Logical Methods in Computer Science
Volume 17, Issue 3, 2021, pp. 22:1–22:33
https://lmcs.episciences.org/

Submitted Dec. 01, 2020
Published Sep. 02, 2021

THE SHAPLEY VALUE OF TUPLES IN QUERY ANSWERING

ESTER LIVSHITS [a], LEOPOLDO BERTOSSI [b], BENNY KIMELFELD [a], AND MOSHE SEBAG [a]

[a] Technion, Haifa, Israel
e-mail address: esterliv@cs.technion.ac.il, bennyk@cs.technion.ac.il, moshesebag@campus.technion.ac.il

[b] Univ. Adolfo Ibañez and Millenium Inst. Foundations of Data (IMFD), Chile
e-mail address: leopoldo.bertossi@uai.cl

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.

1. Introduction

The Shapley value is named after Lloyd Shapley who introduced the value in a seminal 1952 article [Sha53]. 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. 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*.

Key words and phrases: Shapley value, query answering, conjunctive queries, aggregate queries.

LOGICAL METHODS IN COMPUTER SCIENCE
DOI:10.46298/LMCS-17(3:22)2021

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., [AM03, AdK14]), such as bargaining foundations in economics [Gul89], takeover corporate rights in law [Nen03], pollution responsibility in environmental management [PZ03, LZS15], influence measurement in social network analysis [NN11], and utilization of multiple Internet Service Providers (ISPs) in networks [MCL$^+$10]. Closest to database manegement is the application of the Shapley value to attributing a level of *inconsistency* to a statement in an inconsistent knowledge base [HK10, YVCB18]; the idea is natural: as wealth, adopt a measure of inconsistency for a set of logical sentences [GH06], and then associate to each sentence its Shapley value. The Shapley value has also become relevant in machine learning, as the SHAP approach to providing numerical scores to feature values of entities under classification [LL17, LEC$^+$20].

In this article, 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 [MGMS10a, SBSdB16], 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 [DS04]: if the CQ is hierarchical, then the problem is solvable in polynomial time, and otherwise, it is 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 [DS04], 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. [MGMS10a] 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 [HP01]. 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. [SBSdB16] 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 [DS79], and is different from the Shapley value [Rot88, Chapter 5]. We also show that, with the exception of a multiplicative approximation, our complexity results extend to the causal-effect measure. 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.

This article is the full version of a conference publication [LBKS20]. We have added all of the proofs and intermediate results that were excluded from the paper. In particular, we have added the full proof of our main result—the dichotomy in the complexity of computing the Shapley value for Boolean CQs (Theorem 4.1), as well as the proofs of our results for aggregate queries over CQs (Theorem 4.8 and Proposition 4.11). We also provide a complexity analysis for the computation of the causal-effect measure in Section 5.1 and, consequently, confirm a conjecture we made in the conference publication, stating that our complexity results are also applicable to the causal-effect measure (and Banzhaf power index).

The remainder of the article 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.

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 $\vec{c} = (c_1, \ldots, c_k)$ is a tuple of constants and $i \in \{1, \ldots, k\}$, 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^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* $R(c_1, \ldots, c_k)$, stating that

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.

the relation R^D over the k-ary relation symbol R contains the tuple $(c_1, \ldots, c_k) \in \mathsf{Const}^k$. 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 previous work on the explanations and responsibility of facts to query answers [MGMS10b, MGH+10], 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 $D = D_{\mathsf{x}} \cup D_{\mathsf{n}}$.

Example 2.1. 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[1]. 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 $q : \mathrm{DB}(\mathbf{S}) \to \mathcal{P}(\mathsf{Const}^k)$ (where $\mathcal{P}(\mathsf{Const}^k)$ is the power set of Const^k that consists of all subsets of 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 $\exists y_1 \cdots \exists y_m \theta(\vec{x}, y_1, \ldots, y_m)$, where θ is a conjunction of atomic formulas of the form $R(\vec{t})$ with variables among those in $\vec{x}, y_1, \ldots, y_m$. In the remainder of the article, a CQ q will be written shortly as a logic rule, that is, an expression of the form

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

[1]This example is used for illustrative purposes only and does not express any suggestion of a way to rank researchers.

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 $\vec{t}_1, \ldots, \vec{t}_n$. We call $q(\vec{x})$ the *head* of q, and $R_1(\vec{t}_1), \ldots, R_n(\vec{t}_n)$ 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 all homomorphisms from q to D onto the variables of $\vec{x}$, 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 $A_y \cap A_{y'} = \emptyset$ [DRS09]. For example, every CQ with at most two atoms is hierarchical. The smallest non-hierarchical CQ is the following.

$$\mathsf{q}_{\mathsf{RST}}() \text{ :- } R(x), S(x,y), T(y) \tag{2.1}$$

On the other hand, the query $q(x) \text{ :- } R(x), S(x,y), T(y)$, which has a single existential variable y, 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*) [MGH+10, MGMS10a] if the removal of f causes q to become false; that is, $D \models q$ and $D \setminus \{f\} \not\models q$.

Example 2.2. 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 $q_1'() \text{ :- } \textsc{Author}(x, \texttt{UCLA}), \textsc{Pub}(x,z)$, asking whether there is a publication with an author from UCLA, since D satisfies q_1', but if we remove Alice from the database, the query q_1' is not longer satisfied, 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 $\gamma : \mathcal{P}(\mathsf{Const}^k) \to \mathbb{R}$ 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 (hence, F is a numerical function on bags of numbers). 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 2.3. Following are examples of aggregate-relational queries over the relational queries of Example 2.2.

- $\alpha_1 \stackrel{\text{def}}{=} \mathsf{sum}\langle[2]\rangle[q_3]$ calculates the total number of citations of all published papers with an author in the database.
- $\alpha_2 \stackrel{\text{def}}{=} \mathsf{count}[q_3]$ counts the papers in Citations with an author in the database.
- $\alpha_3 \stackrel{\text{def}}{=} \mathsf{sum}\langle[2]\rangle[q_4]$ calculates the total number of citations of papers by Californians.
- $\alpha_4 \stackrel{\text{def}}{=} \mathsf{max}\langle[2]\rangle[q_3]$ 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 article generalize to grouping as in traditional modeling (e.g., [CNS07]). 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* [Sha53] 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.

$$\text{Shapley}(A, v, a) \stackrel{\text{def}}{=} \frac{1}{|A|!} \sum_{\sigma\in\Pi_A} \big(v(\sigma_a \cup \{a\}) - v(\sigma_a)\big) \tag{2.2}$$

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.

$$\text{Shapley}(A, v, a) \stackrel{\text{def}}{=} \sum_{B \subseteq A \setminus \{a\}} \frac{|B|! \cdot (|A| - |B| - 1)!}{|A|!} \Big(v(B \cup \{a\}) - v(B) \Big) \tag{2.3}$$

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 [Rot88].

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 3.1 (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 $\text{Shapley}(D, \alpha, f)$, is the value $\text{Shapley}(A, v, a)$ as given in (2.2), 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, $\text{Shapley}(D, \alpha, f)$ 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.

The choice of v is natural in that the first term collects answers where endogenous facts may interact with exogenous facts, but we remove those answers that come only from exogenous facts. As a special case, if q is a Boolean query, then $\text{Shapley}(D, q, f)$ is the same as the value $\text{Shapley}(D, \mathsf{count}[q], f)$. In this case, the corresponding cooperative game takes the values 0 and 1, and the Shapley value then coincides with the *Shapley-Shubik index* [SS54]. Some fundamental properties of the Shapley value [Sha53] are reflected here as follows:

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

Remark 3.2. Note that $\text{Shapley}(D, \alpha, f)$ is defined for a general numerical query α. The definition is immediately extendible to queries with *grouping* (producing tuples of database constants and numbers [CNS07]), where we would measure the responsibility of f for an answer tuple $\vec{a}$ and write something like $\text{Shapley}(D, \alpha, \vec{a}, f)$. 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 3.3. We begin with a Boolean CQ, and specifically q_1 from Example 2.2. 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: $\text{Shapley}(D, q_1, f_5^\text{a}) = 0$. 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.

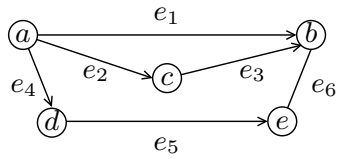
While Alice, Bob, Cathy and David have the same Shapley value for q_1, things change if we consider the relation pub endogenous as well: the Shapley value of Alice and Cathy will be higher than Bob's and David's values, since they have more publications. Specifically, the fact f_1^{a}, for example, will 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 $\{f_2^{\mathrm{a}}, f_3^{\mathrm{p}}\}$, $\{f_3^{\mathrm{a}}, f_4^{\mathrm{p}}\}$, $\{f_3^{\mathrm{a}}, f_5^{\mathrm{p}}\}$, and $\{f_4^{\mathrm{a}}, f_6^{\mathrm{p}}\}$ 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:

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

We can similarly compute the Shapley value for the rest of the authors, concluding that $\mathrm{Shapley}(D, q_1, f_2^{\mathrm{a}}) = \mathrm{Shapley}(D, q_1, f_4^{\mathrm{a}}) = \frac{241}{2520}$ and $\mathrm{Shapley}(D, q_1, f_3^{\mathrm{a}}) = \frac{442}{2520}$. 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. [SBSdB16], illustrates the Shapley value on (Boolean) graph reachability.

Example 3.4. Consider the following database G defined via the relation symbol Edge/2.



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 $\mathrm{Shapley}(G, p_{ab}, e_i)$ 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 $\mathrm{Shapley}(G, p_{ab}, e_1) > \mathrm{Shapley}(G, p_{ab}, e_2) > \mathrm{Shapley}(G, p_{ab}, e_4)$.

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:

$$\begin{aligned}\text{Shapley}(G, p_{ab}, e_1) &= \frac{(6-0-1)!\times 0!}{6!} + 5 \times \frac{(6-1-1)!\times 1!}{6!} + \\ &\quad \left(\binom{5}{2} - 1\right) \times \frac{(6-2-1)!2!}{6!} + \left(\binom{5}{3} - 4\right) \times \frac{(6-3-1)!3!}{6!} \\ &= \frac{1}{6} + 5 \times \frac{1}{30} + (10-1) \times \frac{1}{60} + (10-4) \times \frac{1}{60} = \frac{35}{60}\end{aligned}$$

Similarly, there are 2^5 subsets of G that do not contain e_2, and among them, the subsets that satisfy p_{ab} are the supersets of $\{e_1\}$ and $\{e_4, e_5, e_6\}$. Then,

$$\text{Shapley}(G, p_{ab}, e_2) = \frac{(6-1-1)!1!}{6!} + 3 \times \frac{(6-2-1)!2!}{6!} + 3 \times \frac{(6-3-1)!3!}{6!} = \frac{8}{60}$$

A similar reasoning shows that $\text{Shapley}(G, p_{ab}, e_3) = \frac{8}{60}$. Finally, among the 2^5 subsets of G that do not contain e_4, those that satisfy p_{ab} are the supersets of $\{e_1\}$ and $\{e_2, e_3\}$, and it holds that:

$$\text{Shapley}(G, p_{ab}, e_4) = \frac{(6-2-1)!2!}{6!} + 2 \times \frac{(6-3-1)!3!}{6!} = \frac{3}{60}$$

Similarly, $\text{Shapley}(G, p_{ab}, e_5) = \text{Shapley}(G, p_{ab}, e_6) = \frac{3}{60}$. □

Lastly, we consider aggregate functions over conjunctive queries.

Example 3.5. We consider the queries α_1, α_2, and α_4 from Example 2.3. Ellen has no publications; hence, $\text{Shapley}(D, \alpha_j, f_5^{\text{a}}) = 0$ 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, $\text{Shapley}(D, \alpha_1, f_1^{\text{a}}) = 20$ and $\text{Shapley}(D, \alpha_2, f_1^{\text{a}}) = 2$. 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 $\text{Shapley}(D, \alpha_1, f_3^{\text{a}}) = \frac{1}{3} \cdot 20 + \frac{2}{3} \cdot 12 = \frac{44}{3}$ and $\text{Shapley}(D, \alpha_2, f_3^{\text{a}}) = \frac{1}{3} \cdot 2 + \frac{2}{3} \cdot 1 = \frac{4}{3}$. Using similar computations we obtain that $\text{Shapley}(D, \alpha_1, f_2^{\text{a}}) = \text{Shapley}(D, \alpha_1, f_4^{\text{a}}) = \frac{8}{3}$ and $\text{Shapley}(D, \alpha_2, f_2^{\text{a}}) = \text{Shapley}(D, \alpha_2, f_4^{\text{a}}) = \frac{1}{3}$.

We conclude that 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 $f_3^{\text{a}} \in S$. We have that $\text{Shapley}(D, \alpha_4, f_1^{\text{a}}) = 10$, $\text{Shapley}(D, \alpha_4, f_2^{\text{a}}) = \text{Shapley}(D, \alpha_4, f_4^{\text{a}}) = 2$ and $\text{Shapley}(D, \alpha_4, f_3^{\text{a}}) = 4$ (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 $\text{Shapley}(D, \alpha_4, f_1^{\text{a}}) = 6$, 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 [DS04]: hierarchical CQs without self-joins are tractable, and non-hierarchical ones are intractable.

Theorem 4.1. *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 as input. 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 [Tod91]).

Example 4.2. Consider the query q_1 from Example 2.2. This query is hierarchical; hence, by Theorem 4.1, $\mathrm{Shapley}(D, q_1, f)$ can be calculated in polynomial time, given D and f. On the other hand, the query q_2 is not hierarchical. Thus, Theorem 4.1 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 4.1. While the tractability condition is the same as that of Dalvi and Suciu [DS04], 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 start 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.

4.1.1. *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 based on the following formula, 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 $D' \models q$.

$$\text{Shapley}(D,q,f) = \sum_{E\subseteq(D_\mathsf{n}\setminus\{f\})} \frac{|E|!(m-|E|-1)!}{m!}\Big(q(D_\mathsf{x}\cup E\cup\{f\}) - q(D_\mathsf{x}\cup E)\Big) \tag{4.1}$$

$$= \sum_{E\subseteq(D_\mathsf{n}\setminus\{f\})} \frac{|E|!(m-|E|-1)!}{m!}\Big(q(D_\mathsf{x}\cup E\cup\{f\})\Big) - \sum_{E\subseteq(D_\mathsf{n}\setminus\{f\})} \frac{|E|!(m-|E|-1)!}{m!}\Big(q(D_\mathsf{x}\cup E)\Big)$$

$$= \left(\sum_{k=0}^{m-1} \frac{k!(m-k-1)!}{m!} \times |\text{Sat}(D',q,k)|\right) - \left(\sum_{k=0}^{m-1} \frac{k!(m-k-1)!}{m!} \times |\text{Sat}(D\setminus\{f\},q,k)|\right)$$

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 4.1, it suffices to show the following.

Theorem 4.3. *Let q be a hierarchical Boolean CQ without self-joins. There is a polynomial-time algorithm for computing the number $|\text{Sat}(D,q,k)|$ of subsets E of D_n such that $|E| = k$ and $(D_\mathsf{x}\cup E)\models q$, given D and k as input.*

To prove Theorem 4.3, we show a polynomial-time algorithm for computing $|\text{Sat}(D,q,k)|$ 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 [CS04], regarding the computation of the Shapley value, 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 $D_\mathsf{n}\setminus C$ [CS04, 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 [DS12].

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 $\{v_1,\ldots,v_n\}$ 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

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

if $\mathsf{Vars}(q) = \emptyset$ **then**
  if $\exists a \in \mathsf{Atoms}(q)$ s.t. $a \notin D$ **then**
    return 0
  $A = \mathsf{Atoms}(q) \cap D_{\mathsf{n}}$
  if $|A| = k$ **then**
    return 1
  return 0
result $\leftarrow 0$
if q has a root variable **then**
  $x \leftarrow$ a root variable of q
  $V_x \leftarrow$ the set $\{v_1, \ldots, v_n\}$ of values for x
  for all $i \in \{1, \ldots, |V_x|\}$ **do**
    for all $j \in \{0, \ldots, k\}$ **do**
      $f_{i,j} = \leftarrow \mathsf{CntSat}(D^{v_i}, q_{[x \to v_i]}, j)$
  $P_1^\ell = f_{1,\ell}$ for all $\ell \in \{0, \ldots, k\}$
  for all $i \in \{2, \ldots, |V_x|\}$ **do**
    for all $\ell \in \{0, \ldots, k\}$ **do**
      $P_i^\ell \leftarrow 0$
      for all $j \in \{0, \ldots, \ell\}$ **do**
        $P_i^\ell \leftarrow P_i^\ell + P_{i-1}^{\ell-j} \cdot f_{i,j} + \left[\binom{\sum_{r=1}^{i-1} |D_{\mathsf{n}}^{v_r}|}{\ell-j} - P_{i-1}^{\ell-j}\right] \cdot f_{i,j} + P_{i-1}^{\ell-j} \cdot \left[\binom{|D_{\mathsf{n}}^{v_i}|}{j} - f_{i,j}\right]$
  result $\leftarrow P_n^k$
else
  let $q = q_1 \wedge q_2$ where $\mathsf{Vars}(q_1) \cap \mathsf{Vars}(q_2) = \emptyset$
  let D^1 and D^2 be the restrictions of D to the relations of q_1 and q_2, respectively
  for all k_1, k_2 s.t. $k_1 + k_2 = k$ **do**
    result $\leftarrow$ result $+ \mathsf{CntSat}(D^1, q_1, k_1) \cdot \mathsf{CntSat}(D^2, q_2, k_2)$
return result

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

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 $j \in \{0, \ldots, k\}$, 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 reduce the number of variables in q by substituting x with v_i. Then, for each $\ell \in \{0, \ldots, k\}$ it clearly holds that $P_1^\ell = f_{1,\ell}$. For each $i \in \{2, \ldots, |V_x|\}$ 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 $j \in \{0, \ldots, \ell\}$) 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 $(\bigcup_{r=1}^{i-1} D_{\mathsf{x}}^{v_r} \cup E_2) \models q$,

R	
A	B
1	2
1	3
2	1
3	1

S	
A	B
1	1
1	5
2	3
2	4

T	
A	B
1	1
2	2
3	3
5	6

U
A
1
2
3
4

FIGURE 3. The database of Example 4.5.

(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

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

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.

The correctness and efficiency of CntSat is stated in the following lemma.

Lemma 4.4. *Let q be a hierarchical Boolean CQ without self-joins. Then, $\mathsf{CntSat}(D, q, k)$ returns 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 as input. Moreover, $\mathsf{CntSat}(D, q, k)$ terminates in polynomial time in k and $|D|$.*

We have already established the correctness of the algorithm. Thus, we now consider the complexity claim of Lemma 4.4. The number of recursive calls in each step is polynomial in k and $|D|$. In particular, in the dynamic programming part of the algorithm (lines 12-20), we make $(k + 1) \cdot |V_x|$ recursive calls. Clearly, it holds that $|V_x| \leq |D|$. Furthermore, we make $2(k + 1)$ recursive calls in lines 23-26. Finally, in each recursive call, we reduce the number of variables in q by at least one. Thus, the depth of the reduction is bounded by the number of variables in query q, which is a constant when considering data complexity.

Example 4.5. We now illustrate the execution of $\mathsf{CntSat}(D, q, k)$ on the database D of Figure 3, the query $q() \coloneq R(x, y), S(x, z), T(w, w), U(w)$ and $k = 4$. We assume that all facts in D are endogenous. Since q does not have a root variable, the condition of line 9 does not hold. Hence, we start by considering the two disjoint sub-queries $q_1() \coloneq R(x, y), S(x, z)$

and $q_2()$:- $T(w,w), U(w)$ in line 23, and the corresponding databases D_1 that contains the relations R and S and D_2 that contains the relations T and U. Note that q_1 and q_2 indeed do not share any variables.

Each set of facts that satisfies q contains four facts of the form $R(a,b)$, $S(a,c)$, $T(d,d)$ and $U(d)$ for some values a, b, c, d. Clearly, it holds that $\{R(a,b), S(a,c)\} \models q_1$ and $\{T(d,d), U(d)\} \models q_2$; thus, we compute $\mathsf{CntSat}(D, q, 4)$ using 10 (that is, $2(k+1)$) recursive calls to CntSat.

$$\begin{aligned}\mathsf{CntSat}(D,q,4) = {} & \mathsf{CntSat}(D_1,q_1,0)\cdot\mathsf{CntSat}(D_2,q_2,4)\\ & + \mathsf{CntSat}(D_1,q_1,1)\cdot\mathsf{CntSat}(D_2,q_2,3)\\ & + \mathsf{CntSat}(D_1,q_1,2)\cdot\mathsf{CntSat}(D_2,q_2,2)\\ & + \mathsf{CntSat}(D_1,q_1,3)\cdot\mathsf{CntSat}(D_2,q_2,1)\\ & + \mathsf{CntSat}(D_1,q_1,4)\cdot\mathsf{CntSat}(D_2,q_2,0)\end{aligned}$$

Now, q_1 contains a root variable x; thus, in each recursive call with the query q_1, the condition of line 9 holds. We will illustrate the execution of this part of the algorithm using $\mathsf{CntSat}(D_1, q_1, 3)$. Note that in a homomorphism from $R(x,y)$ to D_1, the variable x is mapped to one of three values, namely $\mathtt{1}$, $\mathtt{2}$, or $\mathtt{3}$. Similarly, in a homomorphism from $S(x,z)$ to D_1, the value x is mapped to either $\mathtt{1}$ or $\mathtt{2}$. Hence, it holds that $V_x = \{\mathtt{1},\mathtt{2},\mathtt{3}\}$.

For each value a_i in V_x (where $a_1 = \mathtt{1}, a_2 = \mathtt{2}, a_3 = \mathtt{3}$), we consider the query $q_{[x\to a_i]}$ which is $R(a_i, y), S(a_i, z)$, and the database D^{a_i} containing the facts that use the value a_i for the variable x. That is, the database $D^{\mathtt{1}}$ contains the facts $\{R(\mathtt{1},\mathtt{2}), R(\mathtt{1},\mathtt{3}), S(\mathtt{1},\mathtt{1}), S(\mathtt{1},\mathtt{5})\}$, the database $D^{\mathtt{2}}$ contains the facts $\{R(\mathtt{2},\mathtt{1}), S(\mathtt{2},\mathtt{3}), S(\mathtt{2},\mathtt{4})\}$, and the database $D^{\mathtt{3}}$ contains the fact $\{R(\mathtt{3},\mathtt{1})\}$. Then, for each one of the three values, and for each $j \in \{0,\ldots,3\}$, we compute the number $f_{i,j}$ of subsets of size j of D^{a_i} that satisfy q, using the recursive call $\mathsf{CntSat}(D^{a_i}, q_{[x\to a_i]}, j)$. The reader can easily verify that the following holds.

$$\begin{array}{llll} f_{1,0}=0 & f_{1,1}=0 & f_{1,2}=4 & f_{1,3}=4\\ f_{2,0}=0 & f_{2,1}=0 & f_{2,2}=2 & f_{2,3}=1\\ f_{3,0}=0 & f_{3,1}=0 & f_{3,2}=0 & f_{3,3}=0\end{array}$$

Next, we compute, for each $i \in \{1,\ldots,3\}$ and $l \in \{0,\ldots,3\}$, the number P_i^l of subsets of size l of $\bigcup_{r=1}^{i} D^{a_r}$ that satisfy q. We begin with a_i (i.e., the value $\mathtt{1}$), in which case it holds that $P_1^l = f_{1,l}$. Hence, we have that:

$$P_1^0 = P_1^1 = 0 \qquad P_1^2 = P_1^3 = 4$$

Next, for each $l \in \{0,\ldots,3\}$, we compute the number P_2^l of subsets of $D^{\mathtt{1}} \cup D^{\mathtt{2}}$ that satisfy q. Each such subset contains j facts from $D^{\mathtt{2}}$ and $l-j$ facts for $D^{\mathtt{1}}$ for some $j \in \{0,\ldots,l\}$. Recall that $D^{\mathtt{1}}$ contains four facts and $D^{\mathtt{2}}$ contains three facts. Hence, we have the following computations for $l = 0$.

$$P_2^0 = 0$$

$$P_2^0 \mathrel{+}= P_1^0 \cdot f_{2,0} + \left[\binom{4}{0} - P_1^0\right] \cdot f_{2,0} + P_1^0 \cdot \left[\binom{3}{0} - f_{2,0}\right] = 0 + 0\cdot 0 + 1\cdot 0 + 0\cdot 1 = 0$$

In the first line we initialize P_2^0. Then, in the second line, we consider $j = 0$, which is the only possible j in this case. Next, for $l = 1$, we compute the following.

$$P_2^1 = 0$$
$$P_2^1 \mathrel{+}= P_1^1 \cdot f_{2,0} + \left[\binom{4}{1} - P_1^1\right] \cdot f_{2,0} + P_1^1 \cdot \left[\binom{3}{0} - f_{2,0}\right] = 0 + 0 \cdot 0 + 4 \cdot 0 + 0 \cdot 1 = 0$$
$$P_2^1 \mathrel{+}= P_1^0 \cdot f_{2,1} + \left[\binom{4}{0} - P_1^0\right] \cdot f_{2,1} + P_0^1 \cdot \left[\binom{3}{1} - f_{2,1}\right] = 0 + 0 \cdot 0 + 1 \cdot 0 + 0 \cdot 3 = 0$$

Here, in the second line, we consider $j = 0$ (i.e., choosing zero facts from D^2 and one fact from D^1), and in the third line we consider $j = 1$ (i.e., choosing one fact from D^2 and zero facts from D^1). Next, we have $l = 2$.

$$P_2^2 = 0$$
$$P_2^2 \mathrel{+}= P_1^2 \cdot f_{2,0} + \left[\binom{4}{2} - P_1^2\right] \cdot f_{2,0} + P_1^2 \cdot \left[\binom{3}{0} - f_{2,0}\right] = 0 + 4 \cdot 0 + 2 \cdot 0 + 4 \cdot 1 = 4$$
$$P_2^2 \mathrel{+}= P_2^1 \cdot f_{2,1} + \left[\binom{4}{1} - P_1^1\right] \cdot f_{2,1} + P_1^1 \cdot \left[\binom{3}{1} - f_{2,1}\right] = 4 + 0 \cdot 0 + 4 \cdot 0 + 0 \cdot 3 = 4$$
$$P_2^2 \mathrel{+}= P_2^0 \cdot f_{2,2} + \left[\binom{4}{0} - P_1^0\right] \cdot f_{2,2} + P_1^0 \cdot \left[\binom{3}{2} - f_{2,2}\right] = 4 + 0 \cdot 2 + 1 \cdot 2 + 0 \cdot 1 = 6$$

Finally, we consider $l = 3$.

$$P_2^3 = 0$$
$$P_2^3 \mathrel{+}= P_1^3 \cdot f_{2,0} + \left[\binom{4}{3} - P_1^3\right] \cdot f_{2,0} + P_1^3 \cdot \left[\binom{3}{0} - f_{2,0}\right] = 0 + 4 \cdot 0 + 0 \cdot 0 + 4 \cdot 1 = 4$$
$$P_2^3 \mathrel{+}= P_1^2 \cdot f_{2,1} + \left[\binom{4}{2} - P_1^2\right] \cdot f_{2,1} + P_1^2 \cdot \left[\binom{3}{1} - f_{2,1}\right] = 4 + 4 \cdot 0 + 2 \cdot 0 + 4 \cdot 3 = 16$$
$$P_2^3 \mathrel{+}= P_1^1 \cdot f_{2,2} + \left[\binom{4}{1} - P_1^1\right] \cdot f_{2,2} + P_1^1 \cdot \left[\binom{3}{2} - f_{2,2}\right] = 16 + 0 \cdot 2 + 4 \cdot 2 + 0 \cdot 1 = 24$$
$$P_2^3 \mathrel{+}= P_1^0 \cdot f_{2,3} + \left[\binom{4}{0} - P_1^0\right] \cdot f_{2,3} + P_1^0 \cdot \left[\binom{3}{3} - f_{2,3}\right] = 24 + 0 \cdot 1 + 1 \cdot 1 + 0 \cdot 0 = 25$$

We conclude that:

$$P_2^0 = P_2^1 = 0 \qquad P_2^2 = 6 \qquad P_2^3 = 25$$

Now, we can compute P_3^l for each $l \in \{0, \ldots, 3\}$ in a similar way, using the above values and the values $f_{3,j}$ that we have computed before. We omit the computations here. The final results are the following.

$$P_3^0 = P_3^1 = 0 \qquad P_3^2 = 6 \qquad P_2^3 = 31$$

Then, $\mathsf{CntSat}(D_1, q_1, 3)$ returns P_2^3 which is the number of subset of size 3 of D_1 that satisfy the query.

Finally, we illustrate the base case of the algorithm (that is, lines 1-7). To do that, we use the recursive call $\mathsf{CntSat}(D_2, q_2, 3)$ from the first step of the execution. Recall that $q_2() \coloneq T(w, w), U(w)$ and D_2 contains all the facts in T and U. The query q_2 contains a single variable w. In a homomorphism from $T(w, w)$ to D_2, this variable is mapped to one

of three values, namely 1, 2, or 3. Note that there is no homomorphism from $T(w, w)$ to the fact $T(5, 6)$; hence, the values 5 and 6 are not in V_w. In addition, in a homomorphism from $U(w)$ to D_2, the variable w is mapped to one of 1, 2, 3, or 4; thus, $V_w = \{1, 2, 3, 4\}$.

In every recursive call, we will substitute one of the values in V_w for w. One of the recursive calls will be $\mathsf{CntSat}(D_2^1, q_2', 2)$, where $q_2'() \text{ :- } T(1, 1), U(1)$. Here, D_2^1 contains every atom of q, and $k = |A|$; hence, the recursive call will return 1. On the other hand, the result of $\mathsf{CntSat}(D_2^1, q_2', 3)$ will be zero; as there are only two facts in D_2^1, while $k = 3$. The result of $\mathsf{CntSat}(D_2^1, q_2', 1)$ will also be zero, since in this case $k = 1$ and $|A| = 2$; thus, $k < |A|$. Finally, for the recursive call $\mathsf{CntSat}(D_2^4, q_2'', 2)$, where $q_2''() \text{ :- } T(4, 4), U(4)$, the result will be zero, as the fact $T(4, 4)$ is not in the database. □

4.1.2. *Hardness side.* We now give the proof of the hardness side of Theorem 4.1. Membership in $\mathrm{FP}^{\#\mathrm{P}}$ is straightforward since, as aforementioned in Equation (4.1), 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 [DS04], our proof of hardness consists of two steps. First, we prove the $\mathrm{FP}^{\#\mathrm{P}}$-hardness of computing $\mathrm{Shapley}(D, \mathsf{q}_{\mathsf{RST}}, f)$, where $\mathsf{q}_{\mathsf{RST}}$ is given in (2.1). Second, we reduce the computation of $\mathrm{Shapley}(D, \mathsf{q}_{\mathsf{RST}}, f)$ 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 [DS04], and we will give the proof here for completeness. The proof of the first step—hardness of computing $\mathrm{Shapley}(D, \mathsf{q}_{\mathsf{RST}}, f)$ (stated by Lemma 4.6), is considerably more involved than the corresponding proof of Dalvi and Suciu [DS04] that computing the probability of $\mathsf{q}_{\mathsf{RST}}$ in a tuple-independent probabilistic database (TID) is $\mathrm{FP}^{\#\mathrm{P}}$-hard. This is due to the coefficients of the Shapley value that do not seem to easily factor out.

Proposition 4.6. *Computing* $\mathrm{Shapley}(D, \mathsf{q}_{\mathsf{RST}}, f)$ *is* $\mathrm{FP}^{\#\mathrm{P}}$*-complete.*

Proof. The proof 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 [DS04] also proved the hardness of $\mathsf{q}_{\mathsf{RST}}$ (for the problem of query evaluation over TIDs) by reduction from #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 [DS12] 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. [AdK14] 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 $n + 2$ different input instances (D_j, f), for $j = 0, \ldots, n + 1$, of the problem of computing $\mathrm{Shapley}(D_j, \mathsf{q}_{\mathsf{RST}}, f)$, 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 $k = 0, \ldots, n$. We then show that the set of equations constitutes a non-singular

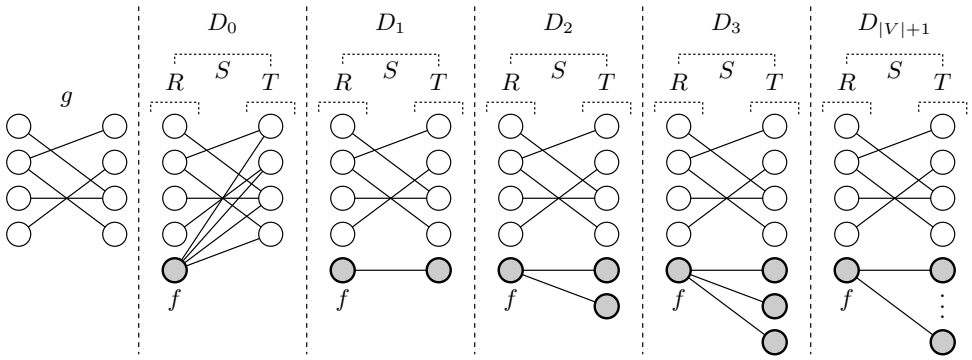


FIGURE 4. Constructions in the reduction of the proof of Lemma 4.6. Relations $R/1$ and $T/1$ consist of endogenous facts and $S/2$ consists of exogenous facts.

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 4. 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 $f = R(\mathsf{0})$ represents a new left node, and we compute $\text{Shapley}(D_j, \mathsf{q}_{\mathsf{RST}}, f)$. In D_0, the node of f is connected to every right vertex (i.e., we add an exogenous fact $S(\mathsf{0},u)$ for every right vertex u). We use this database to compute a specific value (from the Shapley value of f), as we explain next.

Instead of directly computing the Shapley value of f, we compute the complement of the Shapley value. To do that, we consider the permutations σ where f does not affect the query result. This holds in one of two cases:

(1) No fact of T appears before f in σ,
(2) At least one pair $\{R(v), T(u)\}$ of facts, such that there is a fact $S(u,v)$ in D_0, appears in σ before f.

The number of permutations where the first case holds is:

$$P_0^1 = \frac{(n+1)!}{n_T+1}$$

where n_T is the number of vertices on the right-hand side of the graph g (namely, the number of facts in T). This holds since each one of the facts of T and the fact f have an equal chance to be selected first (among these facts) in a random permutation. We are looking for the permutations where f is chosen before any fact of T; hence, we are looking at $1/(n_T+1)$ of all permutations.

Now, we compute the number of permutations σ where the second case holds. To do that, we have to count the permutations σ where σ_f corresponds to a set of vertices from g (the original graph) that is not an independent set. Let us denote by $|\mathsf{NIS}(g,k)|$ the number of subsets of vertices of size k from g that are not independent sets. Then, the number of

permutations satisfying the above is:

$$P_0^2 = \sum_{k=2}^{n} |\mathsf{NIS}(g,k)| \cdot k! \cdot (n-k)!$$

Recall that the fact f corresponds to a new vertex that does not occur in the original bipartite graph g; hence, for each k, we have k facts that appear before f in the permutation and $n+1-k-1 = n-k$ facts that appear after f. We can now express the Shapley value of f in terms of P_0^1 and P_0^2:

$$\text{Shapley}(D_0, \mathsf{q}_{\mathsf{RST}}, f) = 1 - \frac{P_0^1 + P_0^2}{(n+1)!}$$

Then, the value P_0^2 can be computed from $\text{Shapley}(D_0, \mathsf{q}_{\mathsf{RST}}, f)$ using the following formula.

$$P_0^2 = (1 - \text{Shapley}(D_0, \mathsf{q}_{\mathsf{RST}}, f)) \cdot (n+1)! - P_0^1$$

We will use this value later in our proof.

Next, for $j = 1, \ldots, n+1$, we construct a database D_j that is obtained from g by adding $f = R(\mathsf{0})$ and facts $T(0_1), \ldots, T(0_j)$ of j new right nodes, all connected to f (by adding an exogenous fact $S(\mathsf{0}, 0_i)$ for each $i \in \{1, \ldots, j\}$). We again compute the complement of the Shapley value of f, for each database D_j. The permutations where f does not affect the query result are those satisfying one of two properties:

(1) At least one pair $\{R(v), T(u)\}$ of facts for $u, v \in V$, such that there exists a fact $S(v,u)$ in D_j, appears in σ before f,
(2) No pair $\{R(v), T(u)\}$ of facts for $u, v \in V$, such that there exists a fact $S(v,u)$ in D_j, as well as none of the facts $T(0_1), \ldots, T(0_j)$ appear in σ before f.

Recall that V is the set of vertices of the original bipartite graph g.

Note that if the first condition holds, then it does not matter if we choose a fact from the set $\{T(0_1), \ldots, T(0_j)\}$ before choosing f or not (that is, the fact f will not affect the query result regardless of the positions of these facts). Hence, we first ignore these facts, and compute the number of permutations of the rest of the facts that satisfy the first condition:

$$\sum_{k=2}^{n} |\mathsf{NIS}(g,k)| \cdot k! \cdot (n-k)!$$

From each such permutation σ, we can then generate m_j permutations of all the $n+j+1$ facts in D_j by considering all the m_j possibilities to add the facts of $\{T(0_1), \ldots, T(0_j)\}$ to the permutation. Note that this is the same m_j for each permutation, and it holds that $m_j = \binom{n+j+1}{j} \cdot j!$ (i.e., we select j positions for the facts of $\{T(0_1), \ldots, T(0_j)\}$ and place them in these position in one of $j!$ possible permutations, while placing the rest of the facts in the remaining positions in the order defined by σ). Moreover, using this procedure we cover all the permutations of the facts in D_j that satisfy the first condition, since for each one of them there is a single corresponding permutation of the facts in $D_j \setminus \{T(0_1), \ldots, T(0_j)\}$. Hence, the number of permutations of the facts in D_j that satisfy the first property is

$$m_j \cdot \sum_{k=2}^{n} |\mathsf{NIS}(g,k)| \cdot k! \cdot (n-k)! = m_j \cdot P_0^2$$

Recall that we have seen earlier that the value P_0^2 can be computed from $\text{Shapley}(D_0, \mathsf{q}_{\mathsf{RST}}, f)$.

Next, we compute the number of permutations that satisfy the second property:

$$P_j = \sum_{k=0}^{n} |\mathsf{IS}(g,k)| \cdot k!(n+j-k)!$$

This holds since each permutation σ where σ_f does not contain any fact $T(0_j)$ and any pair $\{R(v), T(u)\}$ of facts such that there is a fact $S(u,v)$ in D_j, corresponds to an independent set of g. Hence, for each $j = 1, \ldots, n+1$ we get an equation of the form:

$$\text{Shapley}(D_j, \mathsf{q}_{\mathsf{RST}}, f) = 1 - \frac{m_j \cdot P_0^2 + P_j}{(n+j+1)!}$$

And we can compute P_j from $\text{Shapley}(D_j, \mathsf{q}_{\mathsf{RST}}, f)$ in the following way.

$$P_j = (n+j+1)!\big(1 - \text{Shapley}(D_j, \mathsf{q}_{\mathsf{RST}}, f)\big) - m_j \cdot P_0^2$$

From these equations we now extract a system $Ax = y$ of $n+1$ equations over $n+1$ variables (i.e., $|\mathsf{IS}(g,0)|, \ldots, |\mathsf{IS}(g,n)|$), where each S_j stands for $1 - \text{Shapley}(D_j, \mathsf{q}_{\mathsf{RST}}, f)$.

$$\begin{pmatrix} 0!(n+1)! & 1!n! & \ldots & n!1! \\ 0!(n+2)! & 1!(n+1)! & \ldots & n!2! \\ \vdots & \vdots & \vdots & \vdots \\ 0!(2n+1)! & 1!(2n)! & \ldots & n!(n+1)! \end{pmatrix} \begin{pmatrix} |\mathsf{IS}(g,0)| \\ |\mathsf{IS}(g,1)| \\ \vdots \\ |\mathsf{IS}(g,n)| \end{pmatrix} = \begin{pmatrix} (n+2)!S_1 - m_1 \cdot P_0^2 \\ (n+3)!S_2 - m_2 \cdot P_0^2 \\ \vdots \\ (2n+2)!S_{n+1} - m_{n+1} \cdot P_0^2 \end{pmatrix}$$

By an elementary algebraic manipulation of A (i.e., dividing each column j by the constant $j!$ and reversing the order of the columns), we obtain the matrix with the coefficients $a_{i,j} = (i+j+1)!$ that Bacher [Bac02] 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 $|\mathsf{IS}(g,k)|$ for each k, and, consequently, compute the value $\mathsf{IS}(g) = \sum_{k=0}^{n} \mathsf{IS}(g,k)$. □

Finally, we show that computing $\text{Shapley}(D, q, f)$ is hard for any non-hierarchical Boolean CQ q without self-joins, by constructing a reduction from the problem of computing $\text{Shapley}(D, \mathsf{q}_{\mathsf{RST}}, f)$. As aforementioned, our reduction is very similar to the corresponding reduction of Dalvi and Suciu [DS04], and we give it here for completeness. We will also use this result in Section 5.

Lemma 4.7. *Let q be a non-hierarchical Boolean CQ without self-joins. Then, computing* $\text{Shapley}(D, q, f)$ *is* $\text{FP}^{\#\text{P}}$*-complete*

Proof. We build a reduction from the problem of computing $\text{Shapley}(D, \mathsf{q}_{\mathsf{RST}}, f)$ to the problem of computing $\text{Shapley}(D, q, f)$. Since q is not hierarchical, there exist two variables $x, y \in \mathsf{Vars}(q)$, such that $A_x \cap A_y \neq \emptyset$, while $A_x \not\subseteq A_y$ and $A_y \not\subseteq A_x$; hence, we can choose three atoms α_x, α_y and $\alpha_{(x,y)}$ in q such that:

- $x \in \mathsf{Vars}(\alpha_x)$ and $y \notin \mathsf{Vars}(\alpha_x)$
- $y \in \mathsf{Vars}(\alpha_y)$ and $x \notin \mathsf{Vars}(\alpha_y)$
- $x, y \in \mathsf{Vars}(\alpha_{x,y})$

Recall that $\mathsf{Vars}(\alpha)$ is the set of variables that appear in the atom α.

Given an input database D to the first problem, we build an input database D' to our problem in the following way. Let c be an arbitrary constant that does not occur in D. For each fact $R(\mathsf{a})$ and for each atom $\alpha \in A_x \setminus A_y$, we generate a fact f over the relation corresponding to α by assigning the value a to the variable x and the value c to the rest of the variables in α. We then add the corresponding facts to D'. We define each new fact in

the relation of α_x to be endogenous if and only if the original fact from R is endogenous, and we define the rest of the facts to be exogenous.

Similarly, for each fact $T(\mathtt{b})$ and for each atom $\alpha \in A_y \setminus A_x$, we generate a fact f over the relation corresponding to α by assigning the value $\mathtt{b}$ to the variable y and the value $\mathtt{c}$ to the rest of the variables in α. Moreover, for each fact $S(\mathtt{a}, \mathtt{b})$ and for each atom $\alpha \in A_x \cap A_y$, we generate a fact f over the relation corresponding to α by assigning the value $\mathtt{a}$ to x, the value $\mathtt{b}$ to y and the value $\mathtt{c}$ to the rest of the variables in α. In both cases, we define the new facts in α_x and $\alpha_{x,y}$ to be endogenous if and only if the original fact is endogenous, and we define the rest of the facts to be exogenous. Finally, for each atom α in q that does not use the variables x and y (that is, $\alpha \notin A_x \cup A_y$), we add a single exogenous fact $R_\alpha(\mathtt{c}, \ldots, \mathtt{c})$ to the relation R_α corresponding to α.

We will now show that the Shapley value of each fact $R(\mathtt{a})$ in D w.r.t q_{RST} is equal to the Shapley value of the corresponding fact f over the relation of α_x in D' (i.e., the fact in the relation of α_x that has been generated using the value $\mathtt{a}$ that occurs in $R(\mathtt{a})$). The same holds for a fact $T(\mathtt{b})$ and its corresponding fact in the relation of α_y in D', and for a fact $S(\mathtt{a}, \mathtt{b})$ and its corresponding fact in the relation of $\alpha_{x,y}$ in D'.

By definition, the Shapley value of a fact f is the probability to select a random permutation σ in which the addition of the fact f changes the query result from 0 to 1 (i.e., f is a counterfactual cause for the query w.r.t. $\sigma_f \cup D_{\mathsf{x}}$). From the construction of D', it holds that the number of endogenous facts in D is the same as the number of endogenous facts in D'; hence, the total number of permutations of the facts in D is the same as the total number of permutations of the facts in D'. It is left to show that the number of permutations of the facts in D that satisfy the above condition is the same as the number of permutations of the facts in D' that satisfy the above condition w.r.t. the corresponding fact f'.

From the construction of D' it is straightforward that a subset E of D_{n} is such that $E \cup D_{\mathsf{x}} \models \mathsf{q}_{\mathsf{RST}}$ if and only if the subset E' of D'_{n} that contains for each fact $f \in E$ the corresponding fact $f' \in D'$ is such that $E' \cup D'_{\mathsf{x}} \models q$. Therefore, it also holds that if a fact f is a counterfactual cause for $\mathsf{q}_{\mathsf{RST}}$ w.r.t. $E \cup D_{\mathsf{x}}$, the corresponding fact f' is a counterfactual cause for q w.r.t. $E' \cup D'_{\mathsf{x}}$. Thus, the number of permutations of the endogenous facts in D in which f affects the result of $\mathsf{q}_{\mathsf{RST}}$ is equal to the number of permutations of the endogenous facts in D' in which f' changes the result of q. As aforementioned, the total number of permutations is the same for both D and D', and we conclude, from the definition of the Shapley value, that $\mathrm{Shapley}(D, \mathsf{q}_{\mathsf{RST}}, f) = \mathrm{Shapley}(D', q, f')$. □

4.2. **Aggregate Functions 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 4.1 and states that it is $\mathrm{FP}^{\#\mathrm{P}}$-complete to compute $\mathrm{Shapley}(D, \alpha, f)$ 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, $\mathrm{Shapley}(D, \alpha, f) = 0$ always holds.

Theorem 4.8. *Let $\alpha = \gamma[q]$ be a fixed aggregate-relational query where q is a non-hierarchical CQ without self-joins. Computing* $\mathrm{Shapley}(D, \alpha, f)$*, given D and f as input, is* $\mathrm{FP}^{\#\mathrm{P}}$*-complete, unless α is constant.*

Proof. Since α is not a constant function, there exists a database $\widetilde{D}$, such that $\alpha(\widetilde{D}) \neq \alpha(\emptyset)$. Let $\widetilde{D}$ be a minimal such database; that is, for every database D such that $q(D) \subset q(\widetilde{D})$ it holds that $\alpha(D) = \alpha(\emptyset)$. Let $q(\widetilde{D}) = \{\vec{a}_1, \ldots, \vec{a}_n\}$. We replace the head variables in q with the corresponding constants from the answer $\vec{a}_1$. We denote the result by q'.

We start with the following observation. The query q' is a non-hierarchical Boolean CQ (recall that the definition of hierarchical queries considers only the existential variables, which are left intact in q'). We can break the query q' into connected components $q'_1, ..., q'_m$, such that $\mathsf{Vars}(q'_i) \cap \mathsf{Vars}(q'_j) = \emptyset$ for all $i \neq j$ (it may be the case that there is only one connected component). Since q' is not hierarchical, we have that q'_i is not hierarchical for at least one $i \in \{1, \ldots, m\}$. We assume, without loss of generality, that q'_1 is not hierarchical. Then, Theorem 4.1 implies that computing Shapley(D, q'_1, f) is FP$^{\#\mathrm{P}}$-complete. Therefore, we construct a reduction from the problem of computing Shapley(D, q'_1, f) to the problem of computing Shapley(D', α, f).

Let $x_1, \ldots, x_k$ be the head variables of q. If a tuple $\vec{a} = (v_1, \ldots, v_k)$ is in $q(D)$ for some database D, then for every connected component q_i of q, there is a homomorphism from q_i to D, such that each head variable x_j is mapped to the corresponding value v_j from $\vec{a}$. On the other hand, if this does not hold for at least one of the connected components, then $(v_1, \ldots, v_k)$ is not in $q(D)$.

Given an input database D to the first problem, we build an input database D' to our problem, as we explain next. As in the proof of Theorem 4.3, we assume, without loss of generality, that D contains only facts f such that there is a homomorphism from an atom of q'_1 to f. To construct D', we first add a subset of the facts of $\widetilde{D}$ to D'_{x} (recall that $\widetilde{D}$ is a minimal database satisfying $\alpha(\widetilde{D}) \neq \alpha(\emptyset)$). For each relation R that occurs in q'_i for $i = \{2, \ldots, m\}$, we copy all the facts from $R^{\widetilde{D}}$ to $R^{D'_{\mathsf{x}}}$. As explained above, for each answer $\vec{a}_i = (v_1, \ldots, v_k)$ in $q(\widetilde{D})$ and for each connected component q_i, there is a homomorphism from q_i to $\widetilde{D}$ such that each head variable x_j that appears in q_i is mapped to the value v_j; hence, the same holds for the database D' and every connected component in $\{q_2, \ldots, q_m\}$. Therefore, in order to have all the tuples $\{\vec{a}_1, \ldots, \vec{a}_n\}$ in $q(D')$, we only need to add additional facts to the relations that appear in q_1 to satisfy this connected component.

Now, let $x_{j_1}, \ldots, x_{j_r}$ be the head variables that appear in q_1. For each tuple $\vec{a}_i$ that does not agree with $\vec{a}_1$ on the values of these variables (i.e., the value of at least one x_{j_k} is different in $\vec{a}_1$ and $\vec{a}_i$), we generate a set of exogenous facts as follows. Assume that $\vec{a}_i$ uses the value v_{j_k} for the head variable x_{j_k}. We replace each variable x_{j_k} in q_1 with the value v_{j_k}. Then, we assign a new distinct value to each one of the existential variables of q_1. We then add the corresponding facts to D'_{x} (e.g., if q_1 now contains the atom $R(\mathsf{a}, \mathsf{b}, \mathsf{c})$, then we add the fact $R(\mathsf{a}, \mathsf{b}, \mathsf{c})$ to D'_{x}). At this point, it is rather straightforward that each $\vec{a}_i$ that does not agree with $\vec{a}_1$ on the values of the head variables of q_1 appears in $q(D')$; however, $\vec{a}_1$ and each tuple $\vec{a}_i$ that uses the same values as $\vec{a}_1$ for the head variables of q_1 are not yet in $q(D')$. Since we assumed that $\widetilde{D}$ is minimal, we know that $\alpha(D'_{\mathsf{x}}) = \alpha(\emptyset)$.

Next, we add all the facts of D to D'. Each fact of D_{x} is added to D'_{x}, and each fact of D_{n} is added to D'_{n}. We prove that the following holds:

$$\text{Shapley}(D, q'_1, f) = \frac{\text{Shapley}(D', \alpha, f)}{\alpha(\widetilde{D}) - \alpha(\emptyset)}$$

Let $A = \{\vec{a}_{k_1}, \ldots, \vec{a}_{k_t}\}$ be the set of answers that do not agree with $\vec{a}_1$ on the values of the head variables $x_{j_1}, \ldots, x_{j_r}$ of q_1. As explained above, we have that $A \subseteq q(D'_{\mathsf{x}})$. Moreover, since q'_1 was obtained from q_1 by replacing the head variables with the corresponding values from $\vec{a}_1$, and since we removed from D every fact that does not agree with q'_1 on those values, the only possible answer of q_1 on D is $(v_{j_1}, \ldots, v_{j_r})$ where v_{j_i} is the value in $\vec{a}_1$ corresponding to the head variable x_{j_i} of q_1. Since all the answers in $q(\widetilde{D}) \setminus A$ agree with $\vec{a}_1$ on the values of all these variables, we have that in each permutation σ of the endogenous facts in D', one of the following holds:

(1) $q(\sigma_f \cup D'_{\mathsf{x}}) = A$ and $q(\sigma_f \cup D'_{\mathsf{x}} \cup \{f\}) = A$,
(2) $q(\sigma_f \cup D'_{\mathsf{x}}) = A$ and $q(\sigma_f \cup D'_{\mathsf{x}} \cup \{f\}) = q(\widetilde{D})$.

Clearly, the contribution of each permutation that satisfies the first condition to the Shapley value of f is zero (as we assumed that $\alpha(A) = \alpha(\emptyset)$ for every $A \subset \{\vec{a}_1, \ldots, \vec{a}_n\}$), while the contribution of each permutation that satisfies the second condition to the Shapley value of f is $\alpha(\widetilde{D}) - \alpha(\emptyset)$ (as we assumed that $\alpha(\widetilde{D}) \neq \alpha(\emptyset)$).

Let X_f be a random variable that gets the value 1 if f adds the answer $\vec{a}_1$ to the result of q in the permutation and 0 otherwise. Due to the aforementioned observations and by the definition of the Shapley value, the following holds:

$$\text{Shapley}(D', \alpha, f) = (\alpha(\widetilde{D}) - \alpha(\emptyset)) \cdot E(X_f)$$

Note that D and D' contain the same endogenous facts. Moreover, the fact f adds the answer $\vec{a}_1$ to the result of q in a permutation σ of the endogenous facts in D' if and only if f changes the result of q'_1 from 0 to 1 in the same permutation σ of the endogenous facts of D. This holds since f changes the result of q'_1 in σ if and only if there exist a set of facts in $\sigma_f \cup D_{\mathsf{x}} \cup f$ (that contains f) that satisfies q'_1, which, as explained above, happens if and only if the same set of facts adds the answer $\vec{a}_1$ to the result of q in σ. Therefore, the Shapley value of a fact f in D is:

$$\text{Shapley}(D, q'_1, f) = E(X_f)$$

where X_f is the same random variable that we introduced above, and that concludes our proof. □

For instance, it follows from Theorem 4.8 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 4.9. Consider the numerical query α_3 from Example 2.3. Since q_4 is not hierarchical, Theorem 4.8 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 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. □

Interestingly, it turns out that Theorem 4.8 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 head variable

x_j with the value of x_j in $\vec{a}$. Hence, we have that $\mathsf{sum}\langle\varphi\rangle[q](D) = \sum_{\vec{a}\in q(D)} \left[\varphi(\vec{a}) \cdot q_{[\vec{x}\to\vec{a}]}(D)\right]$. The linearity of the Shapley value (stated as a fundamental property in Section 3) implies that:

$$\text{Shapley}(D, \mathsf{sum}\langle\varphi\rangle[q], f) = \sum_{\vec{a}\in q(D)} \varphi(\vec{a}) \cdot \text{Shapley}(D, q_{[\vec{x}\to\vec{a}]}, f)\,. \tag{4.2}$$

Then, from Theorem 4.1 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 4.1.

Corollary 4.10. *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 as input. In particular, $\text{Shapley}(D, \mathsf{count}[q], f)$ can be computed in polynomial time.*

Together with Theorem 4.8, 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 4.11. *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.*

Proof. Since q consists of a single atom, each fact adds at most one answer to the query result in each permutation. Let $\vec{a}_f \in q(D)$ be the answer corresponding to the fact f (i.e., $q(\{f\}) = \{\vec{a}_f\}$). Let σ be a permutation. First, if there exists an exogenous fact f' such that $\varphi(q(\{f'\})) > \varphi(\{\vec{a}_f\})$, then f will never affect the maximum value, and $\text{Shapley}(D, \mathsf{max}\langle\varphi\rangle[q], f) = 0$. Hence, from now on we assume that this is not the case. If $\vec{a}_f$ already appears in the query result before adding the fact f (that is, $\vec{a}_f \in q(\sigma_f)$), then clearly f does not affect the maximum value. If $\vec{a}_f$ is added to the query result only after adding f in the permutation (that is, $q(\sigma_f \cup \{f\} \cup D_\mathsf{x}) \setminus q(\sigma_f \cup D_\mathsf{x}) = \{\vec{a}_f\}$), then f only affects the maximum value if $\varphi(\{\vec{a}_f\}) > \max_{\vec{a}\in q(\sigma_f\cup D_\mathsf{x})} \varphi(\{\vec{a}\})$. In this case, it holds that

$$v(\sigma_f \cup \{f\} \cup D_\mathsf{x}) - v(\sigma_f \cup D_\mathsf{x}) = \varphi(\{\vec{a}_f\}) - \max_{\vec{a}\in q(\sigma_f\cup D_\mathsf{x})} \varphi(\{\vec{a}\})$$

Let $V = \{v_1, \ldots, v_m\}$ be the set of values associated with the answers in $q(D)$ (that is, V contains every value v_j such that $\varphi(\{\vec{a}\}) = v_j$ for some $\vec{a} \in q(D)$). Note that it may be the case that $\varphi(\{\vec{a}_1\}) = \varphi(\{\vec{a}_2\})$ for $\vec{a}_1 \neq \vec{a}_2$; hence, it holds that $|V| \leq |q(D)|$. For each value v_j we denote by $n^{<}_{v_j}$ the number of endogenous facts f' in the database that correspond to an answer $\vec{a}$ (i.e., $q(\{f'\}) = \{\vec{a}\}$) such that $\varphi(\{\vec{a}\}) < v_j$, and by $n^{=}_{v_j}$ the number of endogenous facts in the database that correspond to an answer $\vec{a}$ such that $\varphi(\{\vec{a}\}) = v_j$. We also denote by $n^{\leq}_{v_j}$ the number $n^{<}_{v_j} + n^{=}_{v_j}$.

Let $\{v_{i_1}, \ldots, v_{i_k}\}$ be the set of values in V such that

$$\left(\max_{\vec{a}\in q(D_\mathsf{x})} \varphi(\{\vec{a}\})\right) \leq v_{i_r} < \varphi(\{\vec{a}_f\})$$

Let us assume, without loss of generality, that $\max_{\vec{a}\in q(D_\mathsf{x})} \varphi(\{\vec{a}\}) = v_{i_1}$. For each $r \in \{2, \ldots, k\}$ and for each $t \in \{1, \ldots, n^{\leq}_{v_{i_r}}\}$, we compute the number of permutations σ in which

σ_f contains t facts, and it holds that $\max_{\vec{a}\in q(\sigma_f\cup D_\mathsf{x})}\varphi(\{\vec{a}\}) = v_{i_r}$:

$$P_r^t = t!\cdot(N-t-1)!\sum_{\ell=1}^{\min(n^{=}_{v_{i_r}},t)}\binom{n^{=}_{v_{i_r}}}{\ell}\binom{n^{<}_{v_{i_r}}}{t-\ell}$$

(That is, we choose at least one fact f' such that $\varphi(q(\{f'\})) = v_{i_r}$ and then we choose the rest of the facts among the facts f'' such that $\varphi(q(\{f''\})) < v_{i_r}$). We count the number of such permutations separately for v_{i_1}, because in this case, we do not have to choose at least one endogenous fact f' such that $\varphi(q(\{f'\})) = v_{i_1}$ (as this is already the maximum value on the exogenous facts). Hence, the number of permutations in this case is:

$$P_1^t = t!\cdot(N-t-1)!\binom{n^{\leq}_{v_{i_1}}}{t}$$

The contribution of each such permutation to the Shapley value of f is:

$$v(S\cup\{f\}) - v(S) = \varphi(\{\vec{a}_f\}) - v_{i_r}$$

Thus, the total contribution of the permutations σ such that $\max_{\vec{a}\in q(\sigma_f)}\varphi(\{\vec{a}\}) = v_{i_r}$ to the Shapley value of f is $(\varphi(\{\vec{a}_f\}) - v_{i_r})\sum_{t=1}^{n^{\leq}_{v_{i_r}}} P_r^t$.

Finally, the Shapley value of f is:

$$\text{Shapley}(D,\mathsf{max}\langle\varphi\rangle[q],f) = \frac{1}{|D_\mathsf{n}|!}\sum_{r=1}^{k}\left\{(\varphi(\{\vec{a}_f\}) - v_{i_r})\sum_{t=1}^{n^{\leq}_{v_{i_r}}} P_r^t\right\}$$

A similar computation works for $\mathsf{min}\langle\varphi\rangle[q]$. The main difference is that now we are looking to minimize the value; hence, instead of considering $\varphi(\{\vec{a}_f\}) - \max_{\vec{a}\in q(\sigma_f\cup D_\mathsf{x})}\varphi(\{\vec{a}\})$, we now use $\varphi(\{\vec{a}_f\}) - \min_{\vec{a}\in q(\sigma_f\cup D_\mathsf{x})}\varphi(\{\vec{a}\})$, and we only consider permutations where $\min_{\vec{a}\in q(\sigma_f\cup D_\mathsf{x})}\varphi(\{\vec{a}\}) > \varphi(\{\vec{a}_f\})$. □

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

The arguments in the proof of Proposition 4.11 heavily rely on the assumption that each fact adds at most one answer to the query result; hence, we can refer to *the* answer associated with a certain fact. Moreover, this answer is independent of the permutation. However, this assumption does not hold for general queries, where the addition of a fact can add multiple answers, and the added set of answers depends on the other facts in the permutation. Hence, the proof does not easily generalize to maximum and minimum over hierarchical queries consisting of more than one atom. We also cannot use here the linearity of expectation that was used to obtain a dichotomy for summation. Therefore, a complete classification of the complexity for general aggregate queries remain an open problem.

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* Boolean query computable in polynomial time, by simply taking the average value over $O(\log(1/\delta)/\epsilon^2)$ trials of the following experiment:

(1) Select a random permutation $(f_1, \ldots, f_n)$ over the set of all endogenous facts.
(2) Suppose that $f = f_i$, and let $D_{i-1} = D_{\mathsf{x}} \cup \{f_1, \ldots, f_{i-1}\}$. Return $q(D_{i-1} \cup \{f\}) - q(D_{i-1})$.

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 4.12. *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 4.13. *For every fixed Boolean CQ, the Shapley value has both an additive and a multiplicative FPRAS.*

Approximation for Aggregate Queries. The Chernoff-Hoeffding bound applies to the additive approximation of any function with a "bounded domain," that is, where the gap between the maximal and minimal value is polynomial in the size of the input. Hence, we immediately conclude that there is an additive FPRAS for count. We also get an additive FPRAS for sum, average, median (or any quantile), min and max in the case where the values are from a bounded domain.

What about multiplicative approximation for aggregate queries? Interestingly, Corollary 4.13 generalizes to a multiplicative FPRAS for summation (including counting) over CQs. By combining Corollary 4.13 with Equation (4.2), 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 Shapley$(D, \mathsf{count}[q], f)$.

Corollary 4.14. *For every fixed CQ q,* 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 4.12 continues to hold when considering *unions of conjunctive queries* (UCQs). Therefore, Corollaries 4.13 and 4.14 remain correct in the case where q is a UCQ.

The existence of additive and multiplicative approximations for other aggregate queries remains an open problem.

5. Related Measures

In this section, we discuss our work in comparison to some alternative measures for the responsibility of tuples to database queries.

Causal responsibility. Causality and causal responsibility [Pea09,Hal16] have been applied in data management, 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 $D \setminus \Gamma$ [MGMS10a]. 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 [MGMS10a]. Intuitively, facts with higher responsibility provide stronger explanations.[2]

Example 5.1. Consider the database of our running example, and the query q_1 from Example 2.2. The fact f_1^{a} is 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 5.2. Consider the database G and the query p_{ab} from Example 3.4. 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 3.4, the Shapley value gives a more intuitive degree of contribution of facts to the query result than causal responsibility. Actually, Example 3.4 was used in [SBSdB16] as a motivation to introduce an alternative to the notion of causal responsibility, that of *causal effect*.

[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) [BS17b, BS17a].

Causal effect. To quantify the contribution of a fact to the query result, Salimi et al. [SBSdB16] 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 (in particular, a Boolean query) is a difference of expected values [SBSdB16]:

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

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.

Example 5.3. Consider again the database of our running example, and the query q_1 from Example 2.2. We compute $\mathrm{CE}(D, q_1, f_1^\mathrm{a})$. It holds that: $\mathbb{E}(q_1 \mid \neg f_1^\mathrm{a}) = 0 \cdot P(q_1 = 0 \mid \neg f_1^\mathrm{a}) + 1 \cdot P(q_1 = 1 \mid \neg f_1^\mathrm{a}) = 1 - P(\neg f_2^\mathrm{a} \wedge \neg f_3^\mathrm{a} \wedge \neg f_4^\mathrm{a}) = \frac{7}{8}$. Similarly, we have that $\mathbb{E}(q_1 \mid f_1^\mathrm{a}) = P(q_1 = 1 \mid f_1^\mathrm{a}) = 1$. Then, $\mathrm{CE}(D, q_1, f_1^\mathrm{a}) = 1 - \frac{7}{8} = \frac{1}{8}$. Using similar computations we obtain that $\mathrm{CE}(D, q_1, f_2^\mathrm{a}) = \mathrm{CE}(D, q_1, f_3^\mathrm{a}) = \mathrm{CE}(D, q_1, f_4^\mathrm{a}) = \frac{1}{8}$.

For G and p_{ab} of Example 3.4, 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 3.3 and Example 3.4, 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 5.4. Consider the query α_1 of Example 2.3. 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 $\mathrm{CE}(D, \alpha_1, f_1^\mathrm{a}) = 20$. Similarly, it holds that $\mathrm{CE}(D, \alpha_1, f_2^\mathrm{a}) = \mathrm{CE}(D, \alpha_1, f_4^\mathrm{a}) = 1$, and $\mathrm{CE}(D, \alpha_1, f_3^\mathrm{a}) = 14$. □

Interestingly, the causal effect coincides with a well known wealth-distribution function in cooperative games, namely the *Banzhaf Power Index* (BPI) [Lee90, DS79, KL10]. This measure is defined similarly to the definition of the Shapley value in Equation (2.3), except that we replace the ratio $\frac{|B|!\cdot(|A|-|B|-1)!}{|A|!}$ with $\frac{1}{2^{|A|-1}}$.

Proposition 5.5. *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.

Proof. The following holds.

$$\begin{aligned}
\mathrm{CE}(D, \alpha, f) &= E(\alpha(D) \mid f) - E(\alpha(D) \mid \neg f) \\
&\stackrel{(*)}{=} \sum_{E \subseteq (D_\mathsf{n} \setminus \{f\})} \frac{1}{2^{|D_\mathsf{n}|-1}} \cdot \alpha(D_\mathsf{x} \cup E \cup \{f\}) - \sum_{E \subseteq (D_\mathsf{n} \setminus \{f\})} \frac{1}{2^{|D_\mathsf{n}|-1}} \cdot \alpha(D_\mathsf{x} \cup E) \\
&= \frac{1}{2^{|D_\mathsf{n}|-1}} \cdot \sum_{E \subseteq (D_\mathsf{n} \setminus \{f\})} \alpha(D_\mathsf{x} \cup E \cup \{f\}) - \alpha(D_\mathsf{x} \cup E)
\end{aligned}$$

The transition $(*)$ is correct since every endogenous fact in the probabilistic database has probability 0.5 and they are all independent; hence, all the possible worlds have the same probability $\frac{1}{2^{|D_n|-1}}$. (Recall that we condition on f being either present or absent from the database, and all exogenous facts are certain; thus, the probability of each possible world depends only on the facts in $D_n \setminus \{f\}$.) Then, for each $E \subseteq D_n \setminus \{f\}$, it holds that $\alpha(D_x \cup E \cup \{f\})$ is the value of the query on the possible world that contains all the exogenous facts, the fact f, and all the endogenous facts in E, but does not contain the endogenous facts in $D_n \setminus (E \cup \{f\})$. Hence, $\sum_{E \subseteq (D_n \setminus \{f\})} \frac{1}{2^{|D_n|-1}} \cdot \alpha(D_x \cup E \cup \{f\})$ is by definition the expected value $E(\alpha(D) \mid f)$. Similarly, $\sum_{E \subseteq (D_n \setminus \{f\})} \frac{1}{2^{|D_n|-1}} \cdot \alpha(D_x \cup E)$ is the expected value $E(\alpha(D) \mid \neg f)$, and that concludes our proof. □

In the next section we will discuss in more detail the complexity of the causal effect.

SHAP score. One of the instantiations of the Shapley value is the *SHAP score* that has been used in the context of machine learning for explaining the prediction of a model [LL17]. This score could be applied to the attribution of responsibility to tuples in query answering, and it would give a measure that is similar in spirit, yet technically different, from our application of the Shapley value. Both apply the Shapley value in a cooperative game where the players are the endogenous facts. The difference is in the definition of the cooperative game. In our case, the players of a coalition, as a subinstance of the database, occur in the database, while the others are excluded from the computation of the wealth function, which is the answer of the query on the resulting subinstance. In the case of the SHAP score as applied to our setting, we could view the database as a tuple-independent probabilistic database where the facts of the coalition are deterministic (probability one) and the others are probabilistic (say with the probability $\frac{1}{2}$), and the wealth function is the *expectation* of the query answer on the resulting probabilistic database.

The complexity of the SHAP score (in a more abstract setting than query answering) has been studied by Van den Broeck et al. [VdBLSS21] and by Arenas et al. [ABBM21b, ABBM21a].[3] Immediate algorithmic approaches that one can derive to compute the SHAP score for Boolean CQs and UCQs are *(a)* to reduce the problem to probabilistic query answering [VdBLSS21] *and (b)* to compile the *provenance* of the query into a *deterministic and decomposable circuit* and apply to the circuit the Shapley computation of Arenas et al. [ABBM21b, ABBM21a]. Yet, albeit the similarity between our Shapley value and the SHAP score, they are different enough that we do not see an immediate way of directly translating results (e.g., via simple reductions) from one to the other. It is sensible, though, that we could apply similar techniques, namely reduction to/from probabilistic query answering and knowledge compilation, to derive our results and perhaps more general results. We leave this investigation to future research.

5.1. **The Complexity of the Causal-Effect Measure (and Banzhaf Power Index).** We now show that the complexity results obtained in this work for the exact computation of the Shapley value also apply to the causal effect (and BPI). These results are, in fact, easier to obtain, via a connection to probabilistic databases [SORK11]. The extension of the approximation results will be discussed later.

[3]The referenced work has been published later than the conference version of this article [LBKS20].

Proposition 5.6. *Theorem 4.1, Theorem 4.8, Corollary 4.10, and Proposition 4.11 hold for the Banzhaf Power Index.*

Proof. We start with Theorem 4.1—the dichotomy for Boolean self-join-free CQs. The positive side of the dichotomy (polynomial-time computation for hierarchical queries) is obtained via a reduction to query evaluation over probabilistic databases. Recall that Dalvi and Suciu [DS13] showed that this problem is solvable in polynomial time for hierarchical Boolean CQs without self-joins. For a Boolean query, we have that $\mathrm{CE}(D, \alpha, f)$ is the probability that q is true when f is present in the database minus the probability that q is true when f is absent from the database. The first probability can be computed via a reduction to query evaluation over probabilistic databases by defining the probability of f to be 1 (and leaving the probabilities of the rest of the facts intact), and the second probability can be computed by a similar reduction after removing f from the database altogether.

As for the negative side of the dichotomy (hardness of computation for non-hierarchical queries), we again use the result of Dalvi and Suciu [DS13]. This time, we construct a reduction from their problem to our problem for the query $\mathsf{q}_{\mathsf{RST}}$. They showed that query evaluation over probabilistic databases is $\mathrm{FP}^{\#\mathrm{P}}$-complete for $\mathsf{q}_{\mathsf{RST}}$, even when the probabilities of all the facts in the database are either 1 or 0.5. Hence, given an input database D to their problem, we construct an input database D' to our problem by adding all the facts of D, as well as two facts $R(\mathtt{a})$ and $T(\mathtt{b})$ with probability 1 and a fact $S(\mathtt{a}, \mathtt{b})$ with probability 0.5, where $\mathtt{a}$ and $\mathtt{b}$ are two fresh constants that do not appear in D. Clearly, if $S(\mathtt{a}, \mathtt{b})$ is present in the database, then the probability of q being true is 1. On the other hand, if $S(\mathtt{a}, \mathtt{b})$ is absent, then the probability of q being true is exactly the probability of q being true on the original D. Therefore, to obtain this probability, we simply need to compute the value $1 - \mathrm{CE}(D', q, S(\mathtt{a}, \mathtt{b}))$. To prove hardness for any other non-hierarchical query we again use the result of Lemma 4.7 that clearly also applies to the BPI. An important observation here is that in the reduction of Lemma 4.7 for the BPI we preserve the probabilities of the facts (0.5 for endogenous facts and 1 for exogenous facts).

Since the Shapley value and the BPI entail the same complexity for Boolean CQs, it is rather straightforward that Theorem 4.8 and Corollary 4.10 hold for the BPI as well, and the proof is almost identical (with the main difference being the fact that for the BPI we consider subsets of facts rather than permutations); hence, we do not give the proofs again here. Note that in the proof of the tractability side of Corollary 4.10 we do not use any special property of the Shapley value, but rather the linearity of expectation, and so this proof applies to the BPI as well. Finally, the proof of Proposition 4.11 is also very similar for the BPI, where, again, instead of counting permutation, we count subsets of facts. □

Note that the dichotomy of Dalvi and Suciu [DS13] for query evaluation over probabilistic databases also applies to queries with self-joins. The tractable cases of their dichotomy immediately translate to the causal effect measure, as in the case of self-join-free queries. However, the probabilities used in the hardness proofs for such queries are not necessarily 0.5 and 1; hence, unlike the case of self-join-free queries, the hardness results of Dalvi and Suciu [DS13] for queries with self-joins cannot be directly translated to complexity results for the causal-effect measure that, as aforementioned, assumes a probability of 0.5 for endogenous facts and a probability of 1 for exogenous facts.

In a recent work [KS21], Kenig and Suciu showed that the hard cases of the dichotomy for queries with self-joins remain hard even when all the probabilities are 0.5 or 1. However, even this stronger result does not immediately translate to a dichotomy for the causal effect

measure. In particular, for self-join-free queries, we could provide an alternative proof for non-hierarchical queries, that does not go through the query q_{RST}; however, this proof is slightly more involved. To use the same idea that we use in the proof of hardness for q_{RST} for other non-hierarchical queries q, we first need to identify a connected component q' of q that is non-hierarchical, and then construct the reduction from probabilistic query answering for the query q'. This method cannot be used in general for queries with self-joins, as a homomorphism from the q to D might map several atoms of q to the same fact of D. We could, however, use this technique to obtain hardness for some queries with self-joins. In particular, we can prove hardness for connected queries that do not use constants. A similar idea has been used in [RKL20] in the proof of Theorem 5.1. A complete classification of the complexity for queries with self-join remains an open problem.

Finally, we discuss the approximate computation of the causal-effect measure. Similarly to case of the Shapley value, we can easily obtain an additive approximation via Monte Carlo Sampling. Using this method we can obtain an additive approximation of $E(\alpha(D) \mid f)$, as well as an additive approximation of $E(\alpha(D) \mid \neg f)$. Clearly, by subtracting the second value from the first one we obtain an additive approximation of $E(\alpha(D) \mid f) - E(\alpha(D) \mid \neg f)$ (with approximation factor 2ϵ). The story is different for multiplicative approximation, as we cannot obtain a multiplicative approximation of $x - y$ from such approximation of x and y, unless the property of Proposition 4.12 holds. The following example shows that this is not the case for the causal-effect measure.

Example 5.7. Consider the query $\mathsf{q}_{\mathsf{RST}}$ and a database consisting of $n+1$ triplets of facts: $R(a_i), S(a_i, b_i), T(b_i)$ for $i \in \{1, \ldots, n+1\}$, where all the facts in R and T are exogenous and the facts of S are endogenous. Let $f = S(a_1, b_1)$. Clearly, we have that $q(D_{\mathsf{x}} \cup E \cup \{f\}) - q(D_{\mathsf{x}} \cup E) = 1$ only if none of the other endogenous facts appears in E (i.e., $E = \emptyset$). Therefore, we have that $\mathrm{CE}(D, q, f) = \frac{1}{2^n}$. □

Note that the above example does not preclude the existence of a multiplicative approximation for the causal-effect measure (and BPI). This is left open for future investigation.

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. Finally, we showed that all of our complexity results for the Shapley value, except for the existence of a multplicative approximation, also hold for the causal-effect measure (and Banzhaf power index).

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 4.8). 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. An additional question that is still open is whether there exists a multiplicative approximation for the causal-effect measure. 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 [GLS15].

Acknowledgments

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. Leopoldo Bertossi is a member of RelationalAI's Academic Network, and has been partially funded by the ANID - Millennium Science Initiative Program - Code ICN17-002.

References

[ABBM21a] Marcelo Arenas, Pablo Barceló, Leopoldo Bertossi, and Mikaël Monet. On the complexity of shap-score-based explanations: Tractability via knowledge compilation and non-approximability results. *arXiv preprint*, 2021.

[ABBM21b] Marcelo Arenas, Pablo Barceló, Leopoldo Bertossi, and Mikaël Monet. The tractability of SHAP-score-based explanations over deterministic and decomposable boolean circuits. In *Proceedings of AAAI*, 2021.

[AdK14] Haris Aziz and Bart de Keijzer. Shapley meets Shapley. In *STACS*, pages 99–111, 2014.

[AM03] 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.

[Bac02] Roland Bacher. Determinants of matrices related to the pascal triangle. *Journal de Théorie des Nombres de Bordeaux*, 14, 01 2002.

[BS17a] 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.

[BS17b] 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.

[CNS07] Sara Cohen, Werner Nutt, and Yehoshua Sagiv. Deciding equivalences among conjunctive aggregate queries. *J. ACM*, 54(2):5, 2007.

[CS04] 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.

[DRS09] Nilesh N. Dalvi, Christopher Ré, and Dan Suciu. Probabilistic databases: diamonds in the dirt. *Commun. ACM*, 52(7):86–94, 2009.

[DS79] Pradeep Dubey and Lloyd S. Shapley. Mathematical properties of the Banzhaf power index. *Mathematics of Operations Research*, 4(2):99–131, 1979.

[DS04] Nilesh N. Dalvi and Dan Suciu. Efficient query evaluation on probabilistic databases. In *VLDB*, pages 864–875. Morgan Kaufmann, 2004.

[DS12] 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.

[DS13] Nilesh Dalvi and Dan Suciu. The dichotomy of probabilistic inference for unions of conjunctive queries. *J. ACM*, 59(6), January 2013.

[GH06] John Grant and Anthony Hunter. Measuring inconsistency in knowledgebases. *J. Intell. Inf. Syst.*, 27(2):159–184, 2006.

[GLS15] Gianluigi Greco, Francesco Lupia, and Francesco Scarcello. Structural tractability of Shapley and Banzhaf values in allocation games. In *IJCAI*, pages 547–553, 2015.

[Gul89] Faruk Gul. Bargaining foundations of Shapley value. *Econometrica: Journal of the Econometric Society*, pages 81–95, 1989.

[Hal16] Joseph Y. Halpern. *Actual Causality*. The MIT Press, 2016.

[HK10] Anthony Hunter and Sébastien Konieczny. On the measure of conflicts: Shapley inconsistency values. *Artif. Intell.*, 174(14):1007–1026, 2010.

[HP01] Joseph Y. Halpern and Judea Pearl. Causes and explanations: A structural-model approach: Part 1: Causes. In *UAI*, pages 194–202, 2001.

[KL10] Werner Kirsch and Jessica Langner. Power indices and minimal winning coalitions. *Social Choice and Welfare*, 34(1):33–46, Jan 2010.

[KS21] Batya Kenig and Dan Suciu. A dichotomy for the generalized model counting problem for unions of conjunctive queries. In *PODS*, pages 312–324. ACM, 2021.

[LBKS20] Ester Livshits, Leopoldo E. Bertossi, Benny Kimelfeld, and Moshe Sebag. The shapley value of tuples in query answering. In *ICDT*, volume 155 of *LIPIcs*, pages 20:1–20:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[LEC+20] Scott M. Lundberg, Gabriel Erion, Hugh Chen, Alex DeGrave, Jordan M. Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee. From local explanations to global understanding with explainable ai for trees. *Nature Machine Intelligence*, 2(1):56–67, Jan 2020.

[Lee90] Dennis Leech. Power indices and probabilistic voting assumptions. *Public Choice*, 66(3):293–299, Sep 1990.

[LL17] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, pages 4765–4774. Curran Associates, Inc., 2017.

[LZS15] 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.

[MCL+10] 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.

[MGH+10] 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.

[MGMS10a] 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.

[MGMS10b] 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.

[Nen03] Tatiana Nenova. The value of corporate voting rights and control: A cross-country analysis. *Journal of financial economics*, 68(3):325–351, 2003.

[NN11] 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.

[Pea09] Judea Pearl. *Causality: Models, Reasoning and Inference*. Cambridge University Press, New York, NY, USA, 2nd edition, 2009.

[PZ03] 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.

[RKL20] Alon Reshef, Benny Kimelfeld, and Ester Livshits. The impact of negation on the complexity of the shapley value in conjunctive queries. In *PODS*, pages 285–297. ACM, 2020.

[Rot88] Alvin E Roth. *The Shapley value: essays in honor of Lloyd S. Shapley*. Cambridge University Press, 1988.

[SBSdB16] Babak Salimi, Leopoldo E. Bertossi, Dan Suciu, and Guy Van den Broeck. Quantifying causal effects on query answering in databases. In *TAPP*, 2016.

[Sha53] 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.

[SORK11] Dan Suciu, Dan Olteanu, Christopher Ré, and Christoph Koch. *Probabilistic Databases*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011.

[SS54] 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.

[Tod91] Seinosuke Toda. PP is as hard as the polynomial-time hierarchy. *SIAM J. Comput.*, 20(5):865–877, 1991.

[VdBLSS21] Guy Van den Broeck, Anton Lykov, Maximilian Schleich, and Dan Suciu. On the tractability of shap explanations. In *Proceedings of AAAI*, 2021.

[YVCB18] Bruno Yun, Srdjan Vesic, Madalina Croitoru, and Pierre Bisquert. Inconsistency measures for repair semantics in OBDA. In *IJCAI*, pages 1977–1983. ijcai.org, 2018.