Relative Inconsistency Measures for Indefinite Databases with Denial Constraints

Francesco Parisi[1] , **John Grant**[2]
[1]Department of Informatics, Modeling, Electronics and System Engineering
University of Calabria, Italy
[2]Department of Computer Science and UMIACS
University of Maryland, College Park, MD, USA
fparisi@dimes.unical.it, grant@cs.umd.edu

Abstract

Handling conflicting information is an important challenge in AI. Measuring inconsistency is an approach that provides ways to quantify the severity of inconsistency and helps understanding the primary sources of conflicts. In particular, a relative inconsistency measure computes, by some criteria, the proportion of the knowledge base that is inconsistent. In this paper we investigate relative inconsistency measures for indefinite databases, which allow for indefinite or partial information which is formally expressed by means of disjunctive tuples. We introduce a postulate-based definition of relative inconsistency measure for indefinite databases with denial constraints, and investigate the compliance of some relative inconsistency measures with rationality postulates for indefinite databases as well as for the special case of definite databases. Finally, we investigate the complexity of the problem of computing the value of the proposed relative 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.

1 Introduction

Handling conflicting information is an important challenge in AI. Data of poor quality can significantly limit the implementation of effective AI solutions [Levy, 1999; Jain *et al.*, 2020]. So, having information on the quality of data used in machine learning and data-driven approaches is crucial, as poor quality data can have serious adverse consequences on the quality of decisions made using AI [Sarker, 2021]. *Measuring inconsistency* [Grant, 1978; Grant and Martinez, 2018] 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 knowledge bases (e.g., [Thimm, 2016; Mu, 2018; Bona *et al.*, 2019; Thimm and Wallner, 2019; Besnard and Grant, 2020; Ulbricht *et al.*, 2020]), and explored in other settings such as software specifications [Mu *et al.*, 2005], databases [Martinez *et al.*, 2007; Decker, 2017; Bertossi, 2019; Parisi and Grant, 2020; Grant *et al.*, 2021], and ontologies [Zhou *et al.*, 2009; Zhang *et al.*, 2017], among others. However, most of the literature on inconsistency measures (IMs) focused on *absolute* measures, rather than *relative* measures. An absolute measure gives the total amount of inconsistency, while a relative measure computes, by some criteria, the proportion of the base that is inconsistent [Besnard and Grant, 2020]. The difference between the two types of measures and the motivation for considering relative measures can be explained as follows. Consider the case of a large database D with numerous inconsistencies and a much smaller database D' that has almost as many inconsistencies as D. The absolute inconsistency of D is greater than the absolute inconsistency of D' just because it has more inconsistencies. But we can consider D' to be more inconsistent than D since relative to its size it has more inconsistencies than D. In fact, if the question is "How inconsistent is the database?" the answer would likely be something like "It's $x\%$ inconsistent", rather than "It has y inconsistencies". Relative inconsistency measures have been used in applications including [Zhou *et al.*, 2009] for ontologies, [Costa and Martins, 2018] for a hospital admission system, and [Fan *et al.*, 2019] for a financial system database.

In this paper, we explore relative inconsistency measures for indefinite databases. The relational model is arguably among the most common types of data models and relational databases (DBs) are often used to store real-world data consumed by AI applications. 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 [Grant and Minker, 1986; Liu and Sunderraman, 1990; Imielinski *et al.*, 1995; Minker and Seipel, 2002; Alviano *et al.*, 2010], and their potential applications include e.g. data integration, extraction and cleaning [Benjelloun *et al.*, 2008; Molinaro *et al.*, 2009]. 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. [Martinez *et al.*, 2007] first developed single-dependency axioms for dirtiness functions quantifying inconsistency w.r.t. one functional de-

pendency (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 supposed to be built on top of a dirtiness function for single FDs. The approach in [Decker, 2017; Decker and Misra, 2017; Decker, 2018] 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 [Decker and Martinenghi, 2008; 2011], relaxing repairs, and repair checking, which are applications that also fit within our framework. However, the degrees of inconsistency defined in [Decker, 2018] 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 [Bertossi, 2018], 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 [Bertossi, 2019]. Rationality postulate compliance and the complexity of some absolute measures for DBs inspired from those for propositional logic are investigated in [Parisi and Grant, 2020]. Provenance-informed annotations of the base tuples are used in [Issa *et al.*, 2020; 2021] 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 denial constraints are introduced. IMs have been considered as the basis of progress indicators for data-cleaning systems in [Livshits *et al.*, 2021], where properties that account for operational aspects of repair systems are introduced as well as a measure satisfying such properties. Finally, the Shapley value [Hausken and Mohr, 2001] of DB tuples is investigated in [Livshits and Kimelfeld, 2021] to calculate the contribution of a tuple to inconsistency for inconsistent DBs w.r.t. FDs. All the above-mentioned works focus on definite DBs. Absolute IMs for indefinite DBs have been explored in [Parisi and Grant, 2023].

Contribution. Our main contributions are as follows.

• We introduce four postulates and a postulate-based definition of the concept of relative inconsistency measure (IM) for indefinite DBs with denial constraints, a class of integrity constraints that includes, for instance, equality generating dependencies [Beeri and Vardi, 1984] and numerical dependencies [Grant and Minker, 1985]. We consider five relative IMs, namely $\mathcal{I}_{mv}$, $\mathcal{I}_M^r$, $\mathcal{I}_P^r$, $\mathcal{I}_H^r$, and $\mathcal{I}_C^r$ (which are formally defined in Section 3), corresponding to different methods of quantifying inconsistency in indefinite DBs.

• Analyzing the compliance of IMs w.r.t. rationality postulates is an important way to evaluate a measure, and is one of the main problems investigated in the area of inconsistency measurement, even though the set of desirable postulates is not universally accepted [Besnard, 2014; Thimm, 2016; 2018] (some postulates proposed in the literature are even incompatible, as they express alternative properties that may be required in different contexts). Given the importance of postulate satisfaction, in addition to the postulates considered in the definition of relative IM, we examine the compliance of definite and indefinite DBs w.r.t. additional rationality postulates (Section 4). Our results are shown in Table 2. Two of the measures ($\mathcal{I}_P^r$ and $\mathcal{I}_H^r$) satisfy 7 of the 8 postulates even for indefinite DBs. As two of the considered postulates are incompatible (MI-Normalization and Contradiction can be satisfied together only for a very specific form of integrity constraints), $\mathcal{I}_P^r$ and $\mathcal{I}_H^r$ satisfy as many postulates in the list as possible. For another two measures ($\mathcal{I}_{mv}$ and $\mathcal{I}_C^r$), the satisfaction results are better for the definite case (in which case $\mathcal{I}_{mv}$ and $\mathcal{I}_C^r$ satisfy as many postulates as possible), but in all cases the majority of the postulates are satisfied.

• 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 relative IM for indefinite and definite DBs (Section 5). The complexity results obtained for these decision problems and for the function problem of computing the value of a relative IM (**IM** problem) are given in Table 3. It turns out that, for indefinite DBs, the considered relative IMs exhibit different levels of intractability. Specifically, for the decision problems (**LV**, **UV**, and **EV**), the complexity ranges from the first level of the polynomial hierarchy [Papadimitriou, 1994] for $\mathcal{I}_H^r$ and $\mathcal{I}_C^r$, to the second level of the hierarchy for $\mathcal{I}_{mv}$ and $\mathcal{I}_P^r$, and up to classes from the counting polynomial hierarchy [Wagner, 1986] for $\mathcal{I}_M^r$. However, three measures ($\mathcal{I}_{mv}$, $\mathcal{I}_P^r$, and $\mathcal{I}_M^r$) turn out to be tractable in the case of definite DBs, while for the other two measures ($\mathcal{I}_H^r$ and $\mathcal{I}_C^r$) restricting to the definite case has no impact on the complexity. Finally, the complexity of the function problem **IM** ranges from being in FP for $\mathcal{I}_{mv}$, $\mathcal{I}_P^r$, and $\mathcal{I}_M^r$ for definite DBs; in $FP^{NP[\log n]}$ for $\mathcal{I}_H^r$ and $\mathcal{I}_C^r$ for both definite and indefinite DBs; in $FP^{\Sigma_2^p[\log n]}$ for $\mathcal{I}_{mv}$ and $\mathcal{I}_P^r$ for indefinite DBs; and in the counting class $\# \cdot coNP$ [Hemaspaandra and Vollmer, 1995] for $\mathcal{I}_M^r$ and indefinite DBs.

2 Preliminaries

We recall indefinite DBs and some complexity classes.

2.1 Indefinite Databases

We assume that the reader is familiar with the relational model and the basic concept of definite DBs. 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 [Minker, 1982] (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

	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*.

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. Note that a definite DB is a special case of an indefinite DB, where each element is a definite tuple and only one possible 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 $\mathcal{IC}$ of integrity constraints over $\mathcal{DS}$, an indefinite DB instance D of $\mathcal{DS}$ is said to be *consistent* w.r.t. $\mathcal{IC}$ (denoted as $D \models \mathcal{IC}$) iff there is at least one possible world of D which is consistent w.r.t. $\mathcal{IC}$ (in the standard model-theoretic sense), that is, $\{W \mid W \in \mathcal{W}(D), W \models \mathcal{IC}\} \neq \emptyset$; otherwise, D is said to be *inconsistent* (w.r.t. $\mathcal{IC}$).

Example 2. *Continuing from Example 1, let the set $\mathcal{IC}_{ex}$ of integrity constraints consist 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$\rightarrow$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 [Grant and Minker, 1985] 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. $\mathcal{IC}_{ex}$. In particular, going through the integrity 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 (it is worth noting 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. $\mathcal{IC}_{ex}$, as each one violates one or more constraints (e.g., c_1 is violated by all of them).

2.2 Complexity Classes

The complexity classes Σ_k^p, Π_k^p and Δ_k^p, with $k \geq 0$, are defined as follows [Papadimitriou, 1994]:
i) $\Sigma_0^p = \Pi_0^p = \Delta_0^p = P$;
ii) $\Sigma_1^p = NP$ and $\Pi_1^p = coNP$;
iii) $\Delta_k^p = P^{\Sigma_{k-1}^p}$, $\Sigma_k^p = NP^{\Sigma_{k-1}^p}$, and $\Pi_k^p = co\Sigma_k^p, \forall k > 0$.
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 $\Sigma_k^p \subseteq \Delta_{k+1}^p \subseteq \Sigma_{k+1}^p \subseteq PSPACE$ and $\Pi_k^p \subseteq \Delta_{k+1}^p \subseteq \Pi_{k+1}^p \subseteq PSPACE$.

A decision problem is in D_k^p iff it is the conjunction of a problem in Σ_k^p and a problem in Π_k^p. D_1^p is also denoted as D^p. It holds that $D^p \subseteq \Delta_2^p$.

We will also use the class CNP from the counting polynomial hierarchy defined in [Wagner, 1986]. This class relies on a counting quantifier C defined as follows. Given a predicate $H(x, y)$ with free variables x and y, $C_y^k H(x, y)$ 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. A canonical problem for CNP is deciding whether an open quantified boolean formula $\forall Y \Phi(X, Y)$ has at least k many satisfying assignments (for the variables X in the open part). 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 [Wagner, 1986], 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. We will focus on $C_{=}D^p$. To the best of our knowledge, it is not known whether $C_{=}NP$ coincides with $C_{=}D^p$, neither whether $C_{=}NP$ coincides with $C_{=}\mathcal{B}(NP)$.

FP is the class of the function problems that can be solved by a deterministic Turing machine in polynomial time. For a class $\mathcal{C}$, $FP^{\mathcal{C}}$ 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., $FP^{\Sigma^p_2[\log n]}$).

We also use the class $\# \cdot coNP$ [Hemaspaandra and Vollmer, 1995]. 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 $\#P$ [Valiant, 1979], which coincides with $\# \cdot P$. It holds that $\# \cdot P \subseteq \# \cdot coNP$ and $\# \cdot coNP = \# \cdot \Delta^p_2$.

3 Relative Inconsistency Measures

In this section, we develop relative IMs for indefinite DBs in analogy with relative IMs for propositional knowledge bases [Besnard and Grant, 2020]. Although these methods are inspired by IMs for propositional logic we do not apply them directly as done e.g. in [Grant and Parisi, 2020; 2022] where a general information space, which encompasses definite databases, is translated into an inconsistency equivalent propositional knowledge base (KB), and then propositional IMs are applied to find the inconsistency. This approach makes no distinction between DB tuples and integrity constraints, i.e., IMs blame simultaneously tuples and constraints without considering that inconsistency in DBs typically refers to the tuples rather than the integrity constraints. Rather, we use well-known methods as inspiration to define (by analogy) IMs that are applicable to indefinite DBs. Every IM measures the inconsistency by blaming (indefinite) tuples only. This is different from measuring the inconsistency of a set of formulas, all of which have the same status, as is the case of propositional KBs. This leads to some substantial differences between the two cases, e.g. some measures that are distinct for KBs become identical in our setting (cf. Proposition 1).

We use **D** to denote the set of all indefinite DB instances over a fixed but arbitrary DB scheme $\mathcal{DS}$. In general, we will omit the DB scheme and the set $\mathcal{IC}$ of integrity constraints in the terminology. A **minimal inconsistent subset** (MIS) of DB D is a set of elements $X \subseteq D$ such that X is inconsistent (w.r.t. $\mathcal{IC}$) and no proper subset of X is inconsistent. We denote by $\mathsf{MI}(D)$ the set of MISs 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. We use $\mathsf{Tuples}(D)$ to denote the set of definite tuples of a DB D, that is $\mathsf{Tuples}(D) = \bigcup_{e \in D} e$.

To define relative IMs, we use the following postulates.

Definition 1 (Basic Postulates)**.** *Let D, D' be DBs, and $\mathcal{I} : \boldsymbol{D} \to \mathbb{R}^{\geq 0}_{\infty}$ a function. The basic postulates are as follows:*

Consistency $\mathcal{I}(D) = 0$ *iff D is consistent.*

Normalization $0 \leq \mathcal{I}(D) \leq 1$.

Free-Element Reduction *For $e \notin D$, if $e \in \mathsf{Free}(D \cup \{e\})$ and $\mathcal{I}(D) \neq 0$, then $\mathcal{I}(D \cup \{e\}) < \mathcal{I}(D)$.*

Relative Separability *If $\mathsf{MI}(D \cup D') = \mathsf{MI}(D) \cup \mathsf{MI}(D')$, $\mathsf{Tuples}(D) \cap \mathsf{Tuples}(D') = \emptyset$, $\mathcal{I}(D) \neq 0$, $\mathcal{I}(D') \neq 0$, and $\mathcal{I}(D) \gtrapprox \mathcal{I}(D')$, then $\mathcal{I}(D) \gtrapprox \mathcal{I}(D \cup D') \gtrapprox \mathcal{I}(D')$, where either $\gtrapprox$ is $<$ in every instance or $\gtrapprox$ is $=$ in every instance.*

Assuming that $\mathcal{I}$ is a function measuring inconsistency, Consistency means that all and only consistent databases get measure 0. Consistency is the only postulate that is required to be satisfied by every IM. In some cases, the satisfaction of a postulate called Monotony is also required for absolute IMs. Monotony means that the enlargement of a database cannot decrease its measure. However, Monotony is not appropriate for relative IMs where the ratio of inconsistency may decrease with the addition of consistent information. In contrast, Normalization is appropriate for relative IMs; it states that an IM cannot have value greater than 1.

Free-Element Reduction requires that adding a free element to an inconsistent DB (that is, adding an element that does not introduce a new conflict) reduces the IM. The reason is that while the numerator remains the same, the denominator decreases. But Free-Element Reduction may not hold for indefinite DBs because while the element is new, it may contain tuples already in the DB. Relative Separability deals with the case where a DB can be split into two inconsistent parts. If $\gtrapprox$ is $=$, then the DB has the same IM as the two parts. Otherwise, the DB has an inconsistency value in between the inconsistency values of the two parts. Thus under the specified conditions the relative measure of the union sort of averages out the relative measures of the components; however, the result need not be the exact average.

We are now ready to define the concept of relative IM. The definition is inspired by that in [Besnard and Grant, 2020], where it is shown that all IMs that are intuitively relative measures satisfy both Consistency and Normalization and any IM

that satisfies neither Free-Formula Independence nor Relative Separability does not satisfy our intuitive notion of a relative measure. This way we restrict what we may consider as a relative measure without making the definition so strict that some reasonable relative IMs would be excluded.

Definition 2 (Relative IM). *A function $\mathcal{I} : \boldsymbol{D} \to \mathbb{R}^{\geq 0}_{\infty}$ is a* **relative inconsistency measure** *iff it satisfies the postulates Consistency, Normalization, and either Free-Formula Reduction or Relative Separability (or both).*

In the following subsections we will define five IMs that will be shown to satisfy Definition 2 in Section 4. It is worth noting that, in general, normalizing an absolute IM may not result in a relative IM (as Definition 2 may not be satisfied).

3.1 Relative Measures Using MISs

We introduce relative IMs that rely on MISs and on the related concepts defined earlier. In the following definition we assume that $D \neq \emptyset$ and set $\mathcal{I}(\emptyset) = 0$ in all cases.

Definition 3 (Relative Inconsistency Measures). *For any DB D, the IMs $\mathcal{I}_{mv}$ $\mathcal{I}^r_M$, $\mathcal{I}^r_P$, and $\mathcal{I}^r_H$ are such that*

- $\mathcal{I}_{mv}(D) = \dfrac{|\mathsf{Tuples}(\bigcup_{X \in \mathsf{MI}(D)} X)|}{|\mathsf{Tuples}(D)|}$
- $\mathcal{I}^r_M(D) = \dfrac{|\mathsf{MI}(D)|}{\binom{|D|}{\lfloor |D|/2 \rfloor}}$
- $\mathcal{I}^r_P(D) = \dfrac{|\mathsf{Problematic}(D)|}{|D|}$
- $\mathcal{I}^r_H(D) = \dfrac{\min\{|X| \text{ } s.t. \text{ } X \subseteq D \text{ and } \forall M \in \mathsf{MI}(D), X \cap M \neq \emptyset\}}{|D|}$

Thus, $\mathcal{I}_{mv}(D)$ is the number of definite tuples occurring in some MIS divided by the amount of all such tuples in the DB (observe that $|\mathsf{Tuples}(D)|$ can be greater than, lower than, or equal to $|D|$ depending on how definite tuples are combined to form indefinite ones). The proportion of propositional variables is considered by this measure in propositional logic [Xiao and Ma, 2012]. $\mathcal{I}^r_M$ is the ratio of the number of MISs to the maximum possible number of such subsets. The rationale is that a MIS represents a minimal inconsistency for a set of database elements; hence this measure is proportional the number of such inconsistencies [Hunter and Konieczny, 2008]. The denominator of $\mathcal{I}^r_M$ is the maximum number of MISs that can occur in a database of size D. As $\mathsf{MI}(D)$ is a Sperner family over D, its size is maximized if its elements have size $\lfloor |D|/2 \rfloor$ [Thimm, 2016]. We use superscript 'r' for relative IMs whose criterion for quantifying inconsistency is obtained by relativizing some absolute IM. Next, $\mathcal{I}^r_P$ is the ratio of the number of elements that are in one or more MISs to the size of the database. This is the relativized version of the so-called problematic measure [Grant and Hunter, 2011]. The numerator of $\mathcal{I}^r_H$ corresponds to the minimal number of elements whose deletion makes the database consistent [Grant and Hunter, 2013]. Hence $\mathcal{I}^r_H$ can be written as $\mathcal{I}^r_H = \min\{|X| \text{ } s.t. \text{ } D \setminus X \text{ is consistent } \}/|D|$.

Example 3. *Continuing with our running example, we have that $\mathsf{MI}(D_{ex}) = \{\{e_2\}, \{e_5, e_6, e_7\}\}$. Note that $\{e_1, e_2\}$ as well as $\{e_2, e_3\}$ are not included because they contains $\{e_2\}$. Thus there are 4 problematic elements in D_{ex}, and 3 free elements. The values of the IMs for D_{ex} are as follow.*

- *$\mathcal{I}_{mv}(D_{ex}) = \frac{5}{9}$ as 5 of the 9 tuples in D_{ex} are in MISs.*
- *$\mathcal{I}^r_M(D_{ex}) = \frac{2}{35}$ as there are 2 MISs as given above while 35 is the maximum number of MIS that can occur in a database consisting of 7 elements.*
- *$\mathcal{I}^r_P(D_{ex}) = \frac{4}{7}$ as 4 of the 7 elements are in $\mathsf{MI}(D_{ex})$.*
- *$\mathcal{I}^r_H(D_{ex}) = \frac{2}{7}$ as no less than 2 of the 7 elements intersect with each MIS.*

3.2 A Relative Measure Using 3VL

We now consider the Contension measure [Grant and Hunter, 2011], 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*). The logical connectives are extended to 3VL interpretations using Priest's three-valued logic, the Logic of Paradox [Priest, 1979]. 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 $B \vee F = B$. The 3VL semantics extends to first-order logic in our case as we consider the grounded versions of the constraints. 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. In the DB context this means that for a given D with a set $\mathcal{IC}$ 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.

We now introduce the IM $\mathcal{I}^r_C$. As before, $\mathcal{I}^r_C(\emptyset) = 0$.

Definition 4 (Relative Contension Measure). *For any DB D,*

$$\mathcal{I}^r_C(D) = \frac{\min\{|\mathsf{Conflictbase}(i)| \mid i \in \mathsf{Models}(D)\}}{|\mathsf{Tuples}(D)|}.$$

Hence, $\mathcal{I}^r_C$ is the minimal number of definite tuples that if we could consider them both true and false would resolve all inconsistencies divided by the size of the DB. For our running example, we have $\mathcal{I}^r_C(D_{ex}) = \frac{2}{9}$ 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.

3.3 Measures Coinciding for Definite DBs

In the propositional case, of the IMs considered in this paper no two give the same result for all KBs. But because of the structure of DBs, for definite DBs two equalities hold: 1) $\mathcal{I}_{mv}(D) = \mathcal{I}^r_P(D)$, and 2) $\mathcal{I}^r_C(D) = \mathcal{I}^r_H(D)$. The first follows from the fact that, for a definite DB, every element is a definite tuple, and thus $\mathsf{Tuples}(D) = D$ and the problematic elements are exactly the tuples in the MISs. The second equality follows from the fact that the minimum number of tuples that need to be assigned B in order to get a 3VL model in $\mathsf{Models}(D)$ is the same as the cardinality of the set $X \subseteq D$ having a non-empty intersection with every MIS of D.

Proposition 1. *For any definite database D, $\mathcal{I}_{mv}(D) = \mathcal{I}^r_P(D)$ and $\mathcal{I}^r_C(D) = \mathcal{I}^r_H(D)$.*

As will be noted after Theorem 1, no other pair of considered measures is identical for definite or indefinite DBs.

For indefinite DBs, $\mathcal{I}_{mv}$ and $\mathcal{I}^r_P$ need not give the same result. For instance, considering $D' = \{\{R(-1), R(-2)\}, \{R(-1),\ R(3)\}\}$ and $\mathcal{IC}' = \{\neg R(x) \vee x > 0\}$, we have that $\mathcal{I}_{mv}(D') = 2/3 > \mathcal{I}^r_P(D') = 1/2$ since there are three tuples in D' forming two elements, and there are two tuples in the first element which is a self-contradiction. Similarly, $\mathcal{I}^r_H$ and $\mathcal{I}^r_C$ need not give the same result. For instance, we have that $\mathcal{I}^r_H(D') = 1/2 > \mathcal{I}^r_C(D') = 1/3$ because one of the two elements is a self-contradiction and for 3VL it suffices to assign $R(-1)$ the value B, $R(-2)$ the value F, and $R(3)$ the value T to eliminate the inconsistency. In general, $\mathcal{I}^r_C(D) \leq \mathcal{I}^r_H(D)$ holds for all indefinite DBs because assigning B to one of the disjuncts in an element that needs removal for $\mathcal{I}^r_H$ suffices to eliminate the inconsistency for $\mathcal{I}^r_C$ as well, and the denominator of $\mathcal{I}^r_C$ is greater than or equal to that of $\mathcal{I}^r_H$. Although our interest is in relative measures, we wish to point out that even the absolute versions, $\mathcal{I}_H$ and $\mathcal{I}_C$ are not the same. They happen to have the same value, 1 for D' but consider D'', the same as D' except that $R(3)$ is replaced by $R(-3)$. As now both elements are contradictions, $\mathcal{I}_H(D'') = 2$, while $\mathcal{I}_C(D'') = 1$ as it suffices to assign $R(-1)$ the value B and both $R(-2)$ and $R(-3)$ the value F.

4 Postulate Satisfaction for Relative IMs

In the previous section we formulated four important postulates relevant for relative IMs. In this section we introduce four additional postulates for indefinite DBs that relative IMs may or may not satisfy. We avoid postulates that were shown in [Besnard and Grant, 2020] not to hold for relative IMs. Then, we show which of the eight postulates are satisfied by the five IMs defined previously.

Definition 5 (Additional Postulates Relevant for Relative IMs)**.** *Let D be an indefinite DB and $\mathcal{I} : \boldsymbol{D} \to \mathbb{R}^{\geq 0}_{\infty}$ a function. The additional postulates are as follows:*

Safe-Element Reduction *If $e \cap \mathsf{Tuples}(D) = \emptyset$ and $\mathcal{I}(D) \neq 0$, then $\mathcal{I}(D \cup \{e\}) < \mathcal{I}(D)$.*

MI-Normalization *If $\mathsf{MI}(D) = D$, then $\mathcal{I}(D) = 1$.*

Equal Conflict *If $\mathsf{MI}(D) = D$, $\mathsf{MI}(D') = D'$, and $|D| = |D'|$, then $\mathcal{I}(D) = \mathcal{I}(D')$.*

Contradiction *$\mathcal{I}(D) = 1$ iff for all $\emptyset \neq D' \subseteq D$, $\mathcal{I}(D') > 0$.*

Safe-Element Reduction is a weak version of Free-Element Reduction where we require that e contain no tuple in D. In fact, Free-Element Reduction implies Safe-Element Reduction. MI-Normalization and Equal Conflict deal specifically with MISs. MI-Normalization requires every database coinciding with a MIS to have measure 1. Equal Conflict requires MISs of the same size to have the same measure, thus stating a similarity between MISs of the same size. MI-Normalization implies Equal Conflict but the converse does not hold. Finally, Contradiction requires that the highest relative inconsistency measure, 1, be reserved for DBs all of whose nonempty subsets are inconsistent. The intuition is that a DB that has a consistent portion should not be considered 100% inconsistent.

	Inconsistency Measures				
	$\mathcal{I}_{mv}$	$\mathcal{I}^r_M$	$\mathcal{I}^r_P$	$\mathcal{I}^r_H$	$\mathcal{I}^r_C$
Consistency	✓	✓	✓	✓	✓
Normalization	✓	✓	✓	✓	✓
Free-Element Reduction	✓✗	✓	✓	✓	✓✗
Relative Separability	✓	✗	✓	✓	✓
Safe-Element Reduction	✓	✓	✓	✓	✓
MI-Normalization	✓	✗	✓	✗	✗
Equal Conflict	✓	✓	✓	✓	✓✗
Contradiction	✗	✗	✗	✓	✓

Table 2: Postulate satisfaction for relative IMs. ✓: satisfied for both definite and indefinite DBs, ✓✗: satisfied for definite DBs but not satisfied for indefinite DBs, ✗: not satisfied for both definite and indefinite DBs.

It is useful to note that MI-Normalization and Contradiction are incompatible postulates because if a database D that is its own MIS is such that $|D| > 1$ then Contradiction fails. We would have to restrict the constraints to ones that contain only a single relation symbol to gain compatibility. Importantly, this also means that no IM can satisfy all 8 postulates; hence the ones that satisfy 7 satisfy as many as possible in the list.

Theorem 1. *For definite and indefinite DBs the satisfaction of postulates for relative IMs is as given in Table 2.*

Thus, the functions defined in the previous section are relative IMs according to Definition 2, as each of them satisfies Consistency, Normalization, and Free-Formula Reduction or Relative Separability. The other four postulates are not mandatory; they simply allow us to better understand the behavior of relative IMs in some circumstances. For instance, if Contradiction is satisfied and the inconsistency value is 100%, then the DB is 'totally inconsistent', in the sense that each of its subset is inconsistent. On the other hand, if MI-Normalization is satisfied and the inconsistency value is 100%, then we should be aware that there is a chance that the inconsistency could be ascribed to a single MIS coinciding with the DB. Overall, the results of Theorem 1 allow us to compare the IMs in terms of the satisfied postulates. Both $\mathcal{I}^r_P$ and $\mathcal{I}^r_H$ satisfy as many postulates as possible, though with alternative behavior w.r.t. the two incompatible postulates, and this holds also for $\mathcal{I}^r_{mv}$ and $\mathcal{I}^r_C$ for definite DBs.

Finally, as a consequence of Theorem 1, we have that, except for the equalities given in Proposition 1 for the case of definite DBs, no other pair of IMs in Table 2 are identical since they do not satisfy exactly the same set of postulates.

5 Complexity of Relative IMs

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 DB.

Definition 6 (Lower (**LV**), Upper (**UV**), and Exact Value (**EV**) problems)**.** *Let $\mathcal{I}$ be an IM. Given a database D over a*

	$\mathbf{LV}_{\mathcal{I}}(D,v)$		$\mathbf{UV}_{\mathcal{I}}(D,v)$		$\mathbf{EV}_{\mathcal{I}}(D,v)$		$\mathbf{IM}_{\mathcal{I}}(D)$	
	definite	indefinite	definite	indefinite	definite	indefinite	definite	indefinite
$\mathcal{I}_{mv}$	P	Σ_2^p-c	P	Π_2^p-c	P	D_2^p-c	FP	$FP^{\Sigma_2^p[\log n]}$
$\mathcal{I}_M^r$	P	$coNP$-h, CNP	P	NP-h, CNP	P	D^p-h, $C_{=}D^p$	FP	$\# \cdot coNP$
$\mathcal{I}_P^r$	P	Σ_2^p-c	P	Π_2^p-c	P	D_2^p-c	FP	$FP^{\Sigma_2^p[\log n]}$
$\mathcal{I}_H^r$	$coNP$-c	$coNP$-c	NP-c	NP-c	D^p-c	D^p-c	$FP^{NP[\log n]}$-c	$FP^{NP[\log n]}$-c
$\mathcal{I}_C^r$	$coNP$-c	$coNP$-c	NP-c	NP-c	D^p-c	D^p-c	$FP^{NP[\log n]}$-c	$FP^{NP[\log n]}$-c

Table 3: Complexity of Lower Value (**LV**), Upper Value (**UV**), Exact Value (**EV**), and Inconsistency Measurement (**IM**) problems for definite and indefinite DBs. 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 $\mathcal{C}'$.

fixed database scheme with a fixed set of integrity constraints, and a value $v \in \mathbb{Q}^{(0,1]}$,
• ***LV***$_{\mathcal{I}}(D, v)$ *is the problem of deciding whether $\mathcal{I}(D) \geq v$.*
Given D and a value $v' \in \mathbb{Q}^{[0,1]}$,
• ***UV***$_{\mathcal{I}}(D, v')$ *is the problem of deciding whether $\mathcal{I}(D) \leq v'$,*
• ***EV***$_{\mathcal{I}}(D, v')$ *is the problem of deciding whether $\mathcal{I}(D) = v'$.*

We avoid considering values for which the considered problems are trivial, e.g. $\mathbf{LV}_{\mathcal{I}}(D, 0)$ is true for any IM.

We also consider the problem of determining the IM value.

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

The following proposition states the complexity of the above-mentioned problems for the case of definite DBs.

Proposition 2. *For any definite DB and IM $\mathcal{I} \in \{\mathcal{I}_{mv}, \mathcal{I}_M^r, \mathcal{I}_P^r, \mathcal{I}_H^r, \mathcal{I}_C^r\}$, the complexity of* ***LV***$_{\mathcal{I}}$*,* ***UV***$_{\mathcal{I}}$*,* ***EV***$_{\mathcal{I}}$*, and* ***IM***$_{\mathcal{I}}$ *is as given in Table 3.*

Thus, $\mathcal{I}_M^r$ and $\mathcal{I}_{mv}$ (and $\mathcal{I}_P^r$ which coincides with $\mathcal{I}_{mv}$ for definite DBs) are tractable for definite databases. As we show next, the presence of disjunction entails that the measures $\mathcal{I}_{mv}$, $\mathcal{I}_M^r$, and $\mathcal{I}_P^r$ are no longer tractable. Indeed, for indefinite DBs, depending on the considered IM, the complexity of the considered problems reaches the first level of the polynomial hierarchy or beyond (even under data-complexity). However, the results obtained for the measures $\mathcal{I}_H^r$ and $\mathcal{I}_C^r$ in the case of definite DBs still hold for indefinite DBs.

Theorem 2. *For any indefinite DB and IM $\mathcal{I} \in \{\mathcal{I}_{mv}, \mathcal{I}_M^r, \mathcal{I}_P^r, \mathcal{I}_H^r, \mathcal{I}_C^r\}$, the complexity of* ***LV***$_{\mathcal{I}}$*,* ***UV***$_{\mathcal{I}}$*,* ***EV***$_{\mathcal{I}}$*, and* ***IM***$_{\mathcal{I}}$ *is as given in Table 3.*

Most of the results in Table 3 hold even if the set of constraints consists of FDs only. In fact, all the membership results trivially hold for FDs as they hold for the more general class of DCs. Moreover, we can show the following result.

Proposition 3. *All the hardness results in Table 3 for* ***LV*** *and* ***UV*** *still hold if the set of constraints consists of FDs only.*

In particular, the NP-hardness results for $\mathcal{I}_H^r$ and $\mathcal{I}_C^r$ can be shown in cases where the set of constraints consists of two FDs only. Moreover, the results for $\mathcal{I}_M^r$ hold even if the constraint is a single FD. Finally, we can show that also the Σ_2^p/Π_2^p-hardness results for $\mathcal{I}_{mv}$ and $\mathcal{I}_P^r$ (for the case of indefinite DBs) hold even in the presence of a single FD.

6 Conclusions and Future Work

We have introduced relative IMs for indefinite DBs and analyzed postulate compliance as well as their complexity for both indefinite and definite DBs. Our work contributes to understanding how the database counterpart of some methods to quantify inconsistency in propositional logic behaves in the DB context, where data are generally the reason for inconsistency, not the integrity constraints. The results in Tables 2 and 3 give insights on the behavior of relative IMs, helping to figure out which measure could be appropriate for specific applications. We do not believe that there is a "best" IM. An IM gives information about a DB that can be used in different ways depending on the circumstance. For instance, maintaining an IM-based progress bar during database repair involving deletions [Livshits *et al.*, 2021] has different requirements than measuring inconsistency for bank holding companies [Fan *et al.*, 2019] where the information cannot be deleted even if it is inconsistent. For this situation, if interested to the proportion of potential issues (i.e., MISs) to be resolved, $\mathcal{I}_M^r$ could be a good candidate unless other criteria take precedence. The computational cost is also relevant. For definite DBs, $\mathcal{I}_M^r$ and $\mathcal{I}_{mv}$, which coincides with $\mathcal{I}_P^r$, can be computed in polynomial time by standard SQL. For indefinite DBs, $\mathcal{I}_H^r$ and $\mathcal{I}_C^r$ look to be the cheaper measures. In this regard, a dichotomy for FDs for the problem of computing the cost of a cardinality repair (that is equivalent to computing the numerator of $\mathcal{I}_H^r$) for definite DBs has been presented in [Livshits *et al.*, 2020]. This entails that there are cases where computing $\mathcal{I}_H^r$ is polynomial, but also cases where it is APX-complete (it cannot be approximated better than some constant) and it has a polynomial-time 2-approximation.

We envisage several interesting directions for future work. They include: *i*) considering other forms of incomplete information (not just disjunctive tuples) such as maybe information [Liu and Sunderraman, 1990] and dealing with DBs with null values; *ii*) extending our work to other types of integrity constraints, particularly to inclusion dependencies; *iii*) devising IMs working at the attribute-level and able to distinguish inconsistencies arising from different (sets of) attributes (by following the idea of dimensional inconsistency measures proposed in [Grant *et al.*, 2021]); and finally *iv*) developing an ASP-based implementation by exploiting e.g. the DB-oriented features of DLV^{DB} [Alviano *et al.*, 2010] in order to evaluate relative IMs.

Acknowledgments

We thank the reviewers for their valuable comments and suggestions. The fist author acknowledges financial support from PNRR MUR project PE0000013-FAIR.

References

[Alviano *et al.*, 2010] M. Alviano, W. Faber, N. Leone, S. Perri, G. Pfeifer, and G. Terracina. The disjunctive datalog system DLV. In *Proc. of International Workshop on Datalog*, pages 282–301, 2010.

[Beeri and Vardi, 1984] C. Beeri and M. Y. Vardi. A proof procedure for data dependencies. *J. ACM*, 31(4):718–741, 1984.

[Benjelloun *et al.*, 2008] O. Benjelloun, A. Das Sarma, A. Y. Halevy, M. Theobald, and J. Widom. Databases with uncertainty and lineage. *VLDB J.*, 17(2):243–264, 2008.

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

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

[Besnard and Grant, 2020] P. Besnard and J. Grant. Relative inconsistency measures. *Artif. Intell.*, 280:103231, 2020.

[Besnard, 2014] P. Besnard. Revisiting postulates for inconsistency measures. In *Proc. of European Conference Logics in Artificial Intelligence (JELIA)*, pages 383–396, 2014.

[Bona *et al.*, 2019] G. De Bona, J. Grant, A. Hunter, and S. Konieczny. Classifying inconsistency measures using graphs. *J. Artif. Intell. Res.*, 66:937–987, 2019.

[Costa and Martins, 2018] D. Costa and M. A. Martins. Inconsistency measures in hybrid logics. In J. Grant and M. V. Martinez, editors, *Measuring Inconsistency in Information*, pages 169–194. College Publications, 2018.

[Decker and Martinenghi, 2008] 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)*, pages 195–204, 2008.

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

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

[Decker, 2017] 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.

[Decker, 2018] H. Decker. Measuring database inconsistency. In J. Grant and M. V. Martinez, editors, *Measuring Inconsistency in Information*, pages 271–311. College Publications, 2018.

[Fan *et al.*, 2019] 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)*, pages 5:1–5:5, 2019.

[Grant and Hunter, 2011] 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)*, pages 362–373, 2011.

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

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

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

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

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

[Grant and Parisi, 2022] John Grant and Francesco Parisi. General information spaces: measuring inconsistency, rationality postulates, and complexity. *Ann. Math. Artif. Intell.*, 90(2-3):235–269, 2022.

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

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

[Hausken and Mohr, 2001] K. Hausken and M. Mohr. The value of a player in n-person games. *Soc. Choice Welf.*, 18(3):465–483, 2001.

[Hemaspaandra and Vollmer, 1995] L. A. Hemaspaandra and H. Vollmer. The satanic notations: counting classes beyond #P and other definitional adventures. *SIGACT News*, 26(1):2–13, 1995.

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

[Imielinski *et al.*, 1995] T. Imielinski, R. van der Meyden, and K. V. Vadaparty. Complexity tailored design: A new design methodology for databases with incomplete information. *J. Comput. Syst. Sci.*, 51(3):405–432, 1995.

[Issa *et al.*, 2020] O. Issa, A. Bonifati, and F. Toumani. Evaluating top-k queries with inconsistency degrees. *Proc. VLDB Endow.*, 13(11):2146–2158, 2020.

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

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

[Levy, 1999] A. Y. Levy. Combibining artificial intelligence and databases for data integration. In *Artificial Intelligence Today: Recent Trends and Developments*, pages 249–268. Springer, 1999.

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

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

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

[Livshits *et al.*, 2021] E. Livshits, R. Kochirgan, S. Tsur, I. F. Ilyas, B. Kimelfeld, and S. Roy. Properties of inconsistency measures for databases. In *Proc. of International Conference on Management of Data (SIGMOD)*, pages 1182–1194, 2021.

[Martinez *et al.*, 2007] 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 Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU)*, pages 103–114, 2007.

[Minker and Seipel, 2002] J. Minker and 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*, pages 472–511, 2002.

[Minker, 1982] J. Minker. On indefinite databases and the closed world assumption. In *Proc. of the 6th Conference on Automated Deduction*, pages 292–308, 1982.

[Molinaro *et al.*, 2009] C. Molinaro, J. Chomicki, and J. Marcinkowski. Disjunctive databases for representing repairs. *Ann. Math. Artif. Intell.*, 57(2):103–124, 2009.

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

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

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

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

[Parisi and Grant, 2023] Francesco Parisi and John Grant. On measuring inconsistency in definite and indefinite databases with denial constraints. *Artif. Intell.*, 318:103884, 2023.

[Priest, 1979] G. Priest. Logic of paradox. *Journal of Philosophical Logic*, 8:219–241, 1979.

[Sarker, 2021] I. H. Sarker. Data science and analytics: An overview from data-driven smart computing, decision-making and applications perspective. *SN Comput. Sci.*, 2(5):377, 2021.

[Thimm and Wallner, 2019] M. Thimm and J. P. Wallner. On the complexity of inconsistency measurement. *Artif. Intell.*, 275:411–456, 2019.

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

[Thimm, 2018] M. Thimm. On the evaluation of inconsistency measures. In J. Grant and M. V. Martinez, editors, *Measuring Inconsistency in Information*, pages 19–60. College Publications, 2018.

[Ulbricht *et al.*, 2020] M. Ulbricht, M. Thimm, and G. Brewka. Handling and measuring inconsistency in non-monotonic logics. *Artif. Intell.*, 286:103344, 2020.

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

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

[Xiao and Ma, 2012] G. Xiao and Y. Ma. Inconsistency measurement based on variables in minimal unsatisfiable subsets. In *Proc. of European Conference on Artificial Intelligence (ECAI)*, pages 864–869, 2012.

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

[Zhou *et al.*, 2009] L. Zhou, H. Huang, G. Qi, Y. Ma, Z. Huang, and Y. Qu. Measuring inconsistency in DL-Lite ontologies. In *Proc. of International Conference on Web Intelligence (WI)*, pages 349–356, 2009.