard to inconsistency tolerance', in *Proc. of International Conference on Principles and Practice of Declarative Programming (PPDP)*, pp. 195–204, (2008).

[14] H. Decker and D. Martinenghi, 'Inconsistency-tolerant integrity checking', *IEEE Trans. Knowl. Data Eng.*, **23**(2), 218–234, (2011).

[15] H. Decker and S. Misra, 'Database inconsistency measures and their applications', in *Proc. of International Conference on Information and Software Technologies (ICIST)*, pp. 254–265, (2017).

[16] L. Fan, M. D. Flood, and J. Grant, 'Measuring inconsistency in bank holding company data', in *Proc. of SIGMOD Workshop on Data Science for Macro-modeling with Financial and Economic Datasets (DSMM)*, pp. 5:1–5:5, (2019).

[17] B. Fazzinga, S. Flesca, F. Furfaro, and F. Parisi, 'DART: A data acquisition and repairing tool', in *EDBT Workshop on Inconsistency and Incompleteness in Databases (IIDB)*, pp. 297–317, (2006).

[18] S. Flesca, F. Furfaro, and F. Parisi, 'Preferred database repairs under aggregate constraints', in *Proc. of International Conference on Scalable Uncertainty Management (SUM)*, pp. 215–229, (2007).

[19] S. Flesca, F. Furfaro, and F. Parisi, 'Consistency checking and querying in probabilistic databases under integrity constraints', *J. Comput. Syst. Sci.*, **80**(7), 1448–1489, (2014).

[20] A. Giuzio, G. Mecca, E. Quintarelli, M. Roveri, D. Santoro, and L. Tanca, 'INDIANA: an interactive system for assisting database exploration', *Inf. Syst.*, **83**, 40–56, (2019).

[21] J. Grant, 'Classifications for inconsistent theories', *Notre Dame Journal of Formal Logic*, **XIX**(3), 435–444, (1978).

[22] J. Grant and A. Hunter, 'Measuring consistency gain and information loss in stepwise inconsistency resolution', in *Proc. of European Conference Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU)*, pp. 362–373, (2011).

[23] J. Grant and A. Hunter, 'Distance-based measures of inconsistency', in *Proc. of European Conf. on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU)*, pp. 230–241, (2013).

[24] J. Grant and M. V. Martinez, *Measuring Inconsistency in Information*, College Publications, 2018.

[25] J. Grant and J. Minker, 'Inferences for numerical dependencies', *Theoretical Computer Science*, **41**, 271–287, (1985).

[26] J. Grant and J. Minker, 'Normalization and axiomatization for numerical dependencies', *Information and Control*, **65**(1), 1–17, (1985).

[27] J. Grant and F. Parisi, 'Measuring inconsistency in a general information space', in *Proc. of Int. Symposium on Foundations of Information and Knowledge Systems (FoIKS)*, pp. 140–156, (2020).

[28] S. Hao, N. Tang, G. Li, J. He, N. Ta, and J. Feng, 'A novel cost-based model for data repairing', *IEEE TKDE*, **29**(4), 727–742, (2017).

[29] J. He, E. Veltri, D. Santoro, G. Li, G. Mecca, P. Papotti, and N. Tang, 'Interactive and deterministic data cleaning', in *Proc. of International Conference on Management of Data (SIGMOD)*, pp. 893–907, (2016).

[30] A. Hunter and S. Konieczny, 'Measuring inconsistency through minimal inconsistent sets', in *Proc. of Int. Conf. on Principles of Knowledge Representation and Reasoning (KR)*, pp. 358–366, (2008).

[31] K. Knight, 'Measuring inconsistency', *J. Philosophical Logic*, **31**(1), 77–98, (2002).

[32] A. Y. Levy, 'Combibining artificial intelligence and databases for data integration', in *Artificial Intelligence Today: Recent Trends and Developments*, 249–268, (1999).

[33] E. Livshits and B. Kimelfeld, 'Counting and enumerating (preferred) database repairs', in *Proc. of Symposium on Principles of Database Systems (PODS)*, pp. 289–301, (2017).

[34] MarketsandMarkets. Data quality tools market by data type. `https://www.marketsandmarkets.com/Market-Reports/data-quality-tools-market-22437870.html`, 2019.

[35] M. V. Martinez, F. Parisi, A. Pugliese, G. I. Simari, and V. S. Subrahmanian, 'Inconsistency management policies', in *Proc. of Int. Conf. on Principles of Knowledge Repr. and Reas. (KR)*, pp. 367–377, (2008).

[36] M. V. Martinez, F. Parisi, A. Pugliese, G. I. Simari, and V. S. Subrahmanian, 'Policy-based inconsistency management in relational databases', *Int. J. Approx. Reasoning*, **55**(2), 501–528, (2014).

[37] M. V. Martinez, A. Pugliese, G. I. Simari, V. S. Subrahmanian, and H. Prade, 'How dirty is your relational database? an axiomatic approach', in *Proc. of European Conf. on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, pp. 103–114, (2007).

[38] C. H. Papadimitriou, *Computational complexity*, Addison-Wesley, Reading, Massachusetts, 1994.

[39] C. H. Papadimitriou and M. Yannakakis, 'The complexity of facets (and some facets of complexity)', *J. C. Syst. Sci.*, **28**(2), 244–259, (1984).

[40] J. S. Provan and M. O. Ball, 'The complexity of counting cuts and of computing the probability that a graph is connected', *SIAM J. Comput.*, **12**(4), 777–788, (1983).

[41] J. Simon, 'On the difference between one and many (preliminary version)', in *Proceedings of Fourth Colloquium on Automata, Languages and Programming (ICALP)*, pp. 480–491, (1977).

[42] M. Thimm, 'On the expressivity of inconsistency measures', *Artif. Intell.*, **234**, 120–151, (2016).

[43] M. Thimm, 'On the evaluation of inconsistency measures', in *Measuring Inconsistency in Information*, eds., John. Grant and Maria Vanina Martinez, 19–60, College Publications, (2018).

[44] M. Thimm and J. P. Wallner, 'Some complexity results on inconsistency measurement', in *Proc. of International Conference Principles of Knowledge Representation and Reasoning (KR)*, pp. 114–124, (2016).

[45] L. G. Valiant, 'The complexity of computing the permanent', *Theor. Comput. Sci.*, **8**, 189–201, (1979).

[46] M. Y. Vardi, 'The complexity of relational query languages', in *Proc. of Symposium on Theory of Computing (STOC)*, pp. 137–146, (1982).

[47] K. W. Wagner, 'The complexity of combinatorial problems with succinct input representation', *Acta Inf.*, **23**(3), 325–356, (1986).