The Shapley Value of Tuples in Query Answering

Ester Livshits
Technion, Haifa, Israel
esterliv@cs.technion.ac.il

Leopoldo Bertossi[1]
Univ. Adolfo Ibañez, Santiago, Chile
RelationalAI Inc., Toronto, Canada
leopoldo.bertossi@uai.cl

Benny Kimelfeld
Technion, Haifa, Israel
bennyk@cs.technion.ac.il

Moshe Sebag
Technion, Haifa, Israel
moshesebag@campus.technion.ac.il

Abstract

We investigate the application of the Shapley value to quantifying the contribution of a tuple to a query answer. The Shapley value is a widely known numerical measure in cooperative game theory and in many applications of game theory for assessing the contribution of a player to a coalition game. It has been established already in the 1950s, and is theoretically justified by being the very single wealth-distribution measure that satisfies some natural axioms. While this value has been investigated in several areas, it received little attention in data management. We study this measure in the context of conjunctive and aggregate queries by defining corresponding coalition games. We provide algorithmic and complexity-theoretic results on the computation of Shapley-based contributions to query answers; and for the hard cases we present approximation algorithms.

2012 ACM Subject Classification Theory of computation → Data provenance

Keywords and phrases Shapley value, query answering, conjunctive queries, aggregate queries

Digital Object Identifier 10.4230/LIPIcs.ICDT.2020.20

Related Version A full version of the paper is available at `https://arxiv.org/abs/1904.08679`.

Funding The work of Ester Livshits, Benny Kimelfeld and Moshe Sebag was supported by the Israel Science Foundation (ISF), grants 1295/15 and 768/19, and the Deutsche Forschungsgemeinschaft (DFG) project 412400621 (DIP program). The work of Ester Livshits was also supported by the Technion Hiroshi Fujiwara Cyber Security Research Center and the Israel Cyber Bureau.

1 Introduction

The Shapley value is named after Lloyd Shapley who introduced the value in a seminal 1952 article [33]. He considered a *cooperative game* that is played by a set A of players and is defined by a *wealth function* v that assigns, to each coalition $S \subseteq A$, the wealth $v(S)$. For instance, in our running example the players are researchers, and $v(S)$ is the total number of citations of papers with an author in S. As another example, A might be a set of politicians, and $v(S)$ the number of votes that a poll assigns to the party that consists of the candidates in S. The question is how to distribute the wealth $v(A)$ among the players, or from a different perspective, how to quantify the contribution of each player to the overall wealth.

[1] Member of Millenium Institute on Foundations of Data (IMFD), Chile.

For example, the removal of a researcher r may have zero impact on the overall number of citations, since each paper has co-authors from A. Does it mean that r has no contribution at all? What if the removal in turns of *every* individual author has no impact? Shapley considered distribution functions that satisfy a few axioms of good behavior. Intuitively, the axioms state that the function should be invariant under isomorphism, the sum over all players should be equal to the total wealth, and the contribution to a sum of wealths is equal to the sum of separate contributions. Quite remarkably, Shapley has established that there is a *single* such function, and this function has become known as the *Shapley value*.

The Shapley value is informally defined as follows. Assume that we select players one by one, randomly and without replacement, starting with the empty set. Whenever we select the player p, its addition to the set S of players selected so far may cause a change in wealth from $v(S)$ to $v(S \cup \{p\})$. The Shapley value of p is the expectation of change that p causes in this probabilistic process.

The Shapley value has been applied in various areas and fields beyond cooperative game theory (e.g., [1, 2]), such as bargaining foundations in economics [14], takeover corporate rights in law [27], pollution responsibility in environmental management [20, 29], influence measurement in social network analysis [26], and utilization of multiple Internet Service Providers (ISPs) in networks [22]. Closest to database manegement is the application of the Shapley value to attributing a level of *inconsistency* to a statement in an inconsistent knowledge base [17, 36]; the idea is natural: as wealth, adopt a measure of inconsistency for a set of logical sentences [12], and then associate to each sentence its Shapley value.

In this paper, we apply the Shapley value to quantifying the contribution of database facts (tuples) to query results. As in previous work on quantification of contribution of facts [24, 31], we view the database as consisting of two types of facts: *endogenous* facts and *exogenous* facts. Exogenous facts are taken as given (e.g., inherited from external sources) without questioning, and are beyond experimentation with hypothetical or counterfactual scenarios. On the other hand, we may have control over the endogenous facts, and these are the facts for which we reason about existence and marginal contribution. Our focus is on queries that can be viewed as mapping databases to numbers. These include Boolean queries (mapping databases to zero and one) and aggregate queries (e.g., count the number of tuples in a multiway join). As a cooperative game, the endogenous facts take the role of the players, and the result of the query is the wealth. The core computational problem for a query is then: given a database and an endogenous fact, compute the Shapley value of the fact.

We study the complexity of computing the Shapley value for Conjunctive Queries (CQs) and aggregate functions over CQs. Our main results are as follows. We first establish a dichotomy in data complexity for the class of Boolean CQs without self-joins. Interestingly, our dichotomy is the same as that of query inference in tuple-independent probabilistic databases [9]: if the CQ is hierarchical, then the problem is solvable in polynomial time, and otherwise, it is $\mathrm{FP}^{\#\mathrm{P}}$-complete (i.e., complete for the intractable class of polynomial-time algorithms with an oracle to, e.g., a counter of the satisfying assignments of a propositional formula). The proof, however, is more challenging than that of Dalvi and Suciu [9], as the Shapley value involves coefficients that do not seem to easily factor out. Since the Shapley value is a probabilistic expectation, we show how to use the linearity of expectation to extend the dichotomy to arbitrary summations over CQs without self-joins. For non-hierarchical queries (and, in fact, all unions of CQs), we show that both Boolean and summation versions are efficiently approximable (i.e., have a multiplicative FPRAS) via Monte Carlo sampling.

The general conclusion is that computing the exact Shapley value is notoriously hard, but the picture is optimistic if approximation is allowed under strong guarantees of error boundedness. Our results immediately generalize to non-Boolean CQs and group-by operators, where the goal is to compute the Shapley value of a fact to each tuple in the answer of a query. For aggregate functions other than summation (where we cannot apply the linearity of expectation), the picture is far less complete, and remains for future investigation. Nevertheless, we give some positive preliminary results about special cases of the minimum and maximum aggregate functions.

Various formal measures have been proposed for quantifying the contribution of a fact f to a query answer. Meliou et al. [24] adopted the quantity of *responsibility* that is inversely proportional to the minimal number of endogenous facts that should be removed to make f counterfactual (i.e., removing f transitions the answer from true to false). This measure adopts earlier notions of formal causality by Halpern and Pearl [16]. This measure, however, is fundamentally designed for non-numerical queries, and it is not at all clear whether it can incorporate the numerical contribution of a fact (e.g., recognizing that some facts contribute more than others due to high numerical attributes). Salimi et al. [31] proposed the *causal effect*: assuming endogenous facts are randomly removed independently and uniformly, what is the difference in the expected query answer between assuming the presence and the absence of f? Interestingly, as we show here, this value is the same as the *Banzhaf power index* that has also been studied in the context of wealth distribution in cooperative games [11], and is different from the Shapley value [30, Chapter 5]. While the justification to measuring fact contribution using one measure over the other is yet to be established, we believe that the suitability of the Shapley value is backed by the aforementioned theoretical justification as well as its massive adoption in a plethora of fields. In addition, the complexity of measuring the causal effect has been left open, and we conjecture that all of our complexity results are applicable to (and, in fact, simpler to prove in) the causal-effect framework.

The remainder of the paper is organized as follows. In the next section, we give preliminary concepts, definitions and notation. In Section 3, we present the Shapley value to measure the contribution of a fact to a query answer, along with illustrating examples. In Section 4, we study the complexity of calculating the Shapley value. Finally, we discuss past contribution measures in Section 5 and conclude in Section 6. For lack of space, missing proofs are given in the extended version of the paper [21].

2 Preliminaries

Databases. A (relational) *schema* $\mathbf{S}$ is a collection of *relation symbols* with each relation symbol R in $\mathbf{S}$ having an associated arity that we denote by $ar(R)$. We assume a countably infinite set Const of *constants* that are used as database values. If $\vec{c} = (c_1, \dots, c_k)$ is a tuple of constants and $i \in \{1, \dots, k\}$, then we use $\vec{c}[i]$ to refer to the constant c_i. A *relation* r is a set of tuples of constants, each having the same arity (length) that we denote by $ar(r)$. A *database* D (over the schema $\mathbf{S}$) associates with each relation symbol R a finite relation r, which we denote by R^D, such that $ar(R) = ar(R^D)$. We denote by $\mathrm{DB}(\mathbf{S})$ the set of all databases over the schema $\mathbf{S}$. Notationally, we identify a database D with its finite set of *facts* $R(c_1, \dots, c_k)$, stating that the relation R^D over the k-ary relation symbol R contains the tuple $(c_1, \dots, c_k) \in \mathsf{Const}^k$. In particular, two databases D and D' over $\mathbf{S}$ satisfy $D \subseteq D'$ if and only if $R^D \subseteq R^{D'}$ for all relation symbols R of $\mathbf{S}$.

Following prior work on explanations and responsibility of facts to query answers [23, 25], we view the database as consisting of two types of facts: *exogenous* facts and *endogenous* facts. Exogenous facts represent a context of information that is taken for granted and assumed not to claim any contribution or responsibility to the result of a query. Our concern is about *the role of the endogenous facts* in establishing the result of the query. In notation, we denote by D_x and D_n the subsets of D that consist of the exogenous and endogenous facts, respectively. Hence, in our notation we have that $D = D_\mathsf{x} \cup D_\mathsf{n}$.

AUTHOR (endo)

	name	*affil*
f_1^{a}	`Alice`	`UCLA`
f_2^{a}	`Bob`	`NYU`
f_3^{a}	`Cathy`	`UCSD`
f_4^{a}	`David`	`MIT`
f_5^{a}	`Ellen`	`UCSD`

INST (exo)

	name	*state*
f_1^{i}	`UCLA`	`CA`
f_2^{i}	`UCSD`	`CA`
f_3^{i}	`NYU`	`NY`
f_4^{i}	`MIT`	`MA`

PUB (exo)

	author	*pub*
f_1^{p}	`Alice`	`A`
f_2^{p}	`Alice`	`B`
f_3^{p}	`Bob`	`C`
f_4^{p}	`Cathy`	`C`
f_5^{p}	`Cathy`	`D`
f_6^{p}	`David`	`C`

CITATIONS (exo)

	paper	*cits*
f_1^{c}	`A`	`18`
f_2^{c}	`B`	`2`
f_2^{c}	`C`	`8`
f_3^{c}	`D`	`12`

Figure 1 The database of the running example.

▶ **Example 1.** Figure 1 depicts the database D of our running example from the domain of academic publications. The relation AUTHOR stores authors along with their affiliations, which are stored with their states in INST. The relation PUB associates authors with their publications, and CITATIONS stores the number of citations for each paper. For example, publication `C` has `8` citations and it is written jointly by `Bob` from `NYU` of `NY` state, `Cathy` from `UCSD` of `CA` state, and `David` from `MIT` of `MA` state. All AUTHOR facts are endogenous, and all remaining facts are exogenous. Hence, $D_{\mathsf{n}} = \{f_1^{\mathrm{a}}, f_2^{\mathrm{a}}, f_3^{\mathrm{a}}, f_4^{\mathrm{a}}, f_5^{\mathrm{a}}\}$ and D_{x} consists of all f_j^x for $x \in \{\mathrm{i}, \mathrm{p}, \mathrm{c}\}$ and relevant j. ◀

Relational and conjunctive queries. Let $\mathbf{S}$ be a schema. A *relational query* is a function that maps databases to relations. More formally, a relational query q of arity k is a function $q : \mathrm{DB}(\mathbf{S}) \to \mathsf{Const}^k$ that maps every database over $\mathbf{S}$ to a finite relation $q(D)$ of arity k. We denote the arity of q by $ar(q)$. Each tuple $\vec{c}$ in $q(D)$ is an *answer* to q on D. If the arity of q is zero, then we say that q is a *Boolean* query; in this case, $D \models q$ denotes that $q(D)$ consists of the empty tuple $()$, while $D \not\models q$ denotes that $q(D)$ is empty.

Our analysis will focus on the special case of *Conjunctive Queries* (CQs). A CQ over the schema $\mathbf{S}$ is a relational query definable by a first-order formula of the form $\exists y_1 \cdots \exists y_m \theta(\vec{x}, y_1, \dots, y_m)$, where θ is a conjunction of atomic formulas of the form $R(\vec{t})$ with variables among those in $\vec{x}, y_1, \dots, y_m$. In the remainder of the paper, a CQ q will be written shortly as a logic rule, that is, an expression of the form

$$q(\vec{x}) \text{ :- } R_1(\vec{t}_1), \dots, R_n(\vec{t}_n)$$

where each R_i is a relation symbol of $\mathbf{S}$, each $\vec{t}_i$ is a tuple of variables and constants with the same arity as R_i, and $\vec{x}$ is a tuple of k variables from $\vec{t}_1, \dots, \vec{t}_n$. We call $q(\vec{x})$ the *head* of q, and $R_1(\vec{t}_1), \dots, R_n(\vec{t}_n)$ the *body* of q. Each $R_i(\vec{t}_i)$ is an *atom* of q. The variables occurring in the head are called the *head variables*, and we make the standard safety assumption that every head variable occurs at least once in the body. The variables occurring in the body but not in the head are existentially quantified, and are called the *existential variables*. The answers to q on a database D are the tuples $\vec{c}$ that are obtained by projecting to $\vec{x}$ all homomorphisms from q to D, and replacing each variable with the constant it is mapped to. A homomorphism from q to D is a mapping of the variables in q to the constants of D, such that every atom in q is mapped to a fact in D.

A *self-join* in a CQ q is a pair of distinct atoms over the same relation symbol. For example, in the query $q() \text{ :- } R(x, y), S(x), R(y, z)$, the first and third atoms constitute a self-join. We say that q is *self-join-free* if it has no self-joins, or in other words, every relation symbol occurs at most once in the body.

Let q be a CQ. For a variable y of q, let A_y be the set of atoms $R_i(\vec{t_i})$ of q that contain y (that is, y occurs in $\vec{t_i}$). We say that q is *hierarchical* if for all existential variables y and y' it holds that $A_y \subseteq A_{y'}$, or $A_{y'} \subseteq A_y$, or $A_y \cap A_{y'} = \emptyset$ [8]. For example, every CQ with at most two atoms is hierarchical. The smallest non-hierarchical CQ is the following.

$$\mathsf{q}_{\mathsf{RST}}() \text{ :- } R(x), S(x,y), T(y) \tag{1}$$

On the other hand, the query $q(x)$:- $R(x), S(x,y), T(y)$, which has a single existential variable, is hierarchical.

Let q be a Boolean query and D a database, both over the same schema, and let $f \in D_{\mathsf{n}}$ be an endogenous fact. We say that f is a *counterfactual cause* (*for q w.r.t. D*) [23,24] if the removal of f causes q to become false; that is, $D \models q$ and $D \setminus \{f\} \not\models q$.

▶ **Example 2.** We will use the following queries in our examples.

$$\begin{aligned}
q_1() &\text{ :- } \textsc{Author}(x,y), \textsc{Pub}(x,z)\\
q_2() &\text{ :- } \textsc{Author}(x,y), \textsc{Pub}(x,z), \textsc{Citations}(z,w)\\
q_3(z,w) &\text{ :- } \textsc{Author}(x,y), \textsc{Pub}(x,z), \textsc{Citations}(z,w)\\
q_4(z,w) &\text{ :- } \textsc{Author}(x,y), \textsc{Pub}(x,z), \textsc{Citations}(z,w), \textsc{Inst}(y, \texttt{CA})
\end{aligned}$$

Note that q_1 and q_2 are Boolean, whereas q_3 and q_4 are not. Also note that q_1 and q_3 are hierarchical, and q_2 and q_4 are not. Considering the database D of Figure 1, none of the Author facts is a counterfactual cause for q_1, since the query remains true even if the fact is removed. The same applies to q_2. However, the fact f_1^{a} is a counterfactual cause for the Boolean CQ $q_1'()$:- $\textsc{Author}(x, \texttt{UCLA}), \textsc{Pub}(x,z)$, asking whether there is a publication with an author from UCLA, since D satisfies q_1' but the removal of Alice causes q_1' to be violated by D, as no other author from UCLA exists. ◀

Numerical and aggregate-relational queries. A *numerical query* α is a function that maps databases to numbers. More formally, a numerical query α is a function $\alpha : \mathrm{DB}(\mathbf{S}) \to \mathbb{R}$ that maps every database D over $\mathbf{S}$ to a real number $\alpha(D)$.

A special form of a numerical query α is what we refer to as an *aggregate-relational query*: a k-ary relational query q followed by an aggregate function $\gamma : \mathcal{P}(\mathsf{Const}^k) \to \mathbb{R}$ (where $\mathcal{P}(\mathsf{Const}^k)$ is the power set of Const^k that consists of all subsets of Const^k) that maps the resulting relation $q(D)$ into a single number $\gamma(q(D))$. We denote this aggregate-relational query as $\gamma[q]$; hence, $\gamma[q](D) \stackrel{\text{def}}{=} \gamma(q(D))$.

Special cases of aggregate-relational queries include the functions of the form $\gamma = F\langle\varphi\rangle$ that transform every tuple $\vec{c}$ into a number $\varphi(\vec{c})$ via a *feature function* $\varphi : \mathsf{Const}^k \to \mathbb{R}$, and then contract the resulting bag of numbers into a single number. Formally, we define $F\langle\varphi\rangle[q](D) \stackrel{\text{def}}{=} F(\{\!\{\varphi(\vec{c}) \mid \vec{c} \in q(D)\}\!\})$ where $\{\!\{\cdot\}\!\}$ is used for bag notation. For example, if we assume that the ith attribute of $q(D)$ takes a numerical value, then φ can simply copy this number (i.e., $\varphi(\vec{c}) = \vec{c}[i]$); we denote this φ by $[i]$. As another example, φ can be the product of two attributes: $\varphi = [i] \cdot [j]$. We later refer to the following aggregate-relational queries.

$$\mathsf{sum}\langle\varphi\rangle[q](D) \stackrel{\text{def}}{=} \sum_{\vec{c} \in q(D)} \varphi(\vec{c})$$

$$\mathsf{max}\langle\varphi\rangle[q](D) \stackrel{\text{def}}{=} \begin{cases} \max\{\varphi(\vec{c}) \mid \vec{c} \in q(D)\} & \text{if } q(D) \neq \emptyset;\\ 0 & \text{if } q(D) = \emptyset. \end{cases}$$

Other popular examples include the minimum (defined analogously to maximum), average and median over the feature values. A special case of $\mathsf{sum}\langle\varphi\rangle[q]$ is $\mathsf{count}[q]$ that counts the number of answers for q. That is, $\mathsf{count}[q]$ is $\mathsf{sum}\langle\mathbf{1}\rangle[q]$, where "$\mathbf{1}$" is the feature function that maps every k-tuple to the number 1. A special case of $\mathsf{count}[q]$ is when q is Boolean; in this case, we may abuse the notation and identify $\mathsf{count}[q]$ with q itself. Put differently, we view q as the numerical query α defined by $\alpha(D) = 1$ if $D \models q$ and $\alpha(D) = 0$ if $D \not\models q$.

▶ **Example 3.** Following are examples of aggregate-relational queries over the relational queries of Example 2.

- $\alpha_1 \overset{\text{def}}{=} \mathsf{sum}\langle[2]\rangle[q_3]$ calculates the total number of citations of all published papers.
- $\alpha_2 \overset{\text{def}}{=} \mathsf{count}[q_3]$ counts the papers in Citations with an author in the database.
- $\alpha_3 \overset{\text{def}}{=} \mathsf{sum}\langle[2]\rangle[q_4]$ calculates the total number of citations of papers by Californians.
- $\alpha_4 \overset{\text{def}}{=} \mathsf{max}\langle[2]\rangle[q_3]$ calculates the number of citations for the most cited paper.

For D of Figure 1 we have $\alpha_1(D) = 40$, $\alpha_2(D) = 4$, $\alpha_3(D) = 40$ and $\alpha_4(D) = 18$. ◀

In terms of presentation, when we mention general functions γ and φ, we make the implicit assumption that they are computable in polynomial time with respect to the representation of their input. Also, observe that our modeling of an aggregate-relational query does not allow for *grouping*, since a database is mapped to a single number. This is done for simplicity of presentation, and all concepts and results of this paper generalize to grouping as in traditional modeling (e.g., [6]). This is explained in the next section.

Shapley value. Let A be a finite set of *players*. A *cooperative game* is a function $v : \mathcal{P}(A) \to \mathbb{R}$, such that $v(\emptyset) = 0$. The value $v(S)$ represents a value, such as wealth, jointly obtained by S when the players of S cooperate. The *Shapley value* [33] measures the share of each individual player $a \in A$ in the gain of A for the cooperative game v. Intuitively, the gain of a is as follows. Suppose that we form a team by taking the players one by one, randomly and uniformly without replacement; while doing so, we record the change of v due to the addition of a as the random contribution of a. Then the Shapley value of a is the expectation of the random contribution.

$$\text{Shapley}(A, v, a) \overset{\text{def}}{=} \frac{1}{|A|!} \sum_{\sigma \in \Pi_A} \big(v(\sigma_a \cup \{a\}) - v(\sigma_a)\big) \tag{2}$$

where Π_A is the set of all possible permutations over the players in A, and for each permutation σ we denote by σ_a the set of players that appear before a in the permutation.

An alternative formula for the Shapley value is the following.

$$\text{Shapley}(A, v, a) \overset{\text{def}}{=} \sum_{B \subseteq A \setminus \{a\}} \frac{|B|! \cdot (|A| - |B| - 1)!}{|A|!} \Big(v(B \cup \{a\}) - v(B)\Big) \tag{3}$$

Note that $|B|! \cdot (|A| - |B| - 1)!$ is the number of permutations over A such that all players in B come first, then a, and then all remaining players. For further reading, we refer the reader to the book by Roth [30].

3 Shapley Value of Database Facts

Let α be a numerical query over a schema $\mathbf{S}$, and let D be a database over $\mathbf{S}$. We wish to quantify the contribution of every endogenous fact to the result $\alpha(D)$. For that, we view α as a cooperative game over D_{n}, where the value of every subset E of D_{n} is $\alpha(E \cup D_{\mathsf{x}})$.

▶ **Definition 4** (Shapley Value of Facts). *Let* $\mathbf{S}$ *be a schema,* α *a numerical query,* D *a database, and* f *an endogenous fact of* D*. The* Shapley value *of* f *for* α*, denoted* $\mathrm{Shapley}(D, \alpha, f)$*, is the value* $\mathrm{Shapley}(A, v, a)$ *as given in* (2)*, where:*

- $A = D_{\mathsf{n}}$*;*
- $v(E) = \alpha(E \cup D_{\mathsf{x}}) - \alpha(D_{\mathsf{x}})$ *for all* $E \subseteq A$*;*
- $a = f$*.*

That is, $\mathrm{Shapley}(D, \alpha, f)$ *is the Shapley value of* f *in the cooperative game that has the endogenous facts as the set of players and values each team by the quantity it adds to* α*.*

As a special case, if q is a Boolean query, then $\mathrm{Shapley}(D, q, f)$ is the same as the value $\mathrm{Shapley}(D, \mathsf{count}[q], f)$. In this case, the corresponding cooperative game takes the values 0 and 1, and the Shapley value then coincides with the *Shapley-Shubik index* [32]. Some fundamental properties of the Shapley value [33] are reflected here as follows:

- $\mathrm{Shapley}(D, a \cdot \alpha + b \cdot \beta, f) = a \cdot \mathrm{Shapley}(D, \alpha, f) + b \cdot \mathrm{Shapley}(D, \beta, f)$.
- $\alpha(D) = \alpha(D_{\mathsf{x}}) + \sum_{f \in D_{\mathsf{n}}} \mathrm{Shapley}(D, \alpha, f)$.

▶ Remark 5. Note that $\mathrm{Shapley}(D, \alpha, f)$ is defined for a general numerical query α. The definition is immediately extendible to queries with *grouping* (producing tuples of database constants and numbers [6]), where we would measure the responsibility of f for an answer tuple $\vec{a}$ and write something like $\mathrm{Shapley}(D, \alpha, \vec{a}, f)$. In that case, we treat every group as a separate numerical query. We believe that focusing on numerical queries (without grouping) allows us to keep the presentation considerably simpler while, at the same time, retaining the fundamental challenges. ◀

In the remainder of this section, we illustrate the Shapley value on our running example.

▶ **Example 6.** We begin with a Boolean CQ, and specifically q_1 from Example 2. Recall that the endogenous facts correspond to the authors. As Ellen has no publications, her addition to any $D_{\mathsf{x}} \cup E$ where $E \subseteq D_{\mathsf{n}}$ does not change the satisfaction of q_1. Hence, its Shapley value is zero: $\mathrm{Shapley}(D, q_1, f_5^{\mathrm{a}}) = 0$. The fact f_1^{a} changes the query result if it is either the first fact in the permutation, or it is the second fact after f_5^{a}. There are 4! permutations that satisfy the first condition, and 3! permutations that satisfy the second. The contribution of f_1^{a} to the query result is one in each of these permutations, and zero otherwise. Therefore, we have $\mathrm{Shapley}(D, q_1, f_1^{\mathrm{a}}) = \frac{4!+3!}{120} = \frac{1}{4}$. The same argument applies to f_2^{a}, f_3^{a} and f_4^{a}, and so, $\mathrm{Shapley}(D, q_1, f_2^{\mathrm{a}}) = \mathrm{Shapley}(D, q_1, f_3^{\mathrm{a}}) = \mathrm{Shapley}(D, q_1, f_4^{\mathrm{a}}) = \frac{1}{4}$. We get the same numbers for q_2, since every paper is mentioned in the CITATIONS relation. Note that the value of the query q_1 on the database is 1, and it holds that $\sum_{i=1}^{5} \mathrm{Shapley}(D, q_1, f_i^{\mathrm{a}}) = 4 \cdot \frac{1}{4} + 0 = 1$; hence, the second fundamental property of the Shapley value mentioned above is satisfied.

While Alice, Bob, Cathy and David have the same Shapley value for q_1, things change if we consider the relation PUB endogenous as well: the Shapley value of Alice and Cathy will be higher than Bob's and David's values, since they have more publications. Specifically, the fact f_1^{a}, for example, will change the query result if and only if at least one of f_1^{p} or f_2^{p} appears earlier in the permutation, and no pair among $\{f_2^{\mathrm{a}}, f_3^{\mathrm{p}}\}$, $\{f_3^{\mathrm{a}}, f_3^{\mathrm{p}}\}$, $\{f_3^{\mathrm{a}}, f_4^{\mathrm{p}}\}$, and $\{f_4^{\mathrm{a}}, f_3^{\mathrm{p}}\}$ appears earlier than f_1^{a}. By rigorous counting, we can show that there are: 2 such sets of size one, 17 such sets of size two, 56 such sets of size three, 90 such sets of size four, 73 such sets of size five, 28 such sets of size six, and 4 such sets of size seven. Therefore, the Shapley value of f_1^{a} is:

$$\mathrm{Shapley}(D, q_1, f_1^{\mathrm{a}}) = 2 \cdot \frac{(11-2)!1!}{11!} + 17 \cdot \frac{(11-3)!2!}{11!} + 56 \cdot \frac{(11-4)!3!}{11!} + 90 \cdot \frac{(11-5)!4!}{11!} + 73 \cdot \frac{(11-6)!5!}{11!} + 28 \cdot \frac{(11-7)!6!}{11!} + 4 \cdot \frac{(11-8)!7!}{11!} = \frac{442}{2520}$$

We can similarly compute the Shapley value for the rest of the authors, concluding that $\text{Shapley}(D, q_1, f_2^a) = \text{Shapley}(D, q_1, f_4^a) = \frac{241}{2520}$ and $\text{Shapley}(D, q_1, f_3^a) = \frac{442}{2520}$. Hence, the Shapley value is the same for Alice and Cathy, who have two publications each, and lower for Bob and David, that have only one publication. ◀

The following example, taken from Salimi et al. [31], illustrates the Shapley value on (Boolean) graph reachability.

▶ **Example 7.** Consider the following database G defined via the relation symbol Edge/2.

$$\begin{tikzcd}
a \arrow[rrr, "e_1"] \arrow[rrd, "e_2"'] \arrow[dd, "e_4"'] & & & b \arrow[dd, "e_6"] \\
& & c \arrow[ru, "e_3"'] & \\
d \arrow[rrr, "e_5"'] & & & e
\end{tikzcd}$$

Here, we assume that all edges e_i are endogenous facts. Let p_{ab} be the Boolean query (definable in, e.g., Datalog) that determines whether there is a path from a to b. Let us calculate $\text{Shapley}(G, p_{ab}, e_i)$ for different edges e_i. Intuitively, we expect e_1 to have the highest value since it provides a direct path from a to b, while e_2 contributes to a path only in the presence of e_3, and e_4 enables a path only in the presence of both e_5 and e_6. We show that, indeed, it holds that $\text{Shapley}(G, p_{ab}, e_1) > \text{Shapley}(G, p_{ab}, e_2) > \text{Shapley}(G, p_{ab}, e_4)$.

To illustrate the calculation, observe that there are 2^5 subsets of G that do not contain e_1, and among them, the subsets that satisfy p_{ab} are the supersets of $\{e_2, e_3\}$ and $\{e_4, e_5, e_6\}$. Hence, we have that $\text{Shapley}(G, p_{ab}, e_1) = \frac{35}{60}$ (the detailed computation is in the extended version of the paper [21]). A similar reasoning shows that $\text{Shapley}(G, p_{ab}, e_2) = \text{Shapley}(G, p_{ab}, e_3) = \frac{8}{60}$, and that $\text{Shapley}(G, p_{ab}, e_i) = \frac{3}{60}$ for $i = 4, 5, 6$. ◀

Lastly, we consider aggregate functions over conjunctive queries.

▶ **Example 8.** We consider the queries α_1, α_2, and α_4 from Example 3. Ellen has no publications; hence, $\text{Shapley}(D, \alpha_j, f_5^a) = 0$ for $j \in \{1, 2, 4\}$. The contribution of f_1^a is the same in every permutation (20 for α_1 and 2 for α_2) since Alice is the single author of two published papers that have a total of 20 citations. Hence, $\text{Shapley}(D, \alpha_1, f_1^a) = 20$ and $\text{Shapley}(D, \alpha_2, f_1^a) = 2$. The total number of citations of Cathy's papers is also 20; however, Bob and David are her coauthors on paper C. Hence, if the fact f_3^a appears before f_2^a and f_4^a in a permutation, its contribution to the query result is 20 for α_1 and 2 for α_2, while if f_3^a appears after at least one of f_2^a or f_4^a in a permutation, its contribution is 12 for α_1 and 1 for α_2. Clearly, f_2^a appears before both f_3^a and f_4^a in one-third of the permutations. Thus, we have that $\text{Shapley}(D, \alpha_1, f_3^a) = \frac{1}{3} \cdot 20 + \frac{2}{3} \cdot 12 = \frac{44}{3}$ and $\text{Shapley}(D, \alpha_2, f_3^a) = \frac{1}{3} \cdot 2 + \frac{2}{3} \cdot 1 = \frac{4}{3}$. Using similar computations we obtain that $\text{Shapley}(D, \alpha_1, f_2^a) = \text{Shapley}(D, \alpha_1, f_4^a) = \frac{8}{3}$ and $\text{Shapley}(D, \alpha_2, f_2^a) = \text{Shapley}(D, \alpha_2, f_4^a) = \frac{1}{3}$.

Hence, the Shapley value of Alice, who is the single author of two papers with a total of 20 citations, is higher than the Shapley value of Cathy who also has two papers with a total of 20 citations, but shares one paper with other authors. Bob and David have the same Shapley value, since they share a single paper, and this value is the lowest among the four, as they have the lowest number of papers and citations.

Finally, consider α_4. The contribution of f_1^a in this case depends on the maximum value before adding f_1^a in the permutation (which can be 0, 8 or 12). For example, if f_1^a is the first fact in the permutation, its contribution is 18 since $\alpha_4(\emptyset) = 0$. If f_1^a appears after f_3^a, then its contribution is 6, since $\alpha_4(S) = 12$ whenever $f_3^a \in S$. We have that $\text{Shapley}(D, \alpha_4, f_1^a) = 10$, $\text{Shapley}(D, \alpha_4, f_2^a) = \text{Shapley}(D, \alpha_4, f_4^a) = 2$ and $\text{Shapley}(D, \alpha_4, f_3^a) = 4$ (we omit the computations here). We see that the Shapley value of f_1^a is much higher than the rest, since

Alice significantly increases the maximum value when added to any prefix. If the number of citations of paper `C` increases to 16, then $\mathrm{Shapley}(D, \alpha_4, f_1^{\mathrm{a}}) = 6$, hence lower. This is because the next highest value is closer; hence, the contribution of f_1^{a} diminishes. ◀

4 Complexity Results

In this section, we give complexity results on the computation of the Shapley value of facts. We begin with exact evaluation for Boolean CQs (Section 4.1), then move on to exact evaluation on aggregate-relational queries (Section 4.2), and finally discuss approximate evaluation (Section 4.3). In the first two parts we restrict the discussion to CQs without self-joins, and leave the problems open in the presence of self-joins. However, the approximate treatment in the third part covers the general class of CQs (and beyond).

4.1 Boolean Conjunctive Queries

We investigate the problem of computing the (exact) Shapley value w.r.t. a Boolean CQ without self-joins. Our main result in this section is a full classification of (i.e., a dichotomy in) the data complexity of the problem. As we show, the classification criterion is the same as that of query evaluation over tuple-independent probabilistic databases [9]: hierarchical CQs without self-joins are tractable, and non-hierarchical ones are intractable.

▶ **Theorem 9.** *Let q be a Boolean CQ without self-joins. If q is hierarchical, then computing* $\mathrm{Shapley}(D, q, f)$ *can be done in polynomial time, given D and f. Otherwise, the problem is* $\mathrm{FP}^{\#\mathrm{P}}$*-complete.*

Recall that $\mathrm{FP}^{\#\mathrm{P}}$ is the class of functions computable in polynomial time with an oracle to a problem in #P (e.g., counting the number of satisfying assignments of a propositional formula). This complexity class is considered intractable, and is known to be above the polynomial hierarchy (Toda's theorem [35]).

▶ **Example 10.** Consider the query q_1 from Example 2. This query is hierarchical; hence, by Theorem 9, $\mathrm{Shapley}(D, q_1, f)$ can be calculated in polynomial time, given D and f. On the other hand, the query q_2 is not hierarchical. Thus, Theorem 9 asserts that computing $\mathrm{Shapley}(D, q_2, f)$ is $\mathrm{FP}^{\#\mathrm{P}}$-complete. ◀

In the rest of this subsection, we discuss the proof of Theorem 9. While the tractability condition is the same as that of Dalvi and Suciu [9], it is not clear whether and/or how we can use their dichotomy to prove ours, in each of the two directions (tractability and hardness). The difference is mainly in that they deal with a random subset of probabilistically independent (endogenous) facts, whereas we reason about random *permutations* over the facts. We stary by discussing the algorithm for computing the Shapley value in the hierarchical case, and then we discuss the proof of hardness for the non-hierarchical case.

Tractability side. Let D be a database, let f be an endogenous fact, and let q be a Boolean query. The computation of $\mathrm{Shapley}(D, q, f)$ easily reduces to the problem of counting the k-sets (i.e., sets of size k) of endogenous facts that, along with the exogenous facts, satisfy q. More formally, the reduction is to the problem of computing $|\mathrm{Sat}(D, q, k)|$ where $\mathrm{Sat}(D, q, k)$ is the set of all subsets E of D_{n} such that $|E| = k$ and $(D_{\mathsf{x}} \cup E) \models q$. The reduction is as follows, where we denote $m = |D_{\mathsf{n}}|$ and slightly abuse the notation by viewing q as a 0/1-numerical query, where $q(D') = 1$ if and only if $D' \models q$.

$$\begin{aligned}\mathrm{Shapley}(D,q,f) &= \sum_{E\subseteq(D_{\mathsf{n}}\setminus\{f\})} \frac{|E|!(m-|E|-1)!}{m!}\Big(q(D_{\mathsf{x}}\cup E\cup\{f\})-q(D_{\mathsf{x}}\cup E)\Big) \qquad (4)\\ &= \sum_{E\subseteq(D_{\mathsf{n}}\setminus\{f\})} \frac{|E|!(m-|E|-1)!}{m!}\Big(q(D_{\mathsf{x}}\cup E\cup\{f\})\Big) - \sum_{E\subseteq(D_{\mathsf{n}}\setminus\{f\})} \frac{|E|!(m-|E|-1)!}{m!}\Big(q(D_{\mathsf{x}}\cup E)\Big)\\ &= \left(\sum_{k=0}^{m-1}\frac{k!(m-k-1)!}{m!}\times|\mathrm{Sat}(D',q,k)|\right)-\left(\sum_{k=0}^{m-1}\frac{k!(m-k-1)!}{m!}\times|\mathrm{Sat}(D\setminus\{f\},q,k)|\right)\end{aligned}$$

In the last expression, D' is the same as D, except that f is viewed as *exogenous* instead of *endogenous*. Hence, to prove the positive side of Theorem 9, it suffices to show the following.

▶ **Theorem 11.** *Let q be a hierarchical Boolean CQ without self-joins. There is a polynomial-time algorithm for computing the number $|\mathrm{Sat}(D,q,k)|$ of subsets E of D_{n} such that $|E|=k$ and $(D_{\mathsf{x}}\cup E)\models q$, given D and k.*

To prove Theorem 11, we show a polynomial-time algorithm for computing $|\mathrm{Sat}(D,q,k)|$ for q as in the theorem. The pseudocode is depicted in Figure 2.

We assume in the algorithm that D_{n} contains only facts that are homomorphic images of atoms of q (i.e., facts f such that there is a mapping from an atom of q to f). In the terminology of Conitzer and Sandholm [7], the function defined by q *concerns* only the subset C of D_{n} consisting of these facts (i.e., the satisfaction of q by any subset of D does not change if we intersect with C), and so, the Shapley value of every fact in $D_{\mathsf{n}}\setminus C$ is zero and the Shapley value of any other fact is unchanged when ignoring $D_{\mathsf{n}}\setminus C$ [7, Lemma 4]. Moreover, these facts can be found in polynomial time.

As expected for a hierarchical query, our algorithm is a recursive procedure that acts differently in three different cases: *(a)* q has no variables (only constants), *(b)* there is a *root* variable x, that is, x occurs in all atoms of q, or *(c)* q consists of two (or more) subqueries that do not share any variables. Since q is hierarchical, at least one of these cases always applies [10].

In the first case (lines 1-7), every atom a of q can be viewed as a fact. Clearly, if one of the facts in q is not present in D, then there is no subset E of D_{n} of any size such that $(D_{\mathsf{x}}\cup E)\models q$, and the algorithm will return 0. Otherwise, suppose that A is the set of endogenous facts of q (and the remaining atoms of q, if any, are exogenous). Due to our assumption that every fact of D_{n} is a homomorphic image of an atom of q, the single choice of a subset of facts that makes the query true is A; therefore, the algorithm returns 1 if $k=|A|$ and 0 otherwise.

Next, we consider the case where q has a root variable x (lines 9-21). We denote by V_x the set $\{v_1,\dots,v_n\}$ of values that D has in attributes that correspond to an occurrence of x. For example, if q contains the atom $R(x,y,x)$ and D contains a fact $R(\mathtt{a},\mathtt{b},\mathtt{a})$, then $\mathtt{a}$ is one of the values in V_x. We also denote by $q_{[x\to v_i]}$ the query that is obtained from q by substituting v_i for x, and by D^{v_i} the subset of D that consists of facts with the value v_i in every attribute where x occurs in q.

We solve the problem for this case using a simple dynamic program. We denote by P_i^ℓ the number of subsets of size ℓ of $\bigcup_{r=1}^{i} D_{\mathsf{n}}^{v_r}$ that satisfy the query (together with the exogenous facts in $\bigcup_{r=1}^{i} D_{\mathsf{x}}^{v_r}$). Our goal is to find P_n^k, which is the number of subsets E of size k of $\bigcup_{r=1}^{n} D_{\mathsf{n}}^{v_r}$. Note that this union is precisely D_{n}, due to our assumption that D_{n} contains only facts that can be obtained from atoms of q via an assignment to the variables. First, we compute, for each value v_i, and for each $j\in\{0,\dots,k\}$, the number $f_{i,j}$ of subsets E of size j of $D_{\mathsf{n}}^{v_i}$ such that $(D_{\mathsf{x}}^{v_i}\cup E)\models q$, using a recursive call. In the recursive call, we replace q with $q_{[x\to v_i]}$, as D^{v_i} contains only facts that use the value v_i for the variable x; hence, we can

Algorithm 1 $\mathsf{CntSat}(D, q, k)$.

```
1:  if $\mathsf{Vars}(q) = \emptyset$ then
2:      if $\exists a \in \mathsf{Atoms}(q)$ s.t. $a \notin D$ then
3:          return 0
4:      $A = \mathsf{Atoms}(q) \cap D_\mathsf{n}$
5:      if $|A| = k$ then
6:          return 1
7:      return 0
8:  result $\leftarrow 0$
9:  if $q$ has a root variable that occurs in all atoms then
10:     $x \leftarrow$ a root variable of $q$
11:     $V_x \leftarrow$ the set $\{v_1, \dots, v_n\}$ of values for $x$
12:     for all $i \in \{1, \dots, |V_x|\}$ do
13:         for all $j \in \{0, \dots, k\}$ do
14:             $f_{i,j} =\leftarrow \mathsf{CntSat}(D^{v_i}, q_{[x \to v_i]}, j)$
15:     $P_1^\ell = f_{1,\ell}$ for all $\ell \in \{0, \dots, k\}$
16:     for all $i \in \{2, \dots, |V_x|\}$ do
17:         for all $\ell \in \{0, \dots, k\}$ do
18:             $P_i^\ell \leftarrow 0$
19:             for all $j \in \{0, \dots, \ell\}$ do
20:                 $P_i^\ell \leftarrow P_i^\ell + P_{i-1}^{\ell-j} \cdot f_{i,j} + \big[\binom{\sum_{r=1}^{i-1} |D_\mathsf{n}^{v_r}|}{\ell-j} - P_{i-1}^{\ell-j}\big] \cdot f_{i,j} + P_{i-1}^{\ell-j} \cdot \big[\binom{|D_\mathsf{n}^{v_i}|}{j} - f_{i,j}\big]$
21:     result $\leftarrow P_n^k$
22: else
23:     let $q = q_1 \wedge q_2$ where $\mathsf{Vars}(q_1) \cap \mathsf{Vars}(q_2) = \emptyset$
24:     let $D^1$ and $D^2$ be the restrictions of $D$ to the relations of $q_1$ and $q_2$, respectively
25:     for all $k_1, k_2$ s.t. $k_1 + k_2 = k$ do
26:         result $\leftarrow$ result $+ \mathsf{CntSat}(D^1, q_1, k_1) \cdot \mathsf{CntSat}(D^2, q_2, k_2)$
27: return result
```

Figure 2 An algorithm for computing $|\mathrm{Sat}(D, q, k)|$ where q is a hierarchical Boolean CQ without self-joins.

reduce the number of variables in q by substituting x with v_i. Then, for each $\ell \in \{0, \dots, k\}$ it clearly holds that $P_1^\ell = f_{1,\ell}$. For each $i \in \{2, \dots, |V_x|\}$ and $\ell \in \{0, \cdots, k\}$, we compute P_i^ℓ in the following way. Each subset E of size ℓ of $\bigcup_{r=1}^{i} D_\mathsf{n}^{v_r}$ contains a set E_1 of size j of facts from $D_\mathsf{n}^{v_i}$ (for some $j \in \{0, \dots, \ell\}$) and a set E_2 of size $\ell - j$ of facts from $\bigcup_{r=1}^{i-1} D_\mathsf{n}^{v_r}$. If the subset E satisfies the query, then precisely one of the following holds:

1. $(D_\mathsf{x}^{v_i} \cup E_1) \models q$ and $(\bigcup_{r=1}^{i-1} D_\mathsf{x}^{v_r} \cup E_2) \models q$,
2. $(D_\mathsf{x}^{v_i} \cup E_1) \models q$, but $(\bigcup_{r=1}^{i-1} D_\mathsf{x}^{v_r} \cup E_2) \not\models q$,
3. $(D_\mathsf{x}^{v_i} \cup E_1) \not\models q$, but $(\bigcup_{r=1}^{i-1} D_\mathsf{x}^{v_r} \cup E_2) \models q$.

Hence, we add to P_i^ℓ the value $P_{i-1}^{\ell-j} \cdot f_{i,j}$ that corresponds to Case (1), the value

$$\left(\binom{\bigcup_{r=1}^{i-1} |D_\mathsf{n}^{v_r}|}{\ell - j} - P_{i-1}^{\ell-j}\right) \cdot f_{i,j}$$

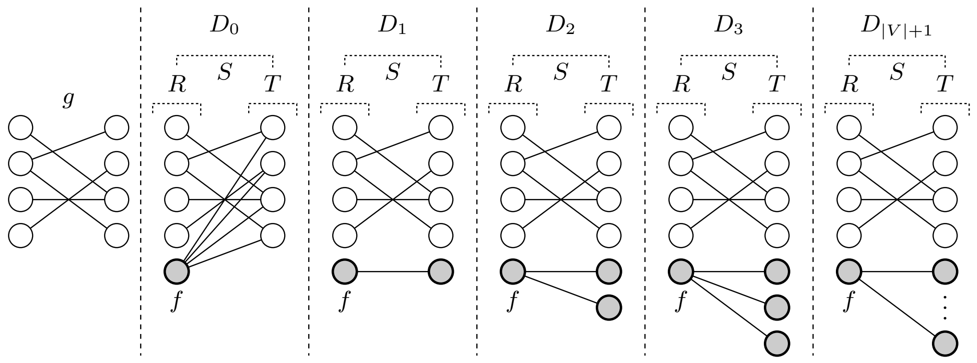

Figure 3 Constructions in the reduction of the proof of Lemma 12. Relations $R/1$ and $T/1$ consist of endogenous facts and $S/2$ consists of exogenous facts.

that corresponds to Case (2), and the value

$$P_{i-1}^{\ell-j} \cdot \left(\binom{|D_{\mathsf{n}}^{v_i}|}{j} - f_{i,j} \right)$$

that corresponds to Case (3). Note that we have all the values $P_{i-1}^{\ell-j}$ from the previous iteration of the for loop of line 16.

Finally, we consider the case where q has two nonempty subqueries q_1 and q_2 with disjoint sets of variables (lines 23-26). For $j \in \{1,2\}$, we denote by D^j the set of facts from D that appear in the relations of q_j. (Recall that q has no self-joins; hence, every relation can appear in either q_1 or q_2, but not in both.) Every subset E of D that satisfies q must contain a subset E_1 of D^1 that satisfies q_1 and a subset E_2 of D^2 satisfying q_2. Therefore, to compute $|\mathrm{Sat}(D,q,k)|$, we consider every pair (k_1,k_2) of natural numbers such that $k_1+k_2=k$, compute $|\mathrm{Sat}(D^1,q_1,k_1)|$ and $|\mathrm{Sat}(D^2,q_2,k_2)|$ via a recursive call, and add the product of the two to the result.

Hardness side. We now sketch the proof of the negative side of Theorem 9. (The complete proof is in [21].) Membership in $\mathrm{FP}^{\#\mathrm{P}}$ is straightforward since, as aforementioned in Equation (4), the Shapley value can be computed in polynomial time given an oracle to the problem of counting the number of subsets $E \subseteq D_{\mathsf{n}}$ of size k such that $(D_{\mathsf{x}} \cup E) \models q$, and this problem is in #P. Similarly to Dalvi and Suciu [9], our proof of hardness consists of two steps. First, we prove the $\mathrm{FP}^{\#\mathrm{P}}$-hardness of computing $\mathrm{Shapley}(D, \mathsf{q_{RST}}, f)$, where $\mathsf{q_{RST}}$ is given in (1). Second, we reduce the computation of $\mathrm{Shapley}(D, \mathsf{q_{RST}}, f)$ to the problem of computing $\mathrm{Shapley}(D,q,f)$ for any non-hierarchical CQ q without self-joins. The second step is the same as that of Dalvi and Suciu [9], so we do not discuss it here. Hence, in what follows, we focus on the first step – hardness of computing $\mathrm{Shapley}(D, \mathsf{q_{RST}}, f)$, as stated next by Lemma 12. The proof, which we discuss after the lemma, is considerably more involved than the corresponding proof of Dalvi and Suciu [9] that computing the probability of $\mathsf{q_{RST}}$ in a tuple-independent probabilistic database (TID) is $\mathrm{FP}^{\#\mathrm{P}}$-hard.

▶ **Lemma 12.** *Computing* $\mathrm{Shapley}(D, \mathsf{q_{RST}}, f)$ *is* $\mathrm{FP}^{\#\mathrm{P}}$*-complete.*

The proof of Lemma 12 is by a (Turing) reduction from the problem of computing the number $|\mathsf{IS}(g)|$ of independent sets of a given bipartite graph g, which is the same (via immediate reductions) as the problem of computing the number of satisfying assignments of a bipartite monotone 2-DNF formula, which we denote by #biSAT. Dalvi and Suciu [9] also proved the hardness of $\mathsf{q_{RST}}$ (for the problem of query evaluation over TIDs) by reduction from $\#\mathsf{biSAT}$. Their reduction is a simple construction of a single input database, followed by a multiplication of the query probability by a number. It is not at all clear to us how such an approach can work in our case and, indeed, our proof is more involved. Our reduction takes the general approach that Dalvi and Suciu [10] used (in a different work) for proving that the CQ $q() \text{ :- } R(x,y), R(y,z)$ is hard over TIDs: solve several instances of the problem for the construction of a full-rank set of linear equations. The problem itself, however, is quite different from ours. This general technique has also been used by Aziz et al. [2] for proving the hardness of computing the Shapley value for a *matching game* on unweighted graphs, which is again quite different from our problem.

In more detail, the idea is as follows. Given an input bipartite graph $g = (V, E)$ for which we wish to compute $|\mathsf{IS}(g)|$, we construct $n+1$ different input instances (D_j, f), for $j = 1, \dots, n+1$, of the problem of computing $\text{Shapley}(D_j, \mathsf{q_{RST}}, f)$, where $n = |V|$. Each instance provides us with an equation over the numbers $|\mathsf{IS}(g,k)|$ of independent sets of size k in g for $k = 0, \dots, n$. We then show that the set of equations constitutes a non-singular matrix that, in turn, allows us to extract the $|\mathsf{IS}(g,k)|$ in polynomial time (e.g., via Gaussian elimination). This is enough, since $|\mathsf{IS}(g)| = \sum_{k=0}^{n} |\mathsf{IS}(g,k)|$.

Our reduction is illustrated in Figure 3. Given the graph g (depicted in the leftmost part), we construct $n+2$ graphs by adding new vertices and edges to g. For each such graph, we build a database that contains an endogenous fact $R(v)$ for every left vertex v, an endogenous fact $T(u)$ for every right vertex u, and an exogenous fact $S(v,u)$ for every edge (v,u). In each constructed database D_j, the fact f represents a new left node, and we compute $\text{Shapley}(D_j, \mathsf{q_{RST}}, f)$. In D_0, the node of f is connected to every right vertex. We use $\text{Shapley}(D_0, \mathsf{q_{RST}}, f)$ to compute a specific value that we refer to later on. For $j = 1, \dots, n+1$, the database D_j is obtained from g by adding f and facts of j new right nodes, all connected to f. We show the following for all $j = 1, \dots, n+1$.

$$\text{Shapley}(D_j, \mathsf{q_{RST}}, f) = 1 - \frac{c_j \cdot v_0 + \sum_{k=0}^{n} |\mathsf{IS}(g,k)| \cdot k!(n+j-k)!}{(n+j+1)!}$$

where v_0 is a value computed using $\text{Shapley}(D_0, \mathsf{q_{RST}}, f)$, and c_j is a constant that depends on j. From these equations we extract a system $Ax = y$ of $n+1$ equations over $n+1$ variables (i.e., $|\mathsf{IS}(g,0)|, \dots, |\mathsf{IS}(g,n)|$), where each S_j stands for $\text{Shapley}(D_j, \mathsf{q_{RST}}, f)$.

$$\begin{pmatrix} 0!(n+1)! & 1!n! & \dots & n!1! \\ 0!(n+2)! & 1!(n+1)! & \dots & n!2! \\ \vdots & \vdots & \vdots & \vdots \\ 0!(2n+1)! & 1!(2n)! & \dots & n!(n+1)! \end{pmatrix} \begin{pmatrix} |\mathsf{IS}(g,0)| \\ |\mathsf{IS}(g,1)| \\ \vdots \\ |\mathsf{IS}(g,n)| \end{pmatrix} = \begin{pmatrix} (n+2)!S_1 - c_1 v_0 \\ (n+3)!S_2 - c_2 v_0 \\ \vdots \\ (2n+2)!S_{n+1} - c_{n+1} v_0 \end{pmatrix}$$

By an elementary algebraic manipulation of A, we obtain the matrix with the coefficients $a_{i,j} = (i+j+1)!$ that Bacher [3] proved to be non-singular (and, in fact, that $\prod_{i=0}^{n-1} i!(i+1)!$ is its determinant). We then solve the system as discussed earlier to obtain $|\mathsf{IS}(g,k)|$.

4.2 Aggregates over Conjunctive Queries

Next, we study the complexity of aggregate-relational queries, where the internal relational query is a CQ. We begin with hardness. The following theorem generalizes the hardness side of Theorem 9 and states that it is $\text{FP}^{\#\text{P}}$-complete to compute $\text{Shapley}(D, \alpha, f)$ whenever α is of the form $\gamma[q]$, as defined in Section 2, and q is a non-hierarchical CQ without self-joins. The only exception is when α is a *constant* numerical query (i.e., $\alpha(D) = \alpha(D')$ for all databases D and D'); in that case, $\text{Shapley}(D, \alpha, f) = 0$ always holds.

▶ **Theorem 13.** *Let $\alpha = \gamma[q]$ be a fixed aggregate-relational query where q is a non-hierarchical CQ without self-joins. Computing* $\text{Shapley}(D, \alpha, f)$*, given D and f, is* $\text{FP}^{\#\text{P}}$*-complete, unless α is constant.*

For instance, it follows from Theorem 13 that, whenever q is a non-hierarchical CQ without self-joins, it is $\text{FP}^{\#\text{P}}$-complete to compute the Shapley value for the aggregate-relational queries $\mathsf{count}[q]$, $\mathsf{sum}\langle\varphi\rangle[q]$, $\mathsf{max}\langle\varphi\rangle[q]$, and $\mathsf{min}\langle\varphi\rangle[q]$, unless $\varphi(\vec{c}) = 0$ for all databases D and tuples $\vec{c} \in q(D)$. Additional examples follow.

▶ **Example 14.** Consider the numerical query α_3 from Example 3. Since q_4 is not hierarchical, Theorem 13 implies that computing $\text{Shapley}(D, \alpha_4, f)$ is $\text{FP}^{\#\text{P}}$-complete. Actually, computing $\text{Shapley}(D, \alpha, f)$ is $\text{FP}^{\#\text{P}}$-complete for any non-constant aggregate-relational query over q_4. Hence, computing the Shapley value w.r.t. $\mathsf{count}[q_4]$ (which counts the number of papers in Citations with an author from California) or w.r.t. $\mathsf{max}\langle[2]\rangle[q_4]$ (which calculates the number of citations for the most cited paper by a Californian) is $\text{FP}^{\#\text{P}}$-complete as well. ◀

To prove hardness in Theorem 13, we break q into connected components $q_1, \dots, q_m$, such that $\mathsf{Vars}(q_i) \cap \mathsf{Vars}(q_j) = \emptyset$ for all $i, j \in \{1, \dots, m\}$. Since q is non-hierarchical, at least one of these connected components is non-hierarchical. We assume, without loss of generality, that this is q_1. Next, since α is not a constant function, there exists a database $\widetilde{D}$ such that $\alpha(\widetilde{D}) \neq \alpha(\emptyset)$. We select one answer $\vec{a}$ from $q(\widetilde{D})$ and substitute the free variables of q_1 with the corresponding constants from $\vec{a}$ to obtain the Boolean CQ q_1'. Theorem 9 states that computing $\text{Shapley}(D, q_1', f)$ is $\text{FP}^{\#\text{P}}$-complete. We then reduce the problem of computing $\text{Shapley}(D, q_1', f)$ to the problem of computing $\text{Shapley}(D, \alpha, f)$, and show that

$$\text{Shapley}(D, q_1', f) = \frac{\text{Shapley}(D', \alpha, f)}{\alpha(\widetilde{D}) - \alpha(\emptyset)}$$

where D' is a database obtained by combining facts from D with facts from $\widetilde{D}$. As usual, the full proof is given in the extended version of the paper [21].

Interestingly, it turns out that Theorem 13 captures precisely the hard cases for computing the Shapley value w.r.t. any summation over CQs without self-joins. In particular, the following argument shows that $\text{Shapley}(D, \mathsf{sum}\langle\varphi\rangle[q], f)$ can be computed in polynomial time if q is a hierarchical CQ without self-joins. Let $q = q(\vec{x})$ be an arbitrary CQ. For $\vec{a} \in q(D)$, let $q_{[\vec{x}\to\vec{a}]}$ be the Boolean CQ obtained from q by substituting every free variable x_j with the value of x_j in $\vec{a}$. Hence, we have that $\mathsf{sum}\langle\varphi\rangle[q] = \sum_{\vec{a}\in q(D)} \varphi(\vec{a}) \cdot q_{[\vec{x}\to\vec{a}]}$. The linearity of the Shapley value (stated as a fundamental property in Section 3) implies that

$$\text{Shapley}(D, \mathsf{sum}\langle\varphi\rangle[q], f) = \sum_{\vec{a}\in q(D)} \varphi(\vec{a}) \cdot \text{Shapley}(D, q_{[\vec{x}\to\vec{a}]}, f)\,. \tag{5}$$

Then, from Theorem 9 we conclude that if q is a hierarchical CQ with self-joins, then $\text{Shapley}(D, q_{[\vec{x}\to\vec{a}]}, f)$ can be computed in polynomial time for every $\vec{a} \in q(D)$. Hence, we have the following corollary of Theorem 9.

▶ **Corollary 15.** *Let q be a hierarchical CQ without self-joins. If α is an aggregate-relational query* $\mathsf{sum}\langle\varphi\rangle[q]$*, then* $\text{Shapley}(D, \alpha, f)$ *can be computed in polynomial time, given D and f. In particular,* $\text{Shapley}(D, \mathsf{count}[q], f)$ *can be computed in polynomial time.*

Together with Theorem 13, we get a full dichotomy for $\mathsf{sum}\langle\varphi\rangle[q]$ over CQs without self-joins.

The complexity of computing $\text{Shapley}(D, \alpha, f)$ for other aggregate-relational queries remains an open problem for the general case where q is a hierarchical CQ without self-joins. We can, however, state a positive result for $\mathsf{max}\langle\varphi\rangle[q]$ and $\mathsf{min}\langle\varphi\rangle[q]$ for the special case where q consists of a single atom (i.e., aggregation over a single relation).

▶ **Proposition 16.** *Let q be a CQ with a single atom. Then,* $\text{Shapley}(D, \mathsf{max}\langle\varphi\rangle[q], f)$ *and* $\text{Shapley}(D, \mathsf{min}\langle\varphi\rangle[q], f)$ *can be computed in polynomial time.*

As an example, if α is the query $\mathsf{max}\langle[2]\rangle[q]$, where q is given by $q(x, y)$:- $\textsc{Citations}(x, y)$, then we can compute in polynomial time $\text{Shapley}(D, \alpha, f)$, determining the responsibility of each publication (in our running example) to the maximum number of citations.

4.3 Approximation

In computational complexity theory, a conventional feasibility notion of arbitrarily tight approximations is via the *Fully Polynomial-Time Approximation Scheme*, FPRAS for short. Formally, an FPRAS for a numeric function f is a randomized algorithm $A(x, \epsilon, \delta)$, where x is an input for f and $\epsilon, \delta \in (0, 1)$, that returns an ϵ-approximation of $f(x)$ with probability $1 - \delta$ (where the probability is over the randomness of A) in time polynomial in x, $1/\epsilon$ and $\log(1/\delta)$. To be more precise, we distinguish between an *additive* (or *absolute*) FPRAS:

$$\Pr\left[f(x) - \epsilon \leq A(x, \epsilon, \delta) \leq f(x) + \epsilon\right] \geq 1 - \delta\,,$$

and a *multiplicative* (or *relative*) FPRAS:

$$\Pr\left[\frac{f(x)}{1+\epsilon} \leq A(x, \epsilon, \delta) \leq (1+\epsilon)f(x)\right] \geq 1 - \delta\,.$$

Using the Chernoff-Hoeffding bound, we easily get an additive FPRAS of $\text{Shapley}(D, q, f)$ when q is *any* monotone Boolean query computable in polynomial time, by simply taking the ratio of successes over $O(\log(1/\delta)/\epsilon^2)$ trials of the following experiment:

1. Select a random permutation $(f_1, \dots, f_n)$ over the set of all endogenous facts.
2. Suppose that $f = f_i$, and let $D_{i-1} = D_{\mathsf{x}} \cup \{f_1, \dots, f_{i-1}\}$. If $q(D_{i-1})$ is false and $q(D_{i-1} \cup \{f\})$ is true, then report "success;" otherwise, "failure."

In general, an additive FPRAS of a function f is not necessarily a multiplicative one, since $f(x)$ can be very small. For example, we can get an additive FPRAS of the satisfaction of a propositional formula over Boolean i.i.d. variables by, again, sampling the averaging, but there is no multiplicative FPRAS for such formulas unless BPP = NP. Nevertheless, the situation is different for $\text{Shapley}(D, q, f)$ when q is a CQ, since the Shapley value is never too small (assuming data complexity).

▶ **Proposition 17.** *Let q be a fixed Boolean CQ. There is a polynomial p such that for all databases D and endogenous facts f of D it is the case that* $\text{Shapley}(D, q, f)$ *is either zero or at least* $1/(p(|D|))$.

Proof. We denote $m = |D_{\mathsf{n}}|$. If there is no subset S of D_{n} such that f is a counterfactual cause for q w.r.t. S, then $\text{Shapley}(D, q, f) = 0$. Otherwise, let S be a minimal such set (i.e., for every $S' \subset S$, we have that $(S' \cup D_{\mathsf{x}}) \not\models q$). Clearly, it holds that $S \leq k$, where k is the number of atoms of q. The probability to choose a permutation σ, such that σ_f is exactly $S \setminus \{f\}$ is $\frac{(|S|-1)!(m-|S|)!}{m!} \geq \frac{(m-k)!}{m!}$ (recall that σ_f is the set of facts that appear before f in σ). Hence, we have that $\text{Shapley}(D, q, f) \geq \frac{1}{(m-k+1)\cdot\ldots\cdot m}$, and that concludes our proof. ◀

It follows that whenever $\text{Shapley}(D, q, f) = 0$, the above additive approximation is also zero, and when $\text{Shapley}(D, q, f) > 0$, the additive FPRAS also provides a multiplicative FPRAS. Hence, we have the following.

▶ **Corollary 18.** *For every fixed Boolean CQ, the Shapley value has both an additive and a multiplicative FPRAS.*

Interestingly, Corollary 18 generalizes to a multiplicative FPRAS for summation (including counting) over CQs. By combining Corollary 18 with Equation (5), we immediately obtain a multiplicative FPRAS for $\text{Shapley}(D, \mathsf{sum}\langle\varphi\rangle[q], f)$, in the case where all the features $\varphi(\vec{a})$ in the summation have the same sign (i.e., they are either all negative or all non-negative). In particular, there is a multiplicative FPRAS for $\text{Shapley}(D, \mathsf{count}[q], f)$.

▶ **Corollary 19.** *For every fixed CQ q,* $\text{Shapley}(D, \mathsf{sum}\langle\varphi\rangle[q], f)$ *has a multiplicative FPRAS if either $\varphi(\vec{a}) \geq 0$ for all $\vec{a} \in q(D)$ or $\varphi(\vec{a}) \leq 0$ for all $\vec{a} \in q(D)$.*

Observe that the above FPRAS results allow the CQ q to have self-joins. This is in contrast to the complexity results we established in the earlier parts of this section, regarding exact evaluation. In fact, an easy observation is that Proposition 17 continues to hold when considering *unions of conjunctive queries* (UCQs). Therefore, Corollaries 18 and 19 remain correct in the case where q is a UCQ.

5 Related Measures

Causality and causal responsibility [15, 28] have been applied in data management [24], defining a fact as a *cause* for a query result as follows: For an instance $D = D_{\mathsf{x}} \cup D_{\mathsf{n}}$, a fact $f \in D_{\mathsf{n}}$ is an *actual cause* for a Boolean CQ q, if there exists $\Gamma \subseteq D_{\mathsf{n}}$, called a *contingency set* for f, such that f is a counterfactual cause for q in $D \setminus \Gamma$. The responsibility of an actual cause f for q is defined by $\rho(f) := \frac{1}{|\Gamma|+1}$, where $|\Gamma|$ is the size of a smallest contingency set for f. If f is not an actual cause, then $\rho(f)$ is zero [24]. Intuitively, facts with higher responsibility provide stronger explanations.[2]

▶ **Example 20.** Consider the database of our running example, and the query q_1 from Example 2. The fact f_1^{a} an actual cause with minimal contingency set $\Gamma = \{f_2^{\mathrm{a}}, f_3^{\mathrm{a}}, f_4^{\mathrm{a}}\}$. So, its responsibility is $\frac{1}{4}$. Similarly, f_2^{a}, f_3^{a} and f_4^{a} are actual causes with responsibility $\frac{1}{4}$.

▶ **Example 21.** Consider the database G and the query p_{ab} from Example 7. All facts in G are actual causes since every fact appears in a path from a to b. It is easy to verify that all the facts in D have the same causal responsibility, $\frac{1}{3}$, which may be considered as counter-intuitive given that e_1 provides a direct path from a to b.

As shown in Example 7, the Shapley value gives a more intuitive degree of contribution of facts to the query result than causal responsibility. Actually, Example 7 was used in [31] as a motivation to introduce an alternative to the notion of causal responsibility, that of *causal effect*, that we now briefly review.

To quantify the contribution of a fact to the query result, Salimi et al. [31] view the database as a tuple-independent probabilistic database where the probability of each endogenous fact is 0.5 and the probability of each exogenous fact is 1 (i.e., it is certain). The *causal effect* of a fact $f \in D_{\mathsf{n}}$ on a numerical query α is a difference of expected values [31]:

$$\text{CE}(D, \alpha, f) \stackrel{\text{def}}{=} \mathbb{E}(\alpha(D) \mid f) - \mathbb{E}(\alpha(D) \mid \neg f)\,.$$

where f is the event that the fact f is present in the database, and $\neg f$ is the event that the fact f is absent from the database.

[2] These notions can be applied to any monotonic query (i.e., whose answer set can only grow when the database grows, e.g., UCQs and Datalog queries) [4, 5].

▶ **Example 22.** Consider again the database of our running example, and the query q_1 from Example 2. We compute $\mathrm{CE}(D, q_1, f_1^a)$. It holds that: $\mathbb{E}(q_1 \mid \neg f_1^a) = 0 \cdot P(q_1 = 0 \mid \neg f_1^a) + 1 \cdot P(q_1 = 1 \mid \neg f_1^a) = 1 - P(\neg f_2^a \wedge \neg f_3^a \wedge \neg f_4^a) = \frac{7}{8}$. Similarly, we have that $\mathbb{E}(q_1 \mid f_1^a) = P(q_1 = 1 \mid f_1^a) = 1$. Then, $\mathrm{CE}(D, q_1, f_1^a) = 1 - \frac{7}{8} = \frac{1}{8}$. Using similar computations we obtain that $\mathrm{CE}(D, q_1, f_2^a) = \mathrm{CE}(D, q_1, f_3^a) = \mathrm{CE}(D, q_1, f_4^a) = \frac{1}{8}$.

For G and p_{ab} of Example 7, we have that $\mathrm{CE}(G, p_{ab}, e_1) = 0.65625$, $\mathrm{CE}(G, p_{ab}, e_2) = \mathrm{CE}(G, p_{ab}, e_3) = 0.21875$, $\mathrm{CE}(G, p_{ab}, e_4) = \mathrm{CE}(G, p_{ab}, e_5) = \mathrm{CE}(G, p_{ab}, e_6) = 0.09375$. ◀

Although the values in the two examples above are different from the Shapley values computed in Example 6 and Example 7, respectively, if we order the facts according to their contribution to the query result, we will obtain the same order in both cases. Note that unlike the Shapley value, for causal effect the sum of the values over all facts is not equal to the query result on the whole database. In the next example we consider aggregate queries.

▶ **Example 23.** Consider the query α_1 of Example 3. If f_1^a is in the database, then the result can be either 20, 28, or 40. If f_1^a is absent, then the query result can be either 0, 8, or 20. By computing the expected value in both cases, we obtain that $\mathrm{CE}(D, \alpha_1, f_1^a) = 20$. Similarly, it holds that $\mathrm{CE}(D, \alpha_1, f_2^a) = \mathrm{CE}(D, \alpha_1, f_4^a) = 1$, and $\mathrm{CE}(D, \alpha_1, f_3^a) = 14$. ◀

Interestingly, the causal effect coincides with a well known wealth-distribution function in cooperative games, namely the *Banzhaf Power Index* (BPI) [11, 18, 19]. This measure is defined similarly to the definition of the Shapley value in Equation (3), except that we replace the ratio $\frac{|B|! \cdot (|A| - |B| - 1)!}{|A|!}$ with $\frac{1}{2^{|A|-1}}$.

▶ **Proposition 24.** *Let α be a numerical query, D be a database, and $f \in D_{\mathsf{n}}$. Then,*

$$\mathrm{CE}(D, \alpha, f) = \frac{1}{2^{|D_{\mathsf{n}}|-1}} \cdot \sum_{E \subseteq (D_{\mathsf{n}} \setminus \{f\})} [\alpha(D_{\mathsf{x}} \cup E \cup \{f\}) - \alpha(D_{\mathsf{x}} \cup E)]$$

Hence, the causal effect coincides with the BPI.

We conjecture that *all* of the complexity results (exact and approximate) obtained in this work for the Shapley value apply to the causal effect (and BPI), with some of them being easier to obtain than for the Shapley value, via a connection to probabilistic databases [34].

6 Conclusions

We introduced the problem of quantifying the contribution of database facts to query results via the Shapley value. We investigated the complexity of the problem for Boolean CQs and for aggregates over CQs. Our dichotomy in the complexity of the problem establishes that computing the exact Shapley value is often intractable. Nevertheless, we also showed that the picture is far more optimistic when allowing approximation with strong precision guarantees.

Many questions, some quite fundamental, remain for future investigation. While we have a thorough understanding of the complexity for Boolean CQs without self-joins, very little is known in the presence of self-joins. For instance, the complexity is open even for the simple query $q() \text{ :- } R(x, y), R(y, z)$. We also have just a partial understanding of the complexity for aggregate functions over CQs, beyond the general hardness result for non-hierarchical queries (Theorem 13). In particular, it is important to complete the complexity analysis for maximum and minimum, and to investigate other common aggregate functions such as average, median, percentile, and standard deviation. Another direction is to investigate whether and how properties of the database, such as low treewidth, can reduce the (asymptotic and empirical) running time of computing the Shapley value. Interestingly, the implication of a low treewidth to Shapley computation has been studied for a different problem [13].

References

1 Robert J Aumann and Roger B Myerson. Endogenous formation of links between players and of coalitions: An application of the Shapley value. In *Networks and Groups*, pages 207–220. Springer, 2003.

2 Haris Aziz and Bart de Keijzer. Shapley meets Shapley. In *STACS*, pages 99–111, 2014.

3 Roland Bacher. Determinants of matrices related to the Pascal triangle. *Journal de Théorie des Nombres de Bordeaux*, 14, January 2002.

4 Leopoldo E. Bertossi and Babak Salimi. Causes for query answers from databases: Datalog abduction, view-updates, and integrity constraints. *Int. J. Approx. Reasoning*, 90:226–252, 2017.

5 Leopoldo E. Bertossi and Babak Salimi. From Causes for Database Queries to Repairs and Model-Based Diagnosis and Back. *Theory Comput. Syst.*, 61(1):191–232, 2017.

6 Sara Cohen, Werner Nutt, and Yehoshua Sagiv. Deciding equivalences among conjunctive aggregate queries. *J. ACM*, 54(2):5, 2007.

7 Vincent Conitzer and Tuomas Sandholm. Computing Shapley Values, Manipulating Value Division Schemes, and Checking Core Membership in Multi-issue Domains. In *AAAI*, pages 219–225. AAAI Press, 2004.

8 Nilesh N. Dalvi, Christopher Ré, and Dan Suciu. Probabilistic databases: diamonds in the dirt. *Commun. ACM*, 52(7):86–94, 2009.

9 Nilesh N. Dalvi and Dan Suciu. Efficient Query Evaluation on Probabilistic Databases. In *VLDB*, pages 864–875. Morgan Kaufmann, 2004.

10 Nilesh N. Dalvi and Dan Suciu. The dichotomy of probabilistic inference for unions of conjunctive queries. *J. ACM*, 59(6):30:1–30:87, 2012.

11 Pradeep Dubey and Lloyd S. Shapley. Mathematical Properties of the Banzhaf Power Index. *Mathematics of Operations Research*, 4(2):99–131, 1979.

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

13 Gianluigi Greco, Francesco Lupia, and Francesco Scarcello. Structural Tractability of Shapley and Banzhaf Values in Allocation Games. In *IJCAI*, pages 547–553, 2015.

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

15 Joseph Y. Halpern. *Actual Causality.* The MIT Press, 2016.

16 Joseph Y. Halpern and Judea Pearl. Causes and Explanations: A Structural-Model Approach: Part 1: Causes. In *UAI*, pages 194–202, 2001.

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

18 Werner Kirsch and Jessica Langner. Power indices and minimal winning coalitions. *Social Choice and Welfare*, 34(1):33–46, January 2010.

19 Dennis Leech. Power indices and probabilistic voting assumptions. *Public Choice*, 66(3):293–299, September 1990.

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

21 Ester Livshits, Leopoldo E. Bertossi, Benny Kimelfeld, and Moshe Sebag. The Shapley Value of Tuples in Query Answering. *CoRR*, abs/1904.08679, 2019.

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

23 Alexandra Meliou, Wolfgang Gatterbauer, Joseph Y. Halpern, Christoph Koch, Katherine F. Moore, and Dan Suciu. Causality in Databases. *IEEE Data Eng. Bull.*, 33(3):59–67, 2010.

24 Alexandra Meliou, Wolfgang Gatterbauer, Katherine F. Moore, and Dan Suciu. The Complexity of Causality and Responsibility for Query Answers and non-Answers. *PVLDB*, 4(1):34–45, 2010.

25 Alexandra Meliou, Wolfgang Gatterbauer, Katherine F. Moore, and Dan Suciu. WHY so? or WHY no? functional causality for explaining query answers. In *MUD*, volume WP10-04 of *CTIT Workshop Proceedings Series*, pages 3–17. CTIT, 2010.

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

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

28 Judea Pearl. *Causality: Models, Reasoning and Inference*. Cambridge University Press, New York, NY, USA, 2nd edition, 2009.

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

30 Alvin E Roth. *The Shapley value: essays in honor of Lloyd S. Shapley*. Cambridge University Press, 1988.

31 Babak Salimi, Leopoldo E. Bertossi, Dan Suciu, and Guy Van den Broeck. Quantifying Causal Effects on Query Answering in Databases. In *TAPP*, 2016.

32 Lloyd Shapley and Martin Shubik. A Method for Evaluating the Distribution of Power in a Committee System. *American Political Science Review*, 48(03):787–792, 1954.

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

34 Dan Suciu, Dan Olteanu, Christopher Ré, and Christoph Koch. *Probabilistic Databases*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011.

35 Seinosuke Toda. PP is as hard as the polynomial-time hierarchy. *SIAM J. Comput.*, 20(5):865–877, 1991.

36 Bruno Yun, Srdjan Vesic, Madalina Croitoru, and Pierre Bisquert. Inconsistency Measures for Repair Semantics in OBDA. In *IJCAI*, pages 1977–1983. ijcai.org, 2018.