The Shapley Value in Database Management

Leopoldo Bertossi
SKEMA Business School
Montreal, Canada
leopoldo.bertossi@skema.edu

Benny Kimelfeld
Technion – Israel Institute of Technology
Haifa, Israel
bennyk@technion.ac.il

Ester Livshits
University of Edinburgh
Edinburgh, UK
ester.livshits@ed.ac.uk

Mikaël Monet
Université de Lille, CNRS, Inria, UMR 9189 -
CRIStAL, F-59000 Lille
Lille, France
mikael.monet@inria.fr

ABSTRACT

Attribution scores can be applied in data management to quantify the contribution of individual items to conclusions from the data, as part of the explanation of what led to these conclusions. In Artificial Intelligence, Machine Learning, and Data Management, some of the common scores are deployments of the Shapley value, a formula for profit sharing in cooperative game theory. Since its invention in the 1950s, the Shapley value has been used for contribution measurement in many fields, from economics to law, with its latest researched applications in modern machine learning. Recent studies investigated the application of the Shapley value to database management. This article gives an overview of recent results on the computational complexity of the Shapley value for measuring the contribution of tuples to query answers and to the extent of inconsistency with respect to integrity constraints. More specifically, the article highlights lower and upper bounds on the complexity of calculating the Shapley value, either exactly or approximately, as well as solutions for realizing the calculation in practice.

1 Introduction

Explanations have been investigated in Artificial Intelligence (AI) and Machine Learning (ML) for some decades, and also, for a much longer time, in other fields such as Philosophy, Logic, Physics, and Statistics. Actually, the explicit study of explanations can be traced back to the ancient Greeks, who were already concerned with causes and effects. A whole new area of research has emerged around explanations in AI (*Explainable AI* or *XAI* [54]) and Data Science. The basic need is fundamental: a human applies AI to make important decisions, and the human is accountable for these decisions as well as for the very choice to involve the specific AI al-

gorithms in deployment. Hence, adopting the algorithm's decision may necessitate some level of human understanding of what led to the decision.

In light of that, a decision or classification model, typically obtained through machine learning, should be able to attach an explanation to its outcome. Stakeholders, such as an applicant for a loan from the bank, may desire a reason for the decision, especially if the request is declined. In a similar vein, a database may contain massive volumes of data or be built according to an intricate schema. Query answers could be difficult to explain or quantify in terms of the relevance of specific data in the database. This is also true for other database phenomena, such as the violation of integrity constraints.

In this article, we will concentrate mostly on explanations in data management, and more specifically, in relational databases (which we refer to simply as *databases* in the remainder of the manuscript). Different data-related phenomena may open a quest for explanations. Typically, one desires to understand *why* an answer is returned when applying the query to the database. One may also want to understand why a particular potential answer is *not* returned. In another scenario, they may wish to understand why an aggregate query result is a number that is unexpectedly high or low. It is also of interest to understand why an integrity constraint is not satisfied by the database. Even more, given a set of integrity constraints and a numerical measure of their violation by the database, we may want to know the extent to which a data item is responsible for the database's inconsistency [8, 45, 48].

We consider explanations that point to data items that contribute to the outcome that we wish to explain. As many items might take part in the outcome one way or another, we need to be able

to *quantify* this contribution. To that aim, one of the fundamental concepts adopted is the *Shapley value* [63], which is a formula for wealth distribution in a cooperative game. The Shapley value has a plethora of applications, including profit sharing between Internet providers [51], influence measurement in social networks [57], the importance of genes for specific body functions [56], and key-player identification in terrorist networks [67], to name a few. In the context of explanations, the idea is straightforward: data items are the players who play the game of establishing the outcome.

It is relevant to emphasize that the Shapley value corresponds to a *principled approach* to quantify the contribution of a player to a wealth function that is shared by a set of players. Its inception started by stating some desired general properties of such a contribution score. It was next proved that there is only one such score that satisfies those properties, namely the Shapley value [60].

An inherent challenge in the application of the Shapley value for explanations (and other tasks) is its computational complexity—the Shapley value is often intractable to calculate, and particularly, the execution cost might grow exponentially with the number of players. Hence, past research investigated islands of tractability and approximation techniques for the Shapley value.

In the remainder of the introduction, we delve into some background on explanations in AI and their development into explanations in databases.

Background on Explanations

Explanations may come in different forms, and different kinds of them have been proposed and investigated in the context of Computer Science. These include the area of *model-based diagnosis*, whose most prominent kinds are *consistency-based diagnosis* and *abductive diagnosis* [64]. *Causality* [58], and particularly *actual causality* [34], deem causes as explanations for observed effects and phenomena. In all these kinds of diagnosis, an explanation comes in the form of a set of basic propositions, which are expressed in the language of the model that describes the situation related to the observations [10]. In the case of causality, they can be values of observed variables or features.

Actual causality [33, 34] provides *counterfactual explanations* to observations. These explanations are obtained by hypothetically *intervening* in (i.e., changing some components of) the system under observation, to detect whether the intervention leads to changes in the observed behavior. In general terms, counterfactuals are basic propositions about

components of a system that may be a cause for the observed behavior.

Importantly, in actual causality, we distinguish between *endogenous* and *exogenous* variables [34, 35]. Only the former are subject to interventions and may become actual causes. The latter are variables that we do not question, or have no control upon. The separation between endogenous and exogenous variables is application dependent.

Counterfactual causes are actual causes that directly explain the observation: changing them leads to a change in the observation. Actual causes that are non-counterfactual ones are *weaker* causes, in that they require the company of other components to explain the observation. This idea is formalized in quantitative terms by means of the *responsibility score* [17] that captures the causal strength of an explanation. In this way, the score takes into account the *amount* of company that a potential explanation requires in order to become a counterfactual cause. Accordingly, a counterfactual explanation may come with a *responsibility score* that represents its causal strength in relation to the observation. A numerical score like this is usually called an *attribution score* (of the explanation for the observation).

Attribution scores in the explanation of causes are widely investigated in ML. They are provided, most typically and frequently, for outcomes of ML-based decision and classification systems. Next, we discuss a few prominent examples. (For a thorough description, c.f. some recent surveys [15, 32, 53, 54].)

The *Resp* score [11] applies the general responsibility score to the outcomes of classification systems. Specifically, it quantifies the relevance of feature values in an input entity to an ML-based system for the obtained classification label. To this aim, Resp generalizes the responsibility score to deal with non-binary variables. The relevance of feature values is also quantified by the highly popular *SHAP* attribution score [49, 50]. SHAP is a particular application of the Shapley value. More precisely, SHAP *is* the Shapley value for a particular cooperative game played by a set of players that correspond to the features. The tractability and approximability of SHAP for certain classes of classifiers has been recently investigated [3, 4, 68]. The Resp and SHAP scores have also been experimentally compared [11].

Finally, in a different direction, the Shapley value was used to quantify the relevance of particular formulas to the inconsistency of a knowledge base [38].

Explanations in Databases

In database management, explanations usually come as database tuples or cells that play a role in the

observed phenomenon, such as the received query answers. Actual causality and responsibility scores for query answers have been introduced and investigated [9, 13, 52]. In this context, connections between actual causality and model-based diagnosis have been established [12, 13]. The counterfactual interventions are tuple updates: insertions or deletions of tuples, or changes of attribute values in them. These interventions are expected to change the outcome, for instance, to eliminate the query answer that we wish to explain. The identified tuples, as actual causes, are supplemented with responsibility scores, as additional quantitative information, that reflects their explanatory strength. As in AI, we can partition the database into sets of *endogenous tuples* and *exogenous tuples*, and interventions are applied only to the former.

One can also formulate the causal approach to query-answer explanations in databases via *causal networks* [58]. For this purpose the *lineage* of the query [65] can be treated as a causal network. Actually, this has been the approach to define and compute another causality-based score for query answers, the *causal effect*, which has its roots in *causality for observational studies* [39]. The causal effect has been offered as an alternative to actual causality and responsibility in databases [43, 61]. The causal effect of a tuple $\tau \in D$ in relation to a Boolean query q, is the expected difference $\mathbb{E}(q|\tau \in D) - \mathbb{E}(q|\tau \notin D)$, where the database is viewed a probabilistic database [65] (where every tuple can be eliminated randomly and independently) and q is treated as a random variable taking values 0 or 1.

Close to the lineage of a query [6], we find the notion of *provenance* of a query [14, 31], which can be used to explain a query answer by tracing back its origins to the underlying data source.

More recently, Livshits et al. [43] investigated the application of Shapley value to define explanations in databases. In this manuscript, we give a survey of this and other applications of the Shapley value as an explanation mechanism in databases. The first application we discuss assigns to (endogenous) database tuples scores that reflect their importance for an obtained query answer. So, similarly to the application of the Shapley value in explainable machine learning, one has to define an appropriate coalitional game that depends on the query and reflects this importance. (See Section 3.)

The Shapley value has also been applied to quantify the contribution of database tuples to the inconsistency of the database in relation to *integrity constraints*, similarly to its application in knowledge-base inconsistency [38]. This has been done for different measures of inconsistency [45]. (See Section 4.) In the same spirit, an attribution score for inconsistency that is directly based on *database repairs* [7] has been introduced and investigated [8].

Organization

The remainder of the manuscript is organized as follows. We give preliminary teminology and concepts in Section 2. In Section 3, we discuss the application of the Shapley value in the explanation of query answers, and in Section 4 we do so for database inconsistency. Finally, we conclude in Section 5.

2   Basic Concepts

We first present the basic concepts and main notation that we use throughout the article.

Databases. A *database schema* **S** is a finite collection of *relation symbols* R, each with an associated signature $(A_1, \ldots, A_\ell)$ of distinct *attributes* A_i. A *database* D over a schema **S** is a finite collection of *facts* of the form $R(c_1, \ldots, c_k)$ where R is a relation symbol of **S** with the signature $(A_1, \ldots, A_k)$, and each c_i is a *value* that is in the domain of the attribute A_i. For our complexity analysis, we assume that all database values come from a fixed infinite (recursively enumerable) domain **Val**; in particular, we assume that the domain of every attribute is simply **Val**. A fact with the relation symbol R is also called an *R-fact*.

Queries. Let **S** be a schema. A *query* q (over **S**) is associated with an arity $k \geq 0$ and it maps every database D over **S** into a finite k-ary relation of values (that is, a finite subset of $\mathbf{Val}^k$). A query q with arity zero is called a *Boolean* query, and then $q(D)$ is either *true* (i.e., consists of the empty tuple) or *false* (i.e., is empty).

A special case of a query is a *Conjunctive Query* (CQ) that captures the select-project-join queries of SQL and has the logical form

$$\{\mathbf{x} \mid \exists \mathbf{y}[\varphi_1(\mathbf{x}, \mathbf{y}) \wedge \cdots \wedge \varphi_m(\mathbf{x}, \mathbf{y})]\}$$

where $\mathbf{x}$ and $\mathbf{y}$ are disjoint sequences of distinct variables and each $\varphi_i(\mathbf{x}, \mathbf{y})$ is an atomic formula over **S**, that is, a formula of the form $R(t_1, \ldots, t_k)$ where $R \in \mathbf{S}$ is a k-ary relation symbol and each t_i is either a value (constant) or a variable from $\mathbf{x}$ or from $\mathbf{y}$. The variables of $\mathbf{x}$ and $\mathbf{y}$ are called the *free* and the *existential* variables, respectively. For convenient notation, we denote a CQ q by

$$q(\mathbf{x}) :- \varphi_1(\mathbf{x}, \mathbf{y}), \ldots, \varphi_m(\mathbf{x}, \mathbf{y})$$

as in the following example that finds all manager-employee pairs such that the manager manages the

department of the employee:

$$\mathrm{Manages}(x, z) := \mathrm{EmpDept}(z, y), \mathrm{DeptMgr}(y, x)$$

We call $q(\mathbf{x})$ the *head* of q and $\varphi_1(\mathbf{x}, \mathbf{y}), \ldots, \varphi_m(\mathbf{x}, \mathbf{y})$ the *body* of q. Each $\varphi_i(\mathbf{x}, \mathbf{y})$ is an *atom* of q. A *self-join* of q is a pair of atoms that use the same relation symbol. For example, the above Manages CQ is *self-join-free* since each relation symbol is used only once. Lastly, we recall that a *union of conjunctive queries* (UCQ) is a logical disjunction of CQs with the same arity.

Integrity constraints. Let $\mathbf{S}$ be a schema. An *integrity constraint* is simply a Boolean query. A database D satisfies a set Δ of integrity constraints, denoted $D \models \Delta$, if D satisfies every constraint in Δ, that is, $D \models \gamma$ for all $\gamma \in \Delta$. When Δ is clear from the context, we say that D is *consistent* if $D \models \Delta$, and otherwise that it is *inconsistent*. Two sets of integrity constraints are *equivalent* if every database that satisfies one also satisfies the other.

A special case of an integrity constraint is a *Functional Dependency* (FD) $R : X \to Y$ where R is a relation symbol of $\mathbf{S}$ and X and Y are sets of attributes of R. A database D satisfies $R : X \to Y$ if every two R-facts that agree on (i.e., have the same value for every attribute of) X also agree on Y. When R is clear from the context, we may write just $X \to Y$.

Shapley value. As explained in the introduction, the Shapley value is a function for wealth distribution in cooperative games. The precise definition is as follows. Let L be a finite set of players. A cooperative game is a function $\mathcal{G} : \mathcal{P}(L) \to \mathbb{R}$, where $\mathcal{P}(L)$ is the power set of L. For $M \subseteq L$, the value $\mathcal{G}(M)$ represents a value, such as wealth, jointly obtained by M when the players of M cooperate. The Shapley value for the player a in the game $\mathcal{G}$ is, intuitively, the following quantity. Suppose that we form a random cooperating team by selecting players, one by one, uniformly and without replacement; what is the expected change of utility when a is selected? The exact definition is:

$$Shapley(L, \mathcal{G}, a) := \tag{1}$$
$$\frac{1}{|L|!} \sum_{\pi \in \Pi_L} (\mathcal{G}(\pi_a \cup \{a\}) - \mathcal{G}(\pi_a))$$

Here, Π_L is the set of all possible permutations over the players in L, and for each permutation π we denote by π_a the set of players that appear before a in the permutation.

Complexity concepts. Throughout the paper, we refer to the standard classes PTime, NP and coNP of decision problems. We will also consider the function classes FP, #P and FP#P. Recall that FP is the class of functions that can be computed in polynomial time, #P is the class of functions can be described as counting the accepting paths of a nondeterministic Turing machine on the given input, and FP#P is the class of functions that can be computed in polynomial time with an oracle to some function in #P.

We assume that integer numbers are represented in the usual binary way, and that non-integers are rational numbers represented using their numerator and denominator, where (n, d) stands for n/d.

Suppose that f is a numerical function that maps its input x to a number $f(x)$. A *Fully Polynomial-time Randomized Approximation Scheme* (FPRAS) for f is a randomized algorithm A that takes as input an instance x of f and an error $\epsilon > 0$, and returns an ϵ-approximation of $f(x)$ with probability at least $3/4$ (which is arbitrary and can be amplified to be arbitrarily close to 1 using standard techniques). More precisely, we distinguish between an *additive FPRAS*:

$$\Pr\big[f(x) - \epsilon \leq A(x) \leq f(x) + \epsilon\big] \; > \; \frac{3}{4}$$

and a *multiplicative FPRAS*:

$$\Pr\big[\frac{f(x)}{1+\epsilon} \leq A(x) \leq (1+\epsilon)f(x)\big] \; > \; \frac{3}{4}$$

3   Contribution to Database Queries

In this section, we survey the use of the Shapley value in the context of explaining query answers. We will focus on Boolean queries, to keep the presentation short, but the framework easily extends to queries with free variables, or to aggregate queries returning numerical values (e.g., as explained by Livshits et al. [43]). We first define the notions and then review recent work.

As mentioned in the introduction, a database D consists of a set D_x of *exogenous* facts, and a set D_n of *endogenous* facts. For a Boolean query q and endogenous fact $f \in D_\mathrm{n}$ of a database $D = D_\mathrm{n} \cup D_\mathrm{x}$, the goal is to measure the contribution of f to the result $q(D)$. To this end, we view a Boolean query as a numerical query that answers 1 if it is satisfied by the database and 0 otherwise. We then apply the Shapley value, where the players are the endogenous facts and the game function is the function $\mathcal{G}_{q, D_\mathrm{x}, D_\mathrm{n}} : \mathcal{P}(D_\mathrm{n}) \to \{0, 1\}$ mapping every subset $E \subseteq D_\mathrm{n}$ to the value $q(E \cup D_\mathrm{x}) - q(D_\mathrm{x})$. (The subtraction is applied to satisfy the requirement that the wealth function is zero on the empty set of players.) Applying Equation (1), *the Shapley value*

of f (in D for q), denoted $Shapley_{\mathsf{Q}}(q, D_n, D_x, f)$, is defined as follows:

$$Shapley_{\mathsf{Q}}(q, D_n, D_x, f) := Shapley(D_n, \mathcal{G}_{q,D_x,D_n}, f)$$

$$= \frac{1}{|D_n|!} \sum_{\pi \in \Pi_{D_n}} \left[q(D_x \cup \pi_f \cup \{f\}) - q(D_x \cup \pi_f) \right].$$

Intuitively, this value represents the contribution of f to the result of the query: the higher this value is, the more f contributes to satisfying q. Note that this value could be negative, in the case where q is not monotone. Moreover, the properties of the Shapley value imply that we always have $q(D) = q(D_x) + \sum_{f \in D_n} Shapley_{\mathsf{Q}}(q, D_n, D_x, f)$. In words, the contributions of all endogenous facts sum up to $q(D) - q(D_x)$; hence, the Shapley value states how the score $q(D)$ on the whole database is to be shared among the endogenous facts.

3.1 Exact Computation

In database theory, the computational complexity of a problem is often measured with what is called the *data complexity* [69], which is the complexity of the problem when a particular query q is fixed (there is, thus, one computational problem for each distinct query). Accordingly, past work [24, 43, 59] has studied the data complexity of exactly computing the Shapley value of a fact, as defined above, depending on the particular query q at hand. Formally, for an arbitrary Boolean query q and slightly abusing the notation, the computational problem $Shapley_{\mathsf{Q}}(q)$ is defined as follows:

PROBLEM:	$Shapley_{\mathsf{Q}}(q)$
PARAM:	Boolean query q
INPUT:	Database $D_x \cup D_n$, fact $f \in D_n$
GOAL:	Compute $Shapley_{\mathsf{Q}}(q, D_n, D_x, f)$

We aim for dichotomies in complexity, for classes of queries (e.g., CQs), that chart the boundary between polynomial-time cases and intractable cases. We report on such results in this section.

3.1.1 Reduction to Probabilistic Databases

Deutch et al. [24] showed that the calculation of the Shapley value can be reduced to *query answering in probabilistic databases*. Hence, this reduction yields a class of tractable queries. To explain that, we need some definitions.

A *Tuple-Independent Database* (*TID* for short) is a pair (D, π) consisting of a database D and a function π that maps each fact $f \in D$ to a probability $\pi(f) \in [0, 1]$. The TID (D, π) defines a probability distribution $\Pr_{D,\pi}$ on $\mathcal{P}(D)$, where each $D' \subseteq$

D has the probability $\Pr_{D,\pi}(D') =: \prod_{f \in D'} \pi(f) \times \prod_{f \in D \setminus D'} (1 - \pi(f))$. We evaluate a Boolean query q by calculating the probability that q is satisfied by the TID: $\Pr(q, (D, \pi)) = \sum_{D' \subseteq D} \Pr_{D,\pi}(D') \cdot q(D')$. The problem of *Probabilistic Query Evaluation* for q, or PQE(q) for short, is the following:

PROBLEM:	PQE(q)
PARAM:	Boolean query q
INPUT:	TID (D, π)
GOAL:	Compute $\Pr(q, (D, \pi))$

We can now state the result of Deutch et al. [24]:

THEOREM 1. *For every Boolean query q, the problem $Shapley_{\mathsf{Q}}(q)$ reduces in polynomial time to the problem PQE(q).*

Notice that this result is quite general, in that q can be an arbitrary Boolean query: a CQ, an FO query, an MSO, Datalog or RPQ query—it does not matter: if PQE(q) is tractable then $Shapley_{\mathsf{Q}}(q)$ is tractable as well. As it turns, a celebrated result by Dalvi and Suciu [19] provides a dichotomy on unions of conjunctive queries for PQE: either q is *safe* and PQE(q) is solvable in polynomial time (FP), or q is not safe and PQE(q) is $FP^{\#P}$-hard. Therefore, a direct corollary of Theorem 1 is that $Shapley_{\mathsf{Q}}(q)$ is in FP for all safe UCQs.

3.1.2 Calculation via Knowledge Compilation

Deutch et al. [24] also provide another route for solving this problem in practice, through *knowledge compilation*. We only sketch this approach here and refer to their work for more details. The idea is to first compute, for the query q and database D, the *lineage* (also called *provenance*) of q on D, which is a Boolean circuit that intuitively captures the dependence of the query answer on individual facts of D. This lineage is then transformed, using a *knowledge compiler* tool, into an equivalent Boolean circuit in restricted classes from knowledge compilation (namely, so-called *deterministic and decomposable Boolean circuits*), over which the authors of [24] design a polynomial-time algorithm to compute the Shapley values.

The queries that can be solved in FP with this method are a subset[1] of those that are captured with Theorem 1, so in terms of theoretical results nothing is gained here (apart from a polynomial of slightly lower degree). Nevertheless, this method allows them to use existing tools to solve the problem in practice instead of implementing everything from

---

[1]And it is unknown if this is a strict subset or not, see, e.g., [55].

scratch. They used in particular ProvSQL [62], a tool integrated into PostgreSQL that can perform lineage computation in various semirings, and the knowledge compiler c2d [21].

3.1.3 Hardness and Dichotomy

As we have discussed, Theorem 1 allows to capture a large class of tractable queries for the problem $Shapley_Q$. It is, however, still unknown whether this actually captures all tractable cases. In particular, it is unknown whether there also exists a general reduction in the other direction:

OPEN PROBLEM 1. *Does* $PQE(q)$ *reduce in polynomial time to* $Shapley_Q(q)$ *for all queries* q?

Combined with known results on the complexity of probabilistic databases, a positive answer to this question would then yield a complete dichotomy of $Shapley_Q$ for the class of all UCQs and, in fact, even for the more general class of queries *closed under homomorphism* [1].

For the class of CQs without self-joins, Livshits et al. [43] obtained a dichotomy. The tractability condition is that q is *hierarchical*: for all variables y and y' it holds that $A_y \subseteq A_{y'}$, or $A_{y'} \subseteq A_y$, or $A_y \cap A_{y'} = \emptyset$, where A_x is the set of atoms of q that use the variable x [20].

THEOREM 2. *Let* q *be a self-join-free CQ. If* q *is hierarchical, then* $Shapley_Q(q)$ *is in FP, otherwise it is* $FP^{\#P}$-*hard.*

We conclude this part with some comments on Theorem 2. First, the positive part is already captured by Theorem 1, as it is known that hierarchical CQs are safe for PQE on TIDs [19]. In contrast, the lower bound requires a reduction that is specifically crafted for the problem. In particular, it is not clear whether and how this reduction can be generalized to solve Open Problem 1, and there is no generalization of Theorem 2 for CQs with self-joins and UCQs. Second, the reduction for the lower bound of Theorem 2 requires the usage of both endogenous and exogenous facts, and it is still open whether the lower bound holds if we assume that all facts are endogenous. Finally, Reshef et al. [59] studied the complexity of $Shapley_Q$ for CQs with negated atoms (but still no self-joins), and established analogous dichotomy results under the restrictions that certain relations can contain only exogenous facts.

3.2 Approximation

As we have seen, computing the exact Shapley values of facts is not always tractable in theory. This naturally brings the question of which queries allow for approximate computation of these values with desired guarantees on the data complexity and the approximation ratio. We review in this section what is known on the approximability of the problems $Shapley_Q(q)$, in terms of FPRAS.

As pointed out in [43], by using the Chernoff-Hoeffding bound one can easily obtain an additive FPRAS for computing $Shapley_Q(q, D_n, D_x, f)$: this can be done for instance by sampling $O(\log(1/\delta)/\epsilon^2)$ permutations π of D_n and computing the average value of $q(D_x \cup \pi_f \cup \{f\}) - q(D_x \cup \pi_f)$, assuming that q itself is answerable in polynomial time. Note that this assumption is true for the most common query classes such as UCQs, Datalog, etc. Formally:

PROPOSITION 1. *If* q *can be answered in polynomial time, then* $Shapley_Q(q)$ *has an additive FPRAS.*

An additive FPRAS might be insufficient when the values to approximate are very small. Nevertheless, for a large class of queries, the Shapley values cannot in fact be too small [43]. Specifically, a (Boolean) query q is said to have the *gap property* if the Shapley value of any fact is either zero or is "not too small," that is, at least the reciprocal of a polynomial. Then, for a query with polynomial-time evaluation *and* the gap property, the additive FPRAS from Proposition 1 can be easily transformed into a multiplicative FPRAS. As it turns out, all UCQs have the gap property [43, Proposition 4.12].

PROPOSITION 2. *If* q *has polynomial-time evaluation and the gap property, then* $Shapley_Q(q)$ *admits a multiplicative FPRAS. This holds for all UCQs.*

Not all queries satisfy the gap property. For instance, some CQs with negated atoms (that we denote by $CQ^\neg$) may lack this property. In fact, by establishing a connection between the existence of a multiplicative FPRAS for $Shapley_Q(q)$ and what is called the *relevance problem for* q, Reshef et al. [59] were able to exhibit a $CQ^\neg$ that does not admit any multiplicative FPRAS (unless P = RP, which is widely believed to be false). So far, however, no dichotomy is known on the existence of a multiplicative FPRAS for $CQ^\neg$s (or even self-join–free $CQ^\neg$s). In fact, we do not know any example of a $CQ^\neg$ that does not have the gap property but nevertheless admits a multiplicative FPRAS.

3.3 Remarks

We conclude with several remarks on the Shapley value for database queries. Interestingly, the state of affairs for approximate Shapley computation contrasts with what happens with the SHAP

score in machine learning. Arenas et al. [4] have recently shown that the SHAP score does not admit a multiplicative FPRAS (under conventional assumptions in complexity theory), even for very simple monotone Boolean models (namely, monotone 2-DNFs) [4, Theorem 9].

A possible alternative to the computation of the Shapley value is to *rank* the endogenous facts according to their Shapley value, while possibly avoiding the actual computation of these values. Indeed, as far as we know, there could exist queries q such that $Shapley_\mathsf{Q}(q)$ is hard to approximate while the ranking problem is tractable, or vice-versa. Deutch et al. [24] proposed a fast heuristic to solve this problem, and this heuristic seems to work well empirically. Arad et al. [2] employed machine learning to learn this ranking. Yet, as of today, a formal study of the complexity of the ranking problem is lacking. For instance, we do not know whether there are queries with intractable Shapley but tractable ranking. Again, this contrasts with what is known on the SHAP-score from machine learning, as it has been shown that the ranking problem for this score is unlikely to be in BPP even for simple monotone Boolean models; see [4, Theorem 16].

Finally, Khalil and Kimelfeld [40] have recently studied the complexity of the Shapley value in the context of graphs with labeled edges. Their problem is similar to what we discussed in this section, except that the players are the (endogenous) edges and/or vertices of the graph, and the queries are Regular Path Queries (RPQs) and Conjunctions of RPQs (CRPQs). An RPQ is associated with a regular expression γ, and a pair (u, v) of vertices is an answer if any path from u to v conforms to γ. They established dichotomies for exact and approximate calculation of the Shapley value. For example, under conventional complexity assumptions, an RPQ has a multiplicative FPRAS if and only if the regular expression recognizes a finite language.

4 Contribution to Database Inconsistency

It is often wrong to assume that the database is clean of errors and conforms to our integrity constraints. Data might be the result of integrating unreliable sources (e.g., social media) that contain mistakes and conflicting information. Moreover, the content of the database may be produced by error-prone procedures (e.g., natural-language or image processing). Given this nature of data, *measures of database inconsistency* quantify the extent to which the database violates a given set of integrity constraints [8, 45, 48]. Inconsistency measures can be used, for example, to estimate the reliability of new

datasets [18] or to build progress indicators for data-cleaning systems [48]. Moreover, such measures can be used to attribute to database facts a level of responsibility to inconsistency, and so to prioritize facts in the explanation, inspection, or resolution of database inconsistency; this task is what we discuss in this part.

The attribution of responsibility to the inconsistency of the database relies on two main components: *(1)* an inconsistency measure *and (2)* a responsibility-sharing mechanism. For the former, we discuss several alternatives in the next section. For the latter, we use the Shapley value.

4.1 Inconsistency Measures

The measurement of the inconsistency of information has been extensively studied by the Knowledge Representation (KR) and Logic communities [27, 36, 38, 42, 66], where several different measures have been introduced. Some of these have been adapted to the database setting [45].

In general, an *inconsistency measure* $\mathcal{I}$ is a function that maps pairs (D, Δ) of a database D and a set Δ of integrity constraints (ICs) to numbers $\mathcal{I}(D, \Delta) \in [0, \infty)$. Intuitively, the higher $\mathcal{I}(D, \Delta)$ is, the stronger D violates Δ. We make only the (reasonable) assumption that $\mathcal{I}(D, \Delta)$ is zero whenever D is empty. Here again, we use the Shapley value framework, where this time the set of players is the whole database, and the utility is the function $\mathcal{G}_{D,\Delta,\mathcal{I}} : \mathcal{P}(D) \to \{0, 1\}$ that maps every subset $D' \subseteq D$ to the value $\mathcal{I}(D', \Delta)$. Hence, applying Equation (1), *the Shapley value of a fact f of D w.r.t.* Δ, denoted $Shapley_\mathsf{I}(D, \Delta, \mathcal{I}, f)$, is defined as

$$Shapley_\mathsf{I}(D, \Delta, \mathcal{I}, f) := Shapley(D, \mathcal{G}_{D,\Delta,\mathcal{I}}, f)$$
$$= \frac{1}{|D|!} \sum_{\pi \in \Pi_D} \left(\mathcal{I}(\pi_f \cup \{f\}, \Delta) - \mathcal{I}(\pi_f, \Delta) \right).$$

Following Livshits and Kimelfeld [45], here we do not distinguish between exogenous and endogenous facts—all facts are considered endogenous (and all definitions naturally extend to account for exogenous facts).

Livshits and Kimelfeld [45] studied the Shapley value for several inconsistency measures that were previously explored in the context of databases and knowledge bases. (See [48] for a study of the behavior of these measures from both a practical and a theoretical perspective.) In the following definitions, D denotes a database and Δ a set of ICs.

1. $\mathcal{I}_\mathsf{d}$, called the *drastic measure* [66] takes the value 1 if the database is inconsistent and the value 0 otherwise.

2. $\mathcal{I}_{\text{MI}}$ counts the *minimal inconsistent subsets* of the database [37,38]. In notation, $\mathcal{I}_{\text{MI}}(D, \Delta) := |\text{MI}(D, \Delta)|$ where $\text{MI}(D, \Delta)$ is the set of all inconsistent subsets $E \subseteq D$ such that $E' \models \Delta$ for all $E' \subset E$.

3. $\mathcal{I}_{\text{P}}$ counts the *problematic facts*, where a fact is *problematic* if it belongs to a minimal inconsistent subset [28]. In notation, $\mathcal{I}_{\text{P}}(D, \Delta) := |\bigcup_{E \in \text{MI}(D, \Delta)} E|$.

4. $\mathcal{I}_{\text{R}}$ is the minimal number of facts that should be deleted from the database in order to satisfy Δ [8, 26, 29]. In notation, $\mathcal{I}_{\text{R}}(D, \Delta) := \min_{E \subseteq D, D \setminus E \models \Delta}(|E|)$.

5. $\mathcal{I}_{\text{MC}}$ counts the *maximal consistent subsets* [28, 30] (also called *subset repairs* [5]). In notation, $\mathcal{I}_{\text{MC}}(D, \Delta) := |\text{MC}(D, \Delta)|$ where $\text{MC}(D, \Delta)$ is the set of all consistent subsets $E \subseteq D$ such that $E' \not\models \Delta$ whenever $E \subset E'$.

Intuitively, the measure I_d is simply an indicator of inconsistency. The measure $\mathcal{I}_{\text{MI}}$ counts the violations (i.e., the minimal sets of facts that jointly violate the constraints) and the measure $\mathcal{I}_{\text{P}}$ counts the facts involved in such violations (and a fact is counted once even if it occurs in multiple violations). For both measures, the higher the number is, the more the constraints are violated; hence, the more inconsistent the database is. The measure $\mathcal{I}_{\text{R}}$ quantifies the distance of the database from a consistent one—the more facts we have to remove to obtain consistency, the higher the measure is. Finally, the measure $\mathcal{I}_{\text{MC}}$ counts the subset repairs; that is, all the different ways to obtain a consistent database by deleting a minimal set of facts—the more repairs there are, the more inconsistent the database is.

4.2 Complexity of Exact Computation

We proceed to discuss the complexity of computing the value $Shapley_{\text{I}}(D, \Delta, \mathcal{I}, f)$ for each one of the aforementioned inconsistency measures $\mathcal{I}$, as studied by Livshits and Kimelfeld [45]. Their study considered the case where Δ is a set of FDs. We consider again the data complexity of computing the Shapley value; this time, the schema, set of FDs, and inconsistency measure are considered fixed, and the input consists of a database D and a fact f. Accordingly, and again slightly abusing notation, we denote $Shapley_{\text{I}}(\Delta, \mathcal{I})$ the corresponding computational problems.

PROBLEM:	$Shapley_{\text{I}}(\Delta, \mathcal{I})$
PARAM:	Set Δ of ICs, inconsistency m. $\mathcal{I}$
INPUT:	Database D and fact $f \in D$
GOAL:	Compute $Shapley_{\text{I}}(D, \Delta, \mathcal{I}, f)$

It turns our that each of the inconsistency measures entails quite a unique picture of complexity.

The measure $\mathcal{I}_{\text{d}}$. While the drastic measure is the simplest one conceptually, it might be intractable to compute the Shapley value of a fact w.r.t. $\mathcal{I}_{\text{d}}$ even for simple FD sets. Livshits and Kimelfeld [45] established a dichotomy in data complexity for this measure, based on the definition of left-hand-side chain [44]. An FD set Δ has a left-hand-side chain (lhs chain, for short) if for every two FDs $X \to Y$ and $X' \to Y'$ in Δ, either $X \subseteq X'$ or $X' \subseteq X$.

THEOREM 3. *Let Δ be a set of FDs. If Δ is equivalent to an FD set with an lhs chain, then $Shapley_{\text{I}}(\Delta, \mathcal{I}_{\text{d}})$ is in FP. Otherwise, the problem is $\text{FP}^{\#\text{P}}$-complete.*

Theorem 3 implies, for example, that the Shapley value can be computed efficiently for the set $\{A \to B, AC \to D\}$, but not for $\{A \to B, B \to A\}$. As a proof technique, they show hardness directly only for $\{A \to B, B \to A\}$. For any other intractable FD set, hardness is established via the so called *fact-wise reductions* [41]. Such reductions essentially map databases over one schema and set of ICs to databases over another schema and set of ICs while preserving (in)consistency. The positive side of Theorem 3 is established via dynamic programming.

We note that the tractability criterion of Theorem 3 is the same as the tractability criterion for the problem of counting subset repairs (or, equivalently, maximal consistent subsets), and is decidable in polynomial time [44].

The measure $\mathcal{I}_{\text{MI}}$. For a set Δ of FDs, it is easy to see that $\mathcal{I}_{\text{MI}}$ simply counts the pairs of facts of the database that jointly violate the FDs. Hence, for a fact f and some permutation π, we have that $\mathcal{I}_{\text{MI}}(\pi_f \cup \{f\}, \Delta) - \mathcal{I}_{\text{MI}}(\pi_f, \Delta)$ is precisely the number of facts in π_f that are in conflict with f. This simple observation implies the tractability of $\mathcal{I}_{\text{MI}}$.

THEOREM 4. [45] *$Shapley_{\text{I}}(\Delta, \mathcal{I}_{\text{MI}})$ is in FP for every set Δ of FDs.*

The measure $\mathcal{I}_{\text{P}}$. For this measure, we have that $\mathcal{I}_{\text{P}}(\pi_f \cup \{f\}, \Delta) - \mathcal{I}_{\text{P}}(\pi_f, \Delta)$ is the number of facts in π_f that: *(1)* are in conflict with f, and *(2)* are

Table 1: The complexity of the (exact ; approximate) Shapley value of different inconsistency measures.

	lhs chain	no lhs chain, PTime cardinality repair	other
$\mathcal{I}_{\mathsf{d}}$	PTime	$\mathrm{FP}^{\#\mathrm{P}}$-complete ; FPRAS	
$\mathcal{I}_{\mathsf{MI}}$	PTime		
$\mathcal{I}_{\mathsf{P}}$	PTime		
$\mathcal{I}_{\mathsf{R}}$	PTime	? ; FPRAS	NP-hard [46] ; no FPRAS
$\mathcal{I}_{\mathsf{MC}}$	PTime	$\mathrm{FP}^{\#\mathrm{P}}$-complete [44] ; ?	

not in conflict with any other fact of π_f (as, otherwise, they are considered "problematic" already before adding the fact f to the picture). Based on this observation, we conclude the following.

THEOREM 5. [45] $Shapley_{\mathsf{I}}(\Delta, \mathcal{I}_{\mathsf{P}})$ *is in FP for every set Δ of FDs.*

The measure $\underline{\mathcal{I}_{\mathsf{R}}}$. The measure $\mathcal{I}_{\mathsf{R}}$ is the only one for which we lack the full complexity picture; however, it is known that there are substantial sets of FDs for which this measure is tractable and others for which it is intractable. The hard cases follow from a prior work on the related problem of computing the cost of a *cardinality repair* (i.e., the minimal number of facts to remove to obtain consistency), as we explain next.

Livshits et al. [46] established a dichotomy for finding a cardinality repair, where the tractability criterion is based on a polynomial-time algorithm, Simplify, that simplifies the FD set in an iterative manner until no further simplification can be applied. If the result of Simplify(Δ) is an empty FD set, then the problem is solvable in polynomial time; otherwise, it is NP-hard. Now, one of the basic properties of the Shapley value is "efficiency"—the sum of the Shapley values over all the players equals the total wealth [63]. This property implies that for an inconsistency measure $\mathcal{I}$, it holds that

$$\sum_{f \in D} Shapley_{\mathsf{I}}(D, \Delta, \mathcal{I}, f) = \mathcal{I}(D, \Delta).$$

Thus, whenever the measure itself is hard to compute, so is the Shapley value of facts. This leads to the following result.

THEOREM 6. *Let Δ be a set of FDs. If the procedure Simplify(Δ) of Livshits et al. [46] returns a nonempty set, then $Shapley_{\mathsf{I}}(\Delta, \mathcal{I}_{\mathsf{R}})$ is NP-hard.*

As in the case of the drastic measure, it can be shown, via a dynamic programming algorithm, that for an FD set with an lhs chain, the measure $\mathcal{I}_{\mathsf{R}}$ is tractable.

THEOREM 7. *Let Δ be a set of FDs. If Δ is equivalent to an FD set with an lhs chain, then $Shapley_{\mathsf{I}}(\Delta, \mathcal{I}_{\mathsf{R}})$ is in FP.*

It remains unknown what is the complexity of the problem for FD sets that fall outside the two cases covered by Theorems 6 and 7.

OPEN PROBLEM 2. *What is the data complexity of $Shapley_{\mathsf{I}}(\Delta, \mathcal{I}_{\mathsf{R}})$ for any set Δ of FDs such that Δ has no lhs chain (up to equivalence) and Simplify(Δ) returns an empty set?*

In particular, the problem is open for the FD set $\{A \to B, B \to A\}$ over the relation symbol $R(A, B)$.

The measure $\underline{\mathcal{I}_{\mathsf{MC}}}$. As we explained for the previous measure, the efficiency property of the Shapley value allows us to conclude that the problem $Shapley_{\mathsf{I}}(\Delta, \mathcal{I}_{\mathsf{MC}})$ is hard whenever it is hard to calculate $\mathcal{I}_{\mathsf{MC}}(D, \Delta)$. Livshits et al. [47] have shown that the subset repairs (i.e., maximal consistent subsets) can be counted in polynomial time for FD sets with an lhs chain (up to equivalence), and is #P-complete for any other FD set. Hence, an lhs chain is a necessary condition for tractability. As in the case of the measures $\mathcal{I}_{\mathsf{d}}$ and $\mathcal{I}_{\mathsf{R}}$, it can be shown that this is also a sufficient condition using a dynamic programming algorithm. Hence, the following holds.

THEOREM 8. [45] *Let Δ be a set of FDs. If Δ is equivalent to an FD set with an lhs chain, then $Shapley_{\mathsf{I}}(\Delta, \mathcal{I}_{\mathsf{MC}})$ is in FP. Otherwise, the problem is $\mathrm{FP}^{\#\mathrm{P}}$-complete.*

The complexity results of this section are summarized in Table 1. (The results on approximation are discussed in the next section.)

4.3 Complexity of Approximation

We have seen that, as far as exact computation is concerned, $\mathcal{I}_{\mathsf{MI}}$ and $\mathcal{I}_{\mathsf{P}}$ are tractable for every set of FDs, but the measures $\mathcal{I}_{\mathsf{d}}$, $\mathcal{I}_{\mathsf{R}}$, and $\mathcal{I}_{\mathsf{MC}}$ can be

intractable even for simple sets of FDs. Therefore, we now discuss the approximate computation of $Shapley_I$. Again, the complexity results are of Livshits and Kimelfeld [45].

As in the case of $Shapley_Q$, using the Chernoff-Hoeffding bound and a Monte-Carlo approach, one can easily obtain an additive FPRAS for computing $Shapley_I(D, \Delta, \mathcal{I}, f)$, assuming that $\mathcal{I}(D, \Delta)$ can be computed in polynomial time, given D. A multiplicative FPRAS can be obtained using the same procedure if a corresponding "gap" property holds, meaning that when the Shapley value is not zero, it is guaranteed to be "large enough".

For instance, in the case of the drastic measure, it can be shown that for all databases D and facts f, $Shapley_I(D, \Delta, \mathcal{I}_d, f)$ is either zero or at least $\frac{1}{|D| \cdot (|D|-1)}$. Since $\mathcal{I}_d(D, \Delta)$ can be computed in polynomial time for every Δ, we conclude that:

PROPOSITION 3. $Shapley_I(\Delta, \mathcal{I}_d)$ has an additive and a multiplicative FPRAS for every set Δ of FDs.

Regarding $\mathcal{I}_R$, when it can be computed in polynomial time (i.e., when $\mathsf{Simplify}(\Delta) = \emptyset$), we can similarly show that $Shapley_I(\Delta, \mathcal{I}_R)$ has both an additive and a multiplicative FPRAS. The gap property holds here as well: $Shapley_I(D, \Delta, \mathcal{I}_R, f) = 0$ or $Shapley_I(D, \Delta, \mathcal{I}_R, f) \geq \frac{1}{|D| \cdot (|D|-1)}$. In contrast, when $\mathcal{I}_R$ is intractable, it is not only NP-hard, but also APX-complete. Consequently, there exists a polynomial-time constant-ratio approximation for $\mathcal{I}_R(D, \Delta)$, but for some $\epsilon > 1$ there is no (randomized) ϵ-approximation or else NP = RP. Then, the fact that $Shapley_I(D, \Delta, \mathcal{I}_R, f) \geq 0$ implies that the existence of a multiplicative FPRAS for the value $Shapley_I(D, \Delta, \mathcal{I}_R, f)$ would imply the existence of a multiplicative FPRAS for $\mathcal{I}_R(D, \Delta)$ via the efficiency property of the Shapley value. Therefore, the following holds.

THEOREM 9. Let Δ be a set of FDs.

1. If $\mathsf{Simplify}(\Delta) = \emptyset$, then $Shapley_I(\Delta, \mathcal{I}_R)$ has both an additive and a multiplicative FPRAS.

2. Otherwise, $Shapley_I(\Delta, \mathcal{I}_R)$ has neither a multiplicative nor an additive FPRAS, assuming $RP \neq NP$.

Interestingly, it is still unknown whether there is *any* polynomial-time constant-ratio multiplicative approximation for $Shapley_I(D, \Delta, \mathcal{I}_R, f)$ for any FD set in the intractable side of Theorem 9.

OPEN PROBLEM 3. *For any set Δ of FDs such that $\mathsf{Simplify}(\Delta)$ is nonempty, is there a polynomial-time constant-ratio multiplicative approximation for $Shapley_I(D, \Delta, \mathcal{I}_R, f)$?*

Finally, the complexity picture for the approximate computation of the Shapley value w.r.t. $\mathcal{I}_{MC}$ is rather incomplete. This is because counting the maximal consistent subsets w.r.t. $\{A \rightarrow B, B \rightarrow A\}$ over $R(A, B)$ is the same as counting the maximal matchings of a bipartite graph. As the values $Shapley_I(D, \Delta, \mathcal{I}_{MC}, f)$ are nonnegative and sum up to the number of maximal consistent subset (via the efficiency property), we conclude that an FPRAS for $Shapley_I(D, \Delta, \mathcal{I}_{MC}, f)$ implies an FPRAS for the number of maximal matchings. To the best of our knowledge, existence of the latter is an open problem. The same holds for any FD set Δ' that is not equivalent to an FD set with an lhs chain, since there is a fact-wise reduction from Δ to such Δ' [44]. Hence, the following remains unknown.

OPEN PROBLEM 4. *For any FD set Δ that has no lhs chain, is there a multiplicative FPRAS for $Shapley_I(\Delta, \mathcal{I}_{MC})$?*

Actually, Open Problem 1 is solved in one special case. It has been recently shown that the problem of counting the maximal consistent subsets for the FD set $\{A \rightarrow B, C \rightarrow D\}$ over $R(A, B, C, D)$ does not admit an FPRAS, unless NP = RP [16]. Therefore, we conclude that the same holds for the problem of computing $Shapley_I(D, \{A \rightarrow B, C \rightarrow D\}, \mathcal{I}_{MC}, f)$.

As said earlier, Table 1 summarizes the complexity results for both the exact and approximate variants of the problems we studied here.

4.4 Contribution to Cleaning Actions

We presented the theoretical results of Livshits and Kimelfeld [45] on the contribution of facts to inconsistency. The application of the Shapley value to machinery for data repairing has been proposed and studied by Deutch et al. [23] from a different angle and a practical treatment. They aimed for explaining the individual actions of a data-repairing system, and particularly its decision to change a given entry in the database. They used the Shapley value to quantify the contribution of *relation cells* and the *ICs* to the decision to change a value. Hence, cells and ICs are the players in the coalitional game of causing the repairing algorithm to change a given cell. While their solutions do not have theoretical guarantees, they give heuristics that apply for a general setting: ICs can be any set of *denial constraints* (that widely generalize FDs) and repairing can be done by an arbitrary *black-box* algorithm.

5 Concluding Remarks

We presented recent work on the application of the Shapley value to database tasks. Our focus has

been on the theoretical analysis of the complexity of the underlying computational problems, and we discussed mainly two of them: computing the contribution of a fact to a query answer, and computing its contribution to inconsistency. While there has been considerable research effort on the practical realization of the first challenge [2, 22, 24, 25], the practical aspects of the contribution to inconsistency are yet to be explored (with the exception of Deutch et al. [23] that, as we explained, studied a problem that is related but different). Hence, many problems remain for future research, both on the theoretical side (e.g., the open problems we listed throughout the manuscript) and the practical one.

Notably, the framework for applying the Shapley value to measuring contribution to query answers [43] has been applied for proposing new bibliometric metrics [25]. The authors did not use the algorithms proposed by Deutch et al. [24] (which are the most efficient to date, to the best of our knowledge), but instead computed the values directly from the definition. In this context, we believe that it is important to facilitate the use of existing algorithms by incorporating them into Postgres extensions such as ProvSQL [62].

6 References

[1] A. Amarilli. Uniform reliability for unbounded homomorphism-closed graph queries. In *ICDT*, volume 255 of *LIPIcs*, pages 14:1–14:17, 2023.

[2] D. Arad, D. Deutch, and N. Frost. LearnShapley: Learning to predict rankings of facts contribution based on query logs. In *CIKM*, pages 4788–4792, 2022.

[3] M. Arenas, P. Barceló, L. Bertossi, and M. Monet. The tractability of SHAP-score-based explanations for classification over deterministic and decomposable boolean circuits. In *AAAI*, pages 6670–6678, 2021.

[4] M. Arenas, P. Barceló, L. Bertossi, and M. Monet. On the complexity of SHAP-score-based explanations: Tractability via knowledge compilation and non-approximability results. *Journal of Machine Learning Research*, 24(63):1–58, 2023.

[5] M. Arenas, L. E. Bertossi, and J. Chomicki. Consistent query answers in inconsistent databases. In *PODS*, pages 68–79. ACM Press, 1999.

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

[7] L. Bertossi. Database repairs and consistent query answering: Origins and further developments. In

D. Suciu, S. Skritek, and C. Koch, editors, *PODS*, pages 48–58. ACM, 2019.

[8] L. Bertossi. Repair-based degrees of database inconsistency. In *LPNMR*, volume 11481 of *LNCS*, pages 195–209. Springer, 2019.

[9] L. Bertossi. Specifying and computing causes for query answers in databases via database repairs and repair-programs. *Knowl. Inf. Syst.*, 63(1):199–231, 2021.

[10] L. Bertossi. Attribution-scores and causal counterfactuals as explanations in artificial intelligence. In *Bertossi, L., Xiao, G. (eds.) Reasoning Web. Causality, Explanations and Declarative Knowledge. Springer LNCS 13759*, pages 1–23, 2023.

[11] L. Bertossi, J. Li, M. Schleich, D. Suciu, and Z. Vagena. Causality-based explanation of classification outcomes. In *DEEM@SIGMOD*, pages 6:1–6:10. ACM, 2020.

[12] L. Bertossi and B. Salimi. Causes for query answers from databases: Datalog abduction, view-updates, and integrity constraints. *Int. J. Approx. Reason.*, 90:226–252, 2017.

[13] L. Bertossi and B. Salimi. From causes for database queries to repairs and model-based diagnosis and back. *Theory Comput. Syst.*, 61(1):191–232, 2017.

[14] P. Buneman and W. Tan. Data provenance: What next? *SIGMOD Rec.*, 47(3):5–16, 2018.

[15] N. Burkart and M. F. Huber. A survey on the explainability of supervised machine learning. *J. Artif. Intell. Res.*, 70:245–317, 2021.

[16] M. Calautti, E. Livshits, A. Pieris, and M. Schneider. Counting database repairs entailing a query: The case of functional dependencies. In *PODS*, pages 403–412. ACM, 2022.

[17] H. Chockler and J. Y. Halpern. Responsibility and blame: A structural-model approach. *J. Artif. Intell. Res.*, 22:93–115, 2004.

[18] L. Cholvy, L. Perrussel, W. Raynaut, and J.-M. Thévenin. Towards consistency-based reliability assessment. In *AAMAS*, pages 1643–1644. ACM, 2015.

[19] N. Dalvi and D. Suciu. The dichotomy of probabilistic inference for unions of conjunctive queries. *Journal of the ACM (JACM)*, 59(6):1–87, 2013.

[20] N. N. Dalvi, C. Ré, and D. Suciu. Probabilistic databases: Diamonds in the dirt. *Commun. ACM*, 52(7):86–94, 2009.

[21] A. Darwiche. New advances in compiling CNF to decomposable negation normal form. In *Proceedings of ECAI*, pages 328–332. Citeseer, 2004.

[22] S. B. Davidson, D. Deutch, N. Frost, B. Kimelfeld, O. Koren, and M. Monet. ShapGraph: An holistic view of explanations through provenance graphs and Shapley values. In *SIGMOD Conference*, pages 2373–2376. ACM, 2022.

[23] D. Deutch, N. Frost, A. Gilad, and O. Sheffer. Explanations for data repair through Shapley values. In *CIKM*, pages 362–371. ACM, 2021.

[24] D. Deutch, N. Frost, B. Kimelfeld, and M. Monet. Computing the Shapley value of facts in query answering. In *SIGMOD*, pages 1570–1583, 2022.

[25] D. Dosso, S. B. Davidson, and G. Silvello. Credit distribution in relational scientific databases. *Information Systems*, 109:102060, 2022.

[26] O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. *J. ACM*, 45(4):653–750, 1998.

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

[28] J. Grant and A. Hunter. Measuring consistency gain and information loss in stepwise inconsistency

resolution. In *ECSQARU*, volume 6717 of *LNCS*, pages 362–373. Springer, 2011.

[29] J. Grant and A. Hunter. Distance-based measures of inconsistency. In *ECSQARU*, volume 7958 of *LNCS*, pages 230–241. Springer, 2013.

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

[31] T. J. Green and V. Tannen. The semiring framework for database provenance. In E. Sallinger, J. V. den Bussche, and F. Geerts, editors, *PODS*, pages 93–99. ACM, 2017.

[32] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi. A survey of methods for explaining black box models. *ACM Comput. Surv.*, 51(5):93:1–93:42, 2019.

[33] J. Y. Halpern. *Actual Causality*. MIT Press, 2016.

[34] J. Y. Halpern and J. Pearl. Causes and explanations: A structural-model approach. part i: Causes. *British Journal for the Philosophy of Science*, 56(4):843–887, 2005.

[35] J. Y. Halpern and J. Pearl. Causes and explanations: A structural-model approach. part ii: Explanations. *British Journal for the Philosophy of Science*, 56(4):889–911, 2005.

[36] A. Hunter and S. Konieczny. Shapley inconsistency values. In *KR*, pages 249–259. AAAI Press, 2006.

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

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

[39] G. W. Imbens and D. B. Rubin. *Causal Inference for Statistics, Social, and Biomedical Sciences: An Introduction*. Cambridge University Press, 2015.

[40] M. Khalil and B. Kimelfeld. The complexity of the Shapley value for regular path queries. *arXiv preprint arXiv:2212.07720*, 2022.

[41] B. Kimelfeld, J. Vondrák, and R. Williams. Maximizing conjunctive views in deletion propagation. In *PODS*, pages 187–198. ACM, 2011.

[42] S. Konieczny, J. Lang, and P. Marquis. Quantifying information and contradiction in propositional logic through test actions. In *IJCAI*, pages 106–111. Morgan Kaufmann, 2003.

[43] E. Livshits, L. Bertossi, B. Kimelfeld, and M. Sebag. The Shapley value of tuples in query answering. *Log. Methods Comput. Sci.*, 17(3), 2021.

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

[45] E. Livshits and B. Kimelfeld. The Shapley value of inconsistency measures for functional dependencies. *Log. Methods Comput. Sci.*, 18(2), 2022.

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

[47] E. Livshits, B. Kimelfeld, and J. Wijsen. Counting subset repairs with functional dependencies. *J. Comput. Syst. Sci.*, 117:154–164, 2021.

[48] E. Livshits, R. Kochirgan, S. Tsur, I. F. Ilyas, B. Kimelfeld, and S. Roy. Properties of inconsistency measures for databases. In *SIGMOD*, pages 1182–1194. ACM, 2021.

[49] S. M. Lundberg, G. G. Erion, H. Chen, A. J. DeGrave, J. M. Prutkin, B. Nair, R. Katz, J. Himmelfarb, N. Bansal, and S. Lee. From local explanations to global understanding with explainable AI for trees. *Nat. Mach. Intell.*, 2(1):56–67, 2020.

[50] S. M. Lundberg and S. Lee. A unified approach to interpreting model predictions. In *NIPS*, pages 4765–4774, 2017.

[51] R. T. B. Ma, D. Chiu, J. C. Lui, V. Misra, and D. Rubenstein. Internet economics: The use of Shapley value for ISP settlement. *IEEE/ACM Trans. Netw.*, 18(3):775–787, 2010.

[52] A. Meliou, W. Gatterbauer, K. F. Moore, and D. Suciu. The complexity of causality and responsibility for query answers and non-answers. *Proc. VLDB Endow.*, 4(1):34–45, 2010.

[53] D. Minh, H. Wang, Y. Li, and T. Nguyen. Explainable artificial intelligence: a comprehensive review. *Artificial Intelligence Review*, 55, 11 2021.

[54] C. Molnar. *Interpretable Machine Learning*. https://christophm.github.io/interpretable-ml-book/, 2019.

[55] M. Monet. Solving a special case of the intensional vs extensional conjecture in probabilistic databases. In *Proceedings of PODS*, pages 149–163, 2020.

[56] S. Moretti, F. Patrone, and S. Bonassi. The class of microarray games and the relevance index for genes. *Top*, 15(2):256–280, 2007.

[57] R. Narayanam and Y. Narahari. A Shapley value-based approach to discover influential nodes in social networks. *IEEE Trans Autom. Sci. Eng.*, 8(1):130–147, 2011.

[58] J. Pearl. *Causality: Models, Reasoning and Inference*. Cambridge University Press, 2nd edition, 2009.

[59] A. Reshef, B. Kimelfeld, and E. Livshits. The impact of negation on the complexity of the Shapley value in conjunctive queries. In *PODS*, pages 285–297. ACM, 2020.

[60] A. E. Roth, editor. *The Shapley value : essays in honor of Lloyd S. Shapley*. Cambridge University Press, 1988.

[61] B. Salimi, L. Bertossi, D. Suciu, and G. V. den Broeck. Quantifying causal effects on query answering in databases. In *TaPP*. USENIX Association, 2016.

[62] P. Senellart, L. Jachiet, S. Maniu, and Y. Ramusat. ProvSQL: Provenance and probability management in PostgreSQL. *Proc. VLDB Endow.*, 11(12):2034–2037, 2018.

[63] L. S. Shapley. A value for n-person games. In H. W. Kuhn and A. W. Tucker, editors, *Contributions to the Theory of Games II*, pages 307–317. Princeton University Press, Princeton, 1953.

[64] P. Struss. Model-based problem solving. In *Handbook of Knowledge Representation*, 2008.

[65] D. Suciu, D. Olteanu, C. Ré, and C. Koch. *Probabilistic Databases*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011.

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

[67] T. van Campen, H. Hamers, B. Husslage, and R. Lindelauf. A new approximation method for the Shapley value applied to the WTC 9/11 terrorist attack. *Soc. Netw. Anal. Min.*, 8(1):3:1–3:12, 2018.

[68] G. Van den Broeck, A. Lykov, M. Schleich, and D. Suciu. On the tractability of SHAP explanations. *J. Artif. Intell. Res.*, 74:851–886, 2022.

[69] M. Y. Vardi. The complexity of relational query languages. In *STOC*, pages 137–146. ACM, 1982.