How Many Inconsistencies Are In Your Database?*

(Discussion Paper)

Francesco Parisi[1], John Grant[2]

[1] DIMES Department, University of Calabria, Italy
[2] University of Maryland at College Park, USA

Abstract

Measuring inconsistency is an approach that provides ways to quantify the severity of inconsistency and helps understanding the primary sources of conflicts. In this paper, we discuss inconsistency measures for indefinite databases, which allow for indefinite or partial information which is formally expressed by means of disjunctive tuples. We introduce inconsistency measures for indefinite databases with denial constraints, and explore the complexity of the problem of computing the value of the proposed inconsistency measures as well as of the problems of deciding whether the inconsistency value is lower than, greater than, or equal to a given threshold for indefinite and definite databases.

Keywords

Relational database, Indefinite database, Inconsistency measure, Denial constraints

1. Introduction

Handling conflicting information is an important challenge. In fact, data of poor quality can significantly limit the implementation of effective AI solutions [2, 3]. So, having information on the quality of data used in data-driven approaches is crucial, as poor quality data can have serious adverse consequences on the quality of decisions made using AI [4]. *Measuring inconsistency* [5, 6] is a well-understood approach that can be used towards assessing data quality, as it provides ways to quantify the severity of inconsistency that help understanding the primary sources of conflicts and devising ways to deal with them. In this regard, inconsistency measurement has been extensively investigated for propositional logic (e.g., [7, 8, 9, 10, 11, 12]), and explored in other settings such as software specifications [13] and ontologies [14, 15], among others.

In this paper, we explore inconsistency measures for definite and indefinite databases (DBs). Classical relational DBs can store definite information only, while in practical situations much of the information is not precise. Indefinite DBs, also known as disjunctive DBs, represent disjunctive information in the form of indefinite tuples, i.e., disjunctive facts. They have been studied for a long time [16, 17, 18, 19, 20], and their potential applications include e.g. data integration, extraction and cleaning [21, 22]. Classical relational DBs are a special case of indefinite DBs where the information is definite, i.e., there is no disjunction of tuples.

There are few interesting works addressing the problem of measuring inconsistency in relational DBs. [23] first developed single-dependency axioms for dirtiness functions quantifying inconsistency w.r.t. one functional dependency (FD)—a simple type of denial constraint (DC)—considered in isolation, and proposed a measure that satisfies these axioms. Then a single axiom for dirtiness functions that handle multiple FDs was proposed, although such functions are sup-

SEBD 2023: 31st Symposium on Advanced Database System, July 02–05, 2023, Galzignano Terme, Padua, Italy

* This is an extended abstract of [1].

fparisi@dimes.unical.it (F. Parisi); grant@cs.umd.edu (J. Grant)
0000-0001-9977-1355 (F. Parisi)

posed to be built on top of a dirtiness function for single FDs. The approach in [24, 25, 26] deals with relational databases from the point of view of first-order logic, as in logic programming. Its purpose is to show how database inconsistency measures (IMs) can be applied to integrity checking [27, 28], relaxing repairs, and repair checking, which are applications that also fit within our framework. However, the degrees of inconsistency defined in [26] form a partially ordered set; hence it is not always possible to compare the inconsistency of different DBs. An IM based on an abstract repair semantics is proposed in [29], where the degree of inconsistency depends on the distance between the database instance and the set of possible repairs under a given repair semantics; an instantiation for cardinality-repairs that can be computed via answer-set programs is proposed in [30]. Provenance-informed annotations of the base tuples are used in [31, 32] to characterize the level of inconsistency of data and query results. In particular, building upon the computed annotations, different measures of inconsistency which consider single and multiple violations of DCs are introduced. IMs have been considered as the basis of progress indicators for data-cleaning systems in [33], where properties that account for operational aspects of repair systems are introduced as well as a measure satisfying such properties. Finally, the Shapley value [34] of tuples is investigated in [35] to calculate the contribution of a tuple to inconsistency for inconsistent DBs w.r.t. FDs. The work in [36], which is an early version of [1], as well as all those discussed above focus on definite DBs only. An approach to translate a general information space into an inconsistency equivalent propositional knowledge base is introduced in [37, 38], enabling propositional IMs to be applied also to DBs. However, this approach makes no distinction between DB tuples and integrity constraints as it blames simultaneously tuples and constraints without considering that inconsistency in DBs typically refers to the tuples rather than the integrity constraints.

We explore the problem of tailoring propositional inconsistency measures to indefinite (and definite) DBs, and in particular analyze the computational complexity of the resulting inconsistency measures for both the general case of indefinite DBs and the special case of definite DBs [1]. Our work contributes to understanding how the database counterpart of well-established methods to quantify inconsistency in propositional logic behaves in the relational database context, where data are generally the reason for inconsistency, not the integrity constraints.

2. Indefinite Databases

We assume that the reader is familiar with the relational model and the basic concept of definite DBs [39]. An *indefinite tuple* over relational scheme $R(A_1, \ldots, A_n)$ is a *set* of (definite) tuples over $R(A_1, \ldots, A_n)$. An *indefinite relation instance* (or simply relation) is a finite set of indefinite tuples over a given relation scheme, and an *indefinite DB instance* (database) is a set of indefinite relations over a given DB scheme. Under the model-theoretic approach to relational DBs, an indefinite DB is a set of minimal models [40] (instead of a unique model of the underlying first-order theory as for the case of definite DBs). Under the proof-theoretic approach, an indefinite tuple corresponds to a logical formula with inclusive disjunctions. The information content of an indefinite DB D consists of a set of definite DBs called *possible worlds*. A possible world for D is a set of (definite) tuples that contains a tuple from each element of D and is minimal w.r.t. set inclusion. More formally, let *Def*(D) be the set of all the definite DBs that can be obtained from an indefinite DB D by selecting a (definite) tuple from each indefinite one in D. The meaning of D is given by the set of possible worlds $\mathcal{W}(D) = \{W \mid W \in \mathit{Def}(D), \nexists W'$ such that $W' \in \mathit{Def}(D)$ and $W' \subset W\}$.

We use the terminology *element* to refer to an indefinite tuple of a database D. A definite DB is a special case of an indefinite DB, where each element is a definite tuple and only one possible

	Id	**Name**	**Birth Year**	**Parent**	**Death Year**	
t_1	1	James	1668	Mary	1751	e_1
t_2	1	James	1670	Mary	1751	
t_3	1	Michael	1643	Mary	1600	e_2
t_4	1	Robert	1668	Michael	1600	e_3
t_1	1	James	1668	Mary	1751	
t_5	2	David	1838	Patricia	1905	e_4
t_6	3	Jennifer	1841	Sarah	1923	e_5
t_7	3	Jennifer	1841	Joseph	1923	e_6
t_8	4	Jennifer	1841	Susan	1923	e_7
t_9	4	Jennifer	1841	Jessica	1923	

Table 1
Database D_{ex}, instance of *Ancestor*.

world exists, that is, $\mathcal{W}(D) = \{D\}$.

Example 1. *Consider a genealogical DB whose scheme $\mathcal{DS}_{ex}$ consists of the relation scheme Ancestor (Id, Name, Birth Year, Parent, Death Year), where every record has an id and contains the name, the birth and death year of a person as well as the name of her/his parent. An instance D_{ex} of $\mathcal{DS}_{ex}$ consisting of 7 elements (obtained from 9 different definite tuples) is shown in Table 1. A possible world for D_{ex} is $\{t_1, t_3, t_5, t_6, t_7, t_8\}$, which is obtained from D_{ex} by selecting the tuple t_1 from the elements e_1 and e_3, the tuples t_3, t_5, t_6, and t_7 from the singleton elements e_2, e_4, e_5, and e_6, respectively, and t_8 from element e_7. Let $T = \{t_3, t_5, t_6, t_7\}$, the set of possible worlds for D_{ex} is $\mathcal{W}(D_{ex}) = \{T \cup \{t_1, t_8\}, T \cup \{t_1, t_9\}, T \cup \{t_2, t_4, t_8\}, T \cup \{t_2, t_4, t_9\}\}$.*

It is worth noting that an indefinite DB may contain redundant information because an indefinite tuple is part of another one. For instance, the database $\{\{R(1,1), R(1,2)\}, \{R(1,1)\}\}$ is equivalent to $\{\{R(1,1)\}\}$, that is, they have the same set of possible worlds. Redundancy can be removed by deleting redundant tuples, that is, indefinite tuples that subsume other tuples. This is polynomial in the number of indefinite tuples, assuming that the size of the largest indefinite tuple is a constant (usually a small integer). In the following, we assume that the given DB is not redundant. It is also worth noting that since each indefinite tuple corresponds to a logical formula with inclusive disjunctions, it is possible for more than one tuple within an indefinite tuple to be the real world truth. For instance, for the database $\{\{R(1,1), R(1,2)\}, \{R(1,1), R(1,3)\}, \{R(1,2), R(1,3)\}\}$, a possible world is $\{R(1,1), R(1,2)\}$, which is obtained from the DB by selecting the definite tuple $R(1,1)$ from the first two indefinite tuples and $R(1,2)$ from the third one.

A *denial constraint* (DC) over a database scheme $\mathcal{DS}$ is a first-order sentence of the form: $\forall\, \vec{x}_1, \ldots, \vec{x}_k\ [\neg R_1(\vec{x}_1) \vee \cdots \vee \neg R_k(\vec{x}_k) \vee \varphi(\vec{x}_1, \ldots, \vec{x}_k)]$ where: (*i*) $\forall\, i \in [1..k]$, $\vec{x}_i$ are tuples of variables and $R_i(\vec{x}_i)$ are atoms over $\mathcal{DS}$; 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, 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.

A *functional dependency* (FD) is a DC 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 written as $R : X \rightarrow Y$ (or simply $X \rightarrow Y$), 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 DB scheme $\mathcal{DS}$ and a set $\mathcal{C}$ of integrity constraints over $\mathcal{DS}$, an indefinite DB instance D of $\mathcal{DS}$ is said to be *consistent* w.r.t. $\mathcal{C}$ (denoted as $D \models \mathcal{C}$) iff there is at least one possible world of D which is consistent w.r.t. $\mathcal{C}$ (in the standard model-theoretic sense), that is, $\{W \mid W \in \mathcal{W}(D), W \models \mathcal{C}\} \neq \emptyset$; otherwise, D is said to be *inconsistent* (w.r.t. $\mathcal{C}$).

Example 2. *Continuing from Example 1, let $\mathcal{C}_{ex}$ be the set of the following DCs:*
• $c_1 = [\neg Ancestor(x_1, x_2, x_3, x_4, x_5) \vee x_5 > x_3]$*, stating that the death year must be greater than the birth year.*
• $c_2 = [\neg Ancestor(x_1, x_2, x_3, x_4, x_5) \vee \neg Ancestor(x_1, x_6, x_7, x_8, x_9) \vee x_2 = x_6]$*, that is the FD Id→Name.*
• $c_3 = [\neg Ancestor(x_1, x_2, x_3, x_4, x_5) \vee \neg Ancestor(x_6, x_2, x_7, x_8, x_9) \vee \neg Ancestor(x_{10}, x_2, x_{11}, x_{12}, x_{13}) \vee x_4 = x_8 \vee x_4 = x_{12} \vee x_8 = x_{12}]$*, that is the numerical dependency [41] Name$\rightarrow^2$Parent stating that for every person there can be at most 2 parents.*

We have that D_{ex} is inconsistent w.r.t. $\mathcal{C}_{ex}$. In particular, going through the constraints we find that 1) $e_2 \not\models c_1$, while e.g. $e_3 \models c_1$ as one of its two tuples (disjuncts) satisfies the constraint; 2) the pairs of elements e_1, e_2 and e_2, e_3 are inconsistent with c_2 (notice that $\{e_1, e_3\} \models c_2$ as there is a world consisting of their common tuple t_1 which is consistent); 3) the three elements e_5, e_6, and e_7 together are inconsistent with c_3. None of the possible worlds for D_{ex} is consistent w.r.t. $\mathcal{C}_{ex}$, as each one violates one or more constraints (e.g., c_1 is violated by all of them).

3. Database Inconsistency Measures

The idea of an inconsistency measure is to assign a nonnegative number to a knowledge base that measures its inconsistency [42]. We introduce inconsistency measures (IMs) for indefinite DBs with denial constraints. We use **D** to denote the set of all indefinite database instances over a fixed but arbitrary database scheme $\mathcal{DS}$. In general, we will omit the database scheme and the set $\mathcal{C}$ of integrity constraints in the terminology.

Definition 1 (Inconsistency Measure)**.** *A function $\mathcal{I} : \boldsymbol{D} \rightarrow \mathbb{R}^{\geq 0}_{\infty}$ is an* **inconsistency measure** *iff the following two conditions hold for all $D, D' \in \boldsymbol{D}$:*
1) ***Consistency*** *$\mathcal{I}(D) = 0$ iff D is consistent; 2)* ***Monotony*** *If $D \subseteq D'$, then $\mathcal{I}(D) \leq \mathcal{I}(D')$.*

Consistency and Monotony are called (rationality) postulates. 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 DBs get measure 0. Monotony means that the enlargement of a DB cannot decrease its measure. Note that we are limiting the integrity constraints to DCs. This means that inconsistencies cannot be resolved by insertions. This is not the case in the presence of existential constraints, such as inclusion dependencies, where Monotony is not appropriate. Additional postulates are given in [1], where satisfaction for definite and indefinite DBs is studied.

We now give some basic definitions needed to define IMs. A **minimal inconsistent subset** (MIS) of D is a set of elements $X \subseteq D$ such that X is inconsistent (w.r.t. $\mathcal{C}$) 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 elements 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 element that occurs in a MIS is **problematic**; otherwise it is **free**. We use $\mathsf{Problematic}(D)$ and $\mathsf{Free}(D)$ to denote the sets of problematic and free elements of D. An element e is **contradictory** if $\{e\}$ is inconsistent w.r.t. $\mathcal{C}$. We write $\mathsf{Contradictory}(D)$ for the set of contradictory elements.

3.1. Measures using Minimal Inconsistent Subsets

We start with the measures that rely on MISs and the related concepts defined above.

Definition 2 (Database IMs)**.** *For any DB D, the IMs $\mathcal{I}_B$, $\mathcal{I}_M$, $\mathcal{I}_P$, $\mathcal{I}_A$, and $\mathcal{I}_H$ are such that*

- *$\mathcal{I}_B(D) = 1$ if D is inconsistent and $\mathcal{I}_B(D) = 0$ if D is consistent.*

- $\mathcal{I}_M(D) = |\mathsf{MI}(D)|$.
- $\mathcal{I}_P(D) = |\mathsf{Problematic}(D)|$.
- $\mathcal{I}_A(D) = (|\mathsf{MC}(D)| + |\mathsf{Contradictory}(D)|) - 1$.
- $\mathcal{I}_H(D) = \min\{|X| \ s.t.\ X \subseteq D \textit{ and } \forall M \in \mathsf{MI}(D),\ X \cap M \neq \emptyset\}$.

We explain the measures as follows. $\mathcal{I}_B$ is also called the drastic measure [43]: 0 means consistent; 1 means inconsistent. $\mathcal{I}_M$ counts the number of MISs [43]. The rationale is that a MIS represents a minimal inconsistency for a set of database elements; hence this measure counts the number of such inconsistencies. $\mathcal{I}_P$ counts the number of elements that are in one or more minimal inconsistencies [44]. $\mathcal{I}_A$ uses the cardinality of the set of maximal consistent subsets [44] (corresponding to repairs for DBs [45]). Intuitively, the larger this set, the larger is the space of different ways to get consistency, the higher the degree of inconsistency. Contradictory elements are added as they do not appear in any way in a maximal consistent set; then 1 must be subtracted to obtain $\mathcal{I}_A(D) = 0$ for a consistent D because every consistent DB has a maximal consistent subset, namely D itself. $\mathcal{I}_H$ counts the minimal number of elements whose deletion makes the DB consistent [46]. Hence $\mathcal{I}_H$ can be written as $\mathcal{I}_H = \min\{|X| \ s.t.\ D \setminus X$ is consistent $\}$. In fact, both $\mathcal{I}_A$ and $\mathcal{I}_H$ link the inconsistency measurement to the ways of restoring consistency, an idea explored for definite DBs in [29, 30] where the degree of inconsistency depends on the distance between the DB and the set of possible repairs under a given repair semantics.

Example 3. *Continuing with our running example,* $\mathsf{MI}(D_{ex}) = \{\{e_2\}, \{e_5, e_6, e_7\}\}$. *Note that* $\{e_1, e_2\}$ *as well as* $\{e_2, e_3\}$ *are not included because they contain* $\{e_2\}$. *Thus there are* 4 *problematic elements in* D_{ex}*, and* 3 *free elements. Also,* $\mathsf{MC}(D_{ex}) = \{\{e_1, e_3, e_4, e_5, e_6\}, \{e_1, e_3, e_4, e_5, e_7\}, \{e_1, e_3, e_4, e_6, e_7\}\}$. *Therefore, the values of the IMs for* D_{ex} *are as follows.* $\mathcal{I}_B(D_{ex}) = 1$ *as the database is inconsistent.* $\mathcal{I}_M(D_{ex}) = 2$ *as there are* 2 *MISs.* $\mathcal{I}_P(D_{ex}) = 4$ *as* 4 *elements are in* $\mathsf{MI}(D_{ex})$. $\mathcal{I}_A(D_{ex}) = 3$ *as there are* 3 *maximal consistent subsets and one contradictory element (that is,* e_2*). Finally,* $\mathcal{I}_H(D_{ex}) = 2$ *as no less than* 2 *elements intersect with each MIS.*

3.2. A Measure Using 3VL

We now consider the Contension measure [44], which uses a three-valued (3VL) logic. 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*). The logical connectives are extended to 3VL interpretations using Priest's three-valued logic, the Logic of Paradox [47]. This interpretation uses an ordering on the truth values where $F < B < T$ and $\wedge$ computes the minimum value while $\vee$ computes the maximum value; also $\neg(B) = B$. So, for example, $B \wedge F = F$ and $B \vee F = B$. In classical two-valued logic, an interpretation is a *model* for a set of formulas if every formula gets the value T (the unique designated value). But in 3VL there are two designated values, T and B. This means that for a given DB D with a set $\mathcal{C}$ of constraints, a 3VL interpretation is a 3VL model iff all the integrity constraints and elements get the value T or B. We use $\mathsf{Models}(D)$ to denote the set of 3VL models for D (with the 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 atoms that have truth value B.

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

For our running example, we have that $\mathcal{I}_C(D_{ex}) = 2$ as the minimal number of B values for an interpretation occurs when assigning B to the tuple in e_2 and a tuple in either e_5 or e_6.

For definite DBs, it can be shown that $\mathcal{I}_C(D) = \mathcal{I}_H(D)$. No other pair of measures considered in this paper is identical for definite DBs. For indefinite DBs, $\mathcal{I}_H$ and $\mathcal{I}_C$ need not give the same result. For instance, $D' = \{\{R(-1), R(-2)\}, \{R(-1), R(-3)\}\}$ and $\mathcal{C}' = \{\neg R(x) \vee x > 0\}$., we have that $\mathcal{I}_H(D') = 2 > \mathcal{I}_C(D') = 1$ because both elements in D' are self-contradictions, while for 3VL it suffices to assign $R(-1)$ the value B and both $R(-2)$ and $R(-3)$ the value F to eliminate the inconsistency. In general, $\mathcal{I}_C(D) \leq \mathcal{I}_H(D)$ holds for all indefinite DBs because assigning B to one of the disjuncts in an element that needs removal for $\mathcal{I}_H$ suffices to eliminate the inconsistency for $\mathcal{I}_C$ as well.

3.3. A Probabilistic Measure

Finally, we define the database counterpart of the probabilistic measure I_η that uses the **PSAT (probabilistic satisfiability)** concept [48]. 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; hence $0 \leq p_i \leq 1$ for $1 \leq i \leq m$. A *probability distribution* 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 distribution 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 ϕ in the knowledge base. A PSAT instance is *satisfiable* if there is a probability distribution 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 DB as a PSAT instance, where every element in the DB is assigned probability η, and every (ground) integrity constraint is assigned a probability 1. Thus, given a DB D, a set of integrity constraints $\mathcal{C}$, and a probability threshold $\eta \in [0, 1]$, we define the PSAT instance $\Gamma_{\mathcal{C},\eta}(D) = \{P(e) \geq \eta \mid e \in D\} \cup \{P(g(c)) = 1 \mid g(c)$ is a ground constraint for $c \in \mathcal{C}\}$, which enables the following definition.

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

Thus, $\mathcal{I}_\eta(D)$ is one minus the maximum probability lower bound one can consistently assign to all elements in D. For the database of our running example we have that $\mathcal{I}_\eta(D_{ex}) = 1$ because the maximum probability that can be assigned to (the contradictory) element e_2 is zero.

4. Complexity of Database Inconsistency Measures

We investigate the data-complexity 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 $\mathcal{I}_\eta$ (as shown in [48]).

Definition 5 (Lower (**LV**), Upper (**UV**), and Exact Value (**EV**) problems)**.** *Let $\mathcal{I}$ be an IM. Given a DB D over a fixed database scheme with a fixed set of constraints, and a positive value $v \in \mathbb{Q}^{>0}$, **LV**$_\mathcal{I}(D, v)$ is the problem of deciding whether $\mathcal{I}(D) \geq v$. Given D and a non-negative value $v' \in \mathbb{Q}^{\geq 0}$, **UV**$_\mathcal{I}(D, v')$ is the problem of deciding whether $\mathcal{I}(D) \leq v'$, and **EV**$_\mathcal{I}(D, v')$ is the problem of deciding whether $\mathcal{I}(D) = v'$.*

We also consider the problem of determining the IM value.

Definition 6 (Inconsistency Measurement (**IM**) problem)**.** *Let $\mathcal{I}$ be an inconsistency measure. Given a DB D over a fixed database scheme with a fixed set of constraints, **IM**$_\mathcal{I}(D)$ is the problem of computing the value of $\mathcal{I}(D)$.*

	LV$_{\mathcal{I}}(D,v)$		**UV**$_{\mathcal{I}}(D,v)$		**EV**$_{\mathcal{I}}(D,v)$		**IM**$_{\mathcal{I}}(D)$	
	def.	indefinite	def.	indefinite	def.	indefinite	def.	indefinite
$\mathcal{I}_B$	P	$coNP$-c	P	NP-c	P	D^p	FP	FNP
$\mathcal{I}_M$	P	$coNP$-h, CNP	P	NP-h, CNP	P	D^p-h, $C_{=}D^p$	FP	$\# \cdot coNP$
$\mathcal{I}_P$	P	Σ_2^p-c	P	Π_2^p-c	P	D_2^p-c	FP	$FP^{\Sigma_2^p[log\ n]}$
$\mathcal{I}_A$	CP	CNP	CP	CNP	CP	$C_{=}\mathcal{B}(NP)$	$\#P$-c	$\#P$-h, $\# \cdot coNP$
$\mathcal{I}_H$	$coNP$-c		NP-c		D^p-c		$FP^{NP[log\ n]}$-c	
$\mathcal{I}_C$	$coNP$-c		NP-c		D^p-c		$FP^{NP[log\ n]}$-c	
$\mathcal{I}_\eta$	$coNP$-c		NP-c		D^p		FP^{NP}	

Table 2
Complexity of Lower Value (**LV**), Upper Value (**UV**), Exact Value (**EV**), and Inconsistency Measurement (**IM**) problems for definite and indefinite databases. For a complexity class $\mathcal{C}$, $\mathcal{C}$-c (resp., $\mathcal{C}$-h) means $\mathcal{C}$-complete (resp., $\mathcal{C}$-hard); only $\mathcal{C}$ means membership in $\mathcal{C}$. For two classes $\mathcal{C}$, $\mathcal{C}'$, the separation by a comma means $\mathcal{C}$-hard and in $\mathcal{C}'$.

A summary of the complexity results obtained for the above-mentioned problems is given in Table 2. We found that while $\mathcal{I}_B$, $\mathcal{I}_M$, and $\mathcal{I}_P$ become tractable for definite DBs and the complexity of $\mathcal{I}_A$ decreases, $\mathcal{I}_H$, $\mathcal{I}_C$ and $\mathcal{I}_\eta$ remain as hard as in the propositional case [10] even under data complexity for both definite and indefinite DBs. Moreover, while $\mathcal{I}_B$, $\mathcal{I}_M$, and $\mathcal{I}_P$ are tractable for definite DBs, for indefinite DBs they are intractable. Specifically, the complexity of **LV**, **UV**, and **EV** for $\mathcal{I}_B$ and $\mathcal{I}_P$ is in the first and second level of the polynomial hierarchy [49], respectively, while that for $\mathcal{I}_M$ relies on classes from the counting polynomial hierarchy [50] (as that for $\mathcal{I}_A$)—we briefly recall complexity classes in Appendix A. Except for the measures $\mathcal{I}_B$, $\mathcal{I}_M$, and $\mathcal{I}_P$ that are tractable in the case of definite DBs, for indefinite DBs the complexity of the function problem **IM** ranges from being in the classes FNP for $\mathcal{I}_B$, $FP^{NP[log\ n]}$ for $\mathcal{I}_H$ and $\mathcal{I}_C$, FP^{NP} for $\mathcal{I}_\eta$, and $FP^{\Sigma_2^p[log\ n]}$ for $\mathcal{I}_P$ to the counting class $\# \cdot coNP$ [51], which includes $\#P$ [52], for $\mathcal{I}_M$ and $\mathcal{I}_A$.

It is worth noting that almost all the results in Table 2 hold even if the set of integrity constraints consists of FDs only. On the one hand, all the membership results trivially hold for FDs since they have been shown to hold for DCs. On the other hand, all the hardness results except the D_2^p-hardness for $\mathbf{EV}_{\mathcal{I}_P}$, the D^p-hardness for $\mathbf{EV}_{\mathcal{I}_H}$, and the $FP^{NP[log\ n]}$-hardness for $\mathbf{IM}_{\mathcal{I}_H}$ can be shown to hold for FDs [1].

5. Conclusions and Future Work

We have introduced a framework for measuring inconsistency in DBs that relies on *absolute* IMs, measuring by some criteria the total amount of inconsistency. In contrast, *relative* IMs recently explored in [53] provide a ratio of the amount of inconsistency w.r.t. some parameter, e.g. the size of the DB. Many interesting issues concerning IMs in DBs remain unexplored. 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 DBs, and devise efficient algorithms for evaluating IMs. In this regard, a dichotomy for FDs for the problem of computing the cost of a cardinality repair (that is equivalent to computing the value of measure $\mathcal{I}_H$) for definite DBs has been recently presented in [54]. In fact, the polynomial-time 2-approximation given in [54] entails an approximability result for the problem $\mathbf{IM}_{\mathcal{I}_H}$. The IMs we have considered work at the tuple-level, without distinguishing inconsistencies arising from different (sets of) attributes, which is another issue we want to address in the future by following the idea of dimensional inconsistency measures proposed in [55] for spatio-temporal databases, where the dimensions considered are those concerning space, time, and (moving) objects [56, 57]. Finally, as indefinite DBs considered in this paper allow the representation of a form of incomplete information (disjunctive information), another interesting direction for future work is considering other forms of incomplete information such as maybe information [17] as well as dealing with databases with null values [16, 58].

A. Appendix: Complexity Classes

The classical classes Σ^p_k, Π^p_k and Δ^p_k, with $k \geq 0$, are defined as follows [49]: *i)* $\Sigma^p_0 = \Pi^p_0 = \Delta^p_0 = P$; *ii)* $\Sigma^p_1 = NP$ and $\Pi^p_1 = coNP$; *iii)* $\Delta^p_k = P^{\Sigma^p_{k-1}}$, $\Sigma^p_k = NP^{\Sigma^p_{k-1}}$, and $\Pi^p_k = co\Sigma^p_k$, $\forall k > 0$. Thus, $P^{\mathcal{C}}$ (resp., $NP^{\mathcal{C}}$) denotes the class of the decision problems that can be solved in polynomial time by using an oracle in the class $\mathcal{C}$ by a deterministic (resp., non-deterministic) Turing machine. It holds that $\Sigma^p_k \subseteq \Delta^p_{k+1} \subseteq \Sigma^p_{k+1} \subseteq PSPACE$ and $\Pi^p_k \subseteq \Delta^p_{k+1} \subseteq \Pi^p_{k+1} \subseteq PSPACE$. Moreover, a decision problem is in D^p_k iff it is the conjunction of a problem in Σ^p_k and a problem in Π^p_k. D^p_1 is also denoted as D^p. It holds that $D^p \subseteq \Delta^p_2$.

We also use the classes CP [59] and CNP from the counting polynomial hierarchy defined in [50]. These classes rely on a counting quantifier C defined as follows. Given a predicate $H(x, y)$ with free variables x and y, $C^k_y H(x, y)$ holds iff $|\{y : H(x, y) \text{ is true}\}| \geq k$, i.e., the counting quantifier is true for predicate H and bound k iff the number of values of y such that $H(x, y)$ holds is at least k. The polynomially bounded version of the counting quantifier is defined as follows. Given a class $\mathcal{C}$ of decision problems, we say that a problem A is in $C\mathcal{C}$ iff there is a problem $B \in \mathcal{C}$, a polynomial-time computable function f, and a polynomial p such that x is a positive instance of A iff $C^{f(x)}_{y,\,|y| \leq p(x)}(x, y) \in B$. That is, instance $x \in A$ iff there are at least $f(x)$ many y's whose size is polynomially bounded by that of x such that a predicate for (x, y) holds, with checking the predicate being in B. The class CP coincides with the class PP of the decision problems that can be solved in polynomial time by a probabilistic Turing machine [59, 50]. The relationships between the classes NP, CP and $coNP$ are as follows: $NP \subseteq CP$ and $coNP \subseteq CP$. Differently from NP that is closed under union and intersection and is not known to be closed under complement, the class CP is closed under complement. Like NP, CP is closed under union and intersection, though this question remained open for several years [60]. Like CP, CNP is closed under complement. It holds that $CP \subseteq CNP$. Moreover, $CNP = C\mathcal{B}(NP)$, where $\mathcal{B}(NP)$ is the Boolean closure of NP [50], that implies that $D^p \subseteq \mathcal{B}(NP)$ and $CNP = CD^p$. The class $C_{=}\mathcal{C}$ is defined exactly as $C\mathcal{C}$ except that the counting quantifier holds iff it is satisfied by equality.

FP (resp., FNP) is the class of the function problems that can be solved by a deterministic (resp., non-deterministic) Turing machine in polynomial time. $FP^{\mathcal{C}}$ is the class of functions computable by a deterministic polynomial-time Turing machine using a $\mathcal{C}$-oracle. Thus, FP^{NP} (resp., $FP^{\Sigma^p_2}$) 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 (resp., Σ^p_2 oracle). If a logarithmic number of queries is asked by the machine, then we have the class $FP^{NP[\log n]}$ (resp., $FP^{\Sigma^p_2[\log n]}$).

Finally, given a class $\mathcal{C}$ of decision problems, $\# \cdot \mathcal{C}$ is the class of the counting problems defined by means of witness functions w that assign to a given input x a set $w(x)$ of witnesses. Herein, a counting problem returns the cardinality $|w(x)|$ of a set of witnesses. For the witness function w, *(i)* for every input x, the size of every witness $y \in w(x)$ is polynomially bounded by that of x; and *(ii)* given x and y, deciding whether $y \in w(x)$ is in class $\mathcal{C}$. A canonical problem for $\# \cdot P$ is counting the satisfying assignments of a SAT formula. This problem is in $\#P$ [52], which coincides with $\# \cdot P$. It holds that $\# \cdot P \subseteq \# \cdot coNP$ and $\# \cdot coNP = \# \cdot \Delta^p_2$.

Acknowledgments

The fist author acknowledges financial support from PNRR MUR project PE0000013-FAIR.

References

[1] F. Parisi, J. Grant, On measuring inconsistency in definite and indefinite databases with denial constraints, Artificial Intelligence 318 (2023) 103884.

[2] A. Y. Levy, Combining artificial intelligence and databases for data integration, in: Artificial Intelligence Today: Recent Trends and Developments, Springer, 1999, pp. 249–268.

[3] A. Jain, H. Patel, L. Nagalapatti, N. Gupta, S. Mehta, S. C. Guttula, S. Mujumdar, S. Afzal, R. S. Mittal, V. Munigala, Overview and importance of data quality for machine learning tasks, in: KDD, ACM, 2020, pp. 3561–3562.

[4] I. H. Sarker, Data science and analytics: An overview from data-driven smart computing, decision-making and applications perspective, SN Comput. Sci. 2 (2021) 377.

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

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

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

[8] K. Mu, Measuring inconsistency with constraints for propositional knowledge bases, Artif. Intell. 259 (2018) 52–90.

[9] G. D. Bona, J. Grant, A. Hunter, S. Konieczny, Classifying inconsistency measures using graphs, J. Artif. Intell. Res. 66 (2019) 937–987.

[10] M. Thimm, J. P. Wallner, On the complexity of inconsistency measurement, Artif. Intell. 275 (2019) 411–456.

[11] P. Besnard, J. Grant, Relative inconsistency measures, Artif. Intell. 280 (2020) 103231. doi:https://doi.org/10.1016/j.artint.2019.103231.

[12] M. Ulbricht, M. Thimm, G. Brewka, Handling and measuring inconsistency in non-monotonic logics, Artif. Intell. 286 (2020) 103344.

[13] K. Mu, Z. Jin, R. Lu, W. Liu, Measuring inconsistency in requirements specifications, in: Proc. of European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU), 2005, pp. 440–451.

[14] L. Zhou, H. Huang, G. Qi, Y. Ma, Z. Huang, Y. Qu, Measuring inconsistency in DL-Lite ontologies, in: Proc. of International Conference on Web Intelligence (WI), 2009, pp. 349–356.

[15] X. Zhang, K. Wang, Z. Wang, Y. Ma, G. Qi, Z. Feng, A distance-based framework for inconsistency-tolerant reasoning and inconsistency measurement in DL-Lite, Int. J. Approx. Reasoning 89 (2017) 58–79.

[16] J. Grant, J. Minker, Answering queries in indefinite databases and the null value problem, Adv. Comput. Res. 3 (1986) 247–267.

[17] K. C. Liu, R. Sunderraman, Indefinite and maybe information in relational databases, ACM Trans. Database Syst. 15 (1990) 1–39.

[18] T. Imielinski, R. van der Meyden, K. V. Vadaparty, Complexity tailored design: A new design methodology for databases with incomplete information, J. Comput. Syst. Sci. 51 (1995) 405–432.

[19] J. Minker, D. Seipel, Disjunctive logic programming: A survey and assessment, in:

Computational Logic: Logic Programming and Beyond, Essays in Honour of Robert A. Kowalski, Part I, 2002, pp. 472–511.

[20] M. Alviano, W. Faber, N. Leone, S. Perri, G. Pfeifer, G. Terracina, The disjunctive datalog system DLV, in: Proc. of International Workshop on Datalog, 2010, pp. 282–301.

[21] O. Benjelloun, A. D. Sarma, A. Y. Halevy, M. Theobald, J. Widom, Databases with uncertainty and lineage, VLDB J. 17 (2008) 243–264.

[22] C. Molinaro, J. Chomicki, J. Marcinkowski, Disjunctive databases for representing repairs, Ann. Math. Artif. Intell. 57 (2009) 103–124.

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

[24] H. Decker, Inconsistency-tolerant database repairs and simplified repair checking by measure-based integrity checking, T. Large-Scale Data- and Knowledge-Centered Systems 34 (2017) 153–183.

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

[26] H. Decker, Measuring database inconsistency, in: J. Grant, M. V. Martinez (Eds.), Measuring Inconsistency in Information, College Publications, 2018, pp. 271–311.

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

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

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

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

[31] O. Issa, A. Bonifati, F. Toumani, Evaluating top-k queries with inconsistency degrees, Proc. VLDB Endow. 13 (2020) 2146–2158.

[32] O. Issa, A. Bonifati, F. Toumani, INCA: inconsistency-aware data profiling and querying, in: Proc. of International Conference on Management of Data (SIGMOD), 2021, pp. 2745–2749.

[33] E. Livshits, R. Kochirgan, S. Tsur, I. F. Ilyas, B. Kimelfeld, S. Roy, Properties of inconsistency measures for databases, in: Proc. of International Conference on Management of Data (SIGMOD), 2021, pp. 1182–1194.

[34] K. Hausken, M. Mohr, The value of a player in n-person games, Soc. Choice Welf. 18 (2001) 465–483.

[35] E. Livshits, B. Kimelfeld, The shapley value of inconsistency measures for functional dependencies, in: Proc. of International Conference on Database Theory (ICDT), volume 186, 2021, pp. 15:1–15:19.

[36] F. Parisi, J. Grant, On measuring inconsistency in relational databases with denial constraints, in: Proc. of European Conference on Artificial Intelligence (ECAI), 2020, pp. 857–864.

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

[38] J. Grant, F. Parisi, General information spaces: measuring inconsistency, rationality postu-

lates, and complexity, Ann. Math. Artif. Intell. 90 (2022) 235–269.

[39] S. Abiteboul, R. Hull, V. Vianu, Foundations of Databases, Addison-Wesley, 1995.

[40] J. Minker, On indefinite databases and the closed world assumption, in: Proc. of the 6th Conference on Automated Deduction, 1982, pp. 292–308.

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

[42] M. Thimm, On the evaluation of inconsistency measures, in: J. Grant, M. V. Martinez (Eds.), Measuring Inconsistency in Information, College Publications, 2018, pp. 19–60.

[43] A. Hunter, S. Konieczny, Measuring inconsistency through minimal inconsistent sets, in: Proc. of International Conference on Principles of Knowledge Representation and Reasoning (KR), 2008, pp. 358–366.

[44] J. Grant, 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), 2011, pp. 362–373.

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

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

[47] G. Priest, Logic of paradox, Journal of Philosophical Logic 8 (1979) 219–241.

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

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

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

[51] L. A. Hemaspaandra, H. Vollmer, The satanic notations: counting classes beyond #P and other definitional adventures, SIGACT News 26 (1995) 2–13.

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

[53] F. Parisi, J. Grant, Relative inconsistency measures for indefinite databases with denial constraints, in: Proc. of International Joint Conference on Artificial Intelligence (IJCAI), 2023.

[54] E. Livshits, B. Kimelfeld, S. Roy, Computing optimal repairs for functional dependencies, ACM Trans. Database Syst. 45 (2020) 4:1–4:46.

[55] J. Grant, M. V. Martinez, C. Molinaro, F. Parisi, Dimensional inconsistency measures and postulates in spatio-temporal databases, J. Artif. Intell. Res. 71 (2021) 733–780.

[56] J. Grant, F. Parisi, A. Parker, V. S. Subrahmanian, An agm-style belief revision mechanism for probabilistic spatio-temporal logics, Artif. Intell. 174 (2010) 72–104.

[57] F. Parisi, J. Grant, On repairing and querying inconsistent probabilistic spatio-temporal databases, Int. J. Approx. Reason. 84 (2017) 41–74.

[58] R. van der Meyden, Logical approaches to incomplete information: A survey, in: Logics for Databases and Information Systems (the book grow out of the Dagstuhl Seminar 9529: Role of Logics in Information Systems, 1995), 1998, pp. 307–356.

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

[60] R. Beigel, N. Reingold, D. A. Spielman, PP is closed under intersection, J. Comput. Syst. Sci. 50 (1995) 191–202.