Tractability Frontiers of the Shapley Value for Aggregate Conjunctive Queries

CHRISTOPH STANDKE*, RWTH Aachen University, Germany
BENNY KIMELFELD*, Technion - Israel Institute of Technology, Israel and RelationalAI, USA

In recent years, the Shapley value has emerged as a general game-theoretic measure for assessing the contribution of a tuple to the result of a database query. We study the complexity of calculating the Shapley value of a tuple for an aggregate conjunctive query, which applies an aggregation function to the result of a conjunctive query (CQ) based on a value function that assigns a number to each query answer. Prior work by Livshits, Bertossi, Kimelfeld, and Sebag (2020) established that this task is #P-hard for every nontrivial aggregation function when the query is non-hierarchical with respect to its existential variables, assuming the absence of self-joins. They further showed that this condition precisely characterizes the class of intractable CQs when the aggregate function is sum or count. In addition, they posed as open problems the complexity of other common aggregate functions such as min, max, count-distinct, average, and quantile (including median).

Towards the resolution of these problems, we identify for each aggregate function a class of hierarchical CQs where the Shapley value is tractable with every value function, as long as it is local (i.e., determined by the tuples of one relation). We further show that each such class is maximal: for every CQ outside of this class, there is a local (easy-to-compute) value function that makes the Shapley value #P-hard. Interestingly, our results reveal that each aggregate function corresponds to a different generalization of the class of hierarchical CQs from Boolean to non-Boolean queries. In particular, max, min, and count-distinct match the class of CQs that are all-hierarchical (i.e., hierarchical with respect to all variables), and average and quantile match the narrower class of q-hierarchical CQs introduced by Berkholz, Keppeler, and Schweikardt (2017) in the context of the fine-grained complexity of query answering. Finally, we show an example of an aggregate function—"has-duplicates"—that gives rise to a new variation of hierarchy that is more restricted than q-hierarchy.

CCS Concepts: • **Information systems** → **Data provenance**.

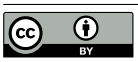
Additional Key Words and Phrases: Shapley value, aggregate queries

ACM Reference Format:
Christoph Standke and Benny Kimelfeld. 2025. Tractability Frontiers of the Shapley Value for Aggregate Conjunctive Queries. *Proc. ACM Manag. Data* 3, 5 (PODS), Article 284 (November 2025), 21 pages. https://doi.org/10.1145/3767720

1 Introduction

Significant research efforts have been dedicated in recent decades to devising concepts, formalisms, and methodologies for explaining the outcomes of algorithms in data science, particularly in machine learning and databases [9, 16, 25, 45, 55]. Within this broader context, various approaches have been proposed and studied for quantifying the contribution of a database tuple to the result of a query [35, 40, 41, 50]. Our work focuses on the game-theoretic approach of the *Shapley value* [51], a well-established function for distributing wealth in cooperative games, underpinned

Authors' Contact Information: Christoph Standke, standke@cs.rwth-aachen.de, RWTH Aachen University, Aachen, Germany; Benny Kimelfeld, bennyk@cs.technion.ac.il, Technion - Israel Institute of Technology, Haifa, Israel and RelationalAI, Berkeley, USA.



ACM 2836-6573/2025/11-ART284
https://doi.org/10.1145/3767720

by various theoretical justifications [48, 51]. The Shapley value has found broad application across diverse fields such as economics, law, environmental science, and network analysis, as well as in data-centric paradigms including knowledge representation [29], machine learning [37, 38, 49], and databases [10, 12, 20, 34, 36, 39]. Specifically, prior work by Livshits, Bertossi, Kimelfeld, and Sebag [35] initiated the study of the complexity of calculating the Shapley value of a database tuple's contribution to the result of a conjunctive query (CQ). This problem has subsequently been extended to related game-theoretic measures, such as the Banzhaf Power Index [1] and, more generally, *Shapley-like* scores [33].

In this manuscript, we investigate the complexity of computing the Shapley value for database tuples contributing to the numeric result of an *aggregate conjunctive query* (AggCQ). An AggCQ is defined as $\alpha \circ \tau \circ Q$, evaluated by first applying a conjunctive query Q, then transforming each resulting tuple into a numerical value via a *value function* τ, and finally aggregating this bag of numbers into a single numerical result using an aggregate function α. Throughout this section, we implicitly assume that CQs are without self-joins. Furthermore, our primary focus is on *localized* value functions τ, which are determined by the attributes of a single relation. We defer a discussion on non-localized value functions to Section 7.3. Moreover, we always assume that τ is computable in polynomial time.

The complexity analysis by Livshits et al. [35], along with subsequent work [12, 47], primarily focused on the Shapley value for Boolean CQs and, more generally, for a database tuple's contribution to an answer tuple in the output. We refer to this specific task as *membership*. They established that the tractability criterion for CQs in this context is the property of being ∃-*hierarchical*, meaning hierarchical with respect to the existential variables. (Recall that "hierarchical" means that for every two variables, their corresponding sets of query atoms are either disjoint or one contains the other [17].) This criterion precisely mirrors that for CQ evaluation over tuple-independent probabilistic databases [17, 18]. The similarity in complexities is not coincidental, as computational hardness for both problems boils down to *model counting*. Bienvenu, Figueira, and Lafourcade [12] and Kara, Olteanu, and Suciu [32] further explored the in-depth connection between the Shapley value and model counting. Livshits et al. [35] initially motivated the study of membership, in part, due to the potential for extending this task to aggregate queries. They demonstrated this potential with two key findings: on the negative side, they showed that being ∃-hierarchical is necessary for the tractability of the Shapley value for every non-trivial aggregate function (under conventional complexity assumptions); on the positive side, they proved it is also sufficient for the aggregate functions sum and count.

The algorithm of sum (and count) over ∃-hierarchical CQs leverages the linearity of expectation. Livshits et al. [35] posed as open problems the cases of other common aggregate functions that do not seem to benefit from the linearity of expectation, specifically the functions min, max, average, and quantile. As a preliminary step, they gave a polynomial-time algorithm for min and max in the special case of a single-atom CQ. Recent work [2] that was done independently of this paper extended the algorithm for min and max to the class of *all-hierarchical* CQs, that is, CQs hierarchical w.r.t. *all* variables (and not just the existential ones as in ∃-hierarchical). Yet, prior work left unanswered the question of tractability frontiers for other common aggregate functions and, in particular, whether the tractability frontier of sum and count also applies to these.

In this manuscript, we answer this question (negatively) for min, max, average, and quantile, as well as count-distinct (and also "has-duplicates" that we refer to later on) in the case of AggCQs and with localized value functions. In particular, we identify for each aggregate function a class of hierarchical CQs where the Shapley value is tractable with *every* localized value function. Importantly, we show that each such class is *the maximal possible*, forming a frontier, in the

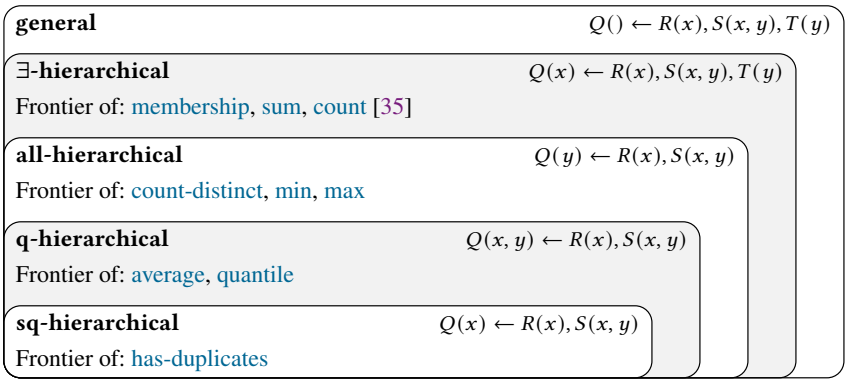


Fig. 1. Containment among classes CQs without self-joins. For each class, the box states the aggregate functions where the class captures precisely the tractable CQs without self-joins for Shapley computation.

following sense: for every CQ outside of this class, there is a local value function where the Shapley value of a fact is #P-hard. We elaborate on our results in the remainder of this section.

Contributions. We begin by showing (in Proposition 3.2) a property of aggregate functions that, when held, makes the Shapley value intractable for every AggCQ, unless it is all-hierarchical. This property holds for all aforementioned aggregate functions, except for the functions sum and count; hence, we establish a negative answer to the above open question. In the case of min, max, and count-distinct, the property of being all-hierarchical is not only necessary for tractability, but also sufficient. Hence, we establish a full classification for these three functions.

Nevertheless, we show that being all-hierarchical is insufficient for the aggregate functions *average* and *quantile*: there are #P-hard average and quantile AggCQs with all-hierarchical CQs. We establish that the tractable class of CQs for these two aggregate functions is precisely that of the *q-hierarchical* CQs, introduced by Berkholz, Keppeler and Schweikardt [7] in the context of structure maintenance for answering CQs with constant-time delay. To the best of our knowledge, the property of q-hierarchical has been shown so far to characterize tractable fragments only in the scope of fine-grained complexity (separating different classes of polynomial time, e.g., [7, 8, 30, 42, 54]). Hence, this work is the first to show that q-hierarchical also characterizes tractability in a natural context of coarse-grained complexity (separating polynomial time from exponential time). The hardness proofs are via matrix inversion, as done previously for proving the hardness of Shapley-value calculation [5, 32, 33, 35, 36]; notably, in the case of average, our matrix construction uses the Kronecker product of the Hilbert matrix and the Hankel matrix with factorial entries.

Recall that we started with the class of ∃-hierarchical CQs for the tractability of sum and count, restricted it to the all-hierarchical CQs for min, max and count distinct, and needed to further restrict it to the q-hierarchical CQs for average and quantile. Is the class of q-hierarchical CQs the "ultimate extension" of a single relation into multi-way joins? More precisely, is it the case that, for every aggregate function α (possibly with certain natural assumptions), if the Shapley value can be computed in polynomial time over a single relation, then it can also be computed in polynomial time on every q-hierarchical CQ? The answer turns out to be false.

We identify a natural aggregate function where the Shapley value can be computed in polynomial time over a single relation, yet is intractable on certain q-hierarchical CQs. This is the aggregate

function "has-duplicates" that tests whether its input list of values includes at least one duplicate.[1] Moreover, we delineate the precise class of CQs where this aggregate function is tractable—this class is defined by restricting the class of q-hierarchical CQs so that no free variable can have a set of atoms that is strictly contained in that of any other atom. We call CQs of this class *strongly q-hierarchical*, or *sq-hierarchical* for short. This class, like all other classes mentioned earlier, coincides with the class of hierarchical CQs in the Boolean case. Whether this restricted class is the "ultimate extension" in the sense described above remains an open problem for future investigation.

Figure 1 summarizes the results of this manuscript. Each box represents a class of CQs without self-joins, and their layout reflects containment among the classes, with "general" being the most general and "sq-hierarchical" being the most restrictive. In each box, we list the aggregate functions α for which the corresponding CQ class is a *tractability frontier*; that is, it is the precise class of CQs without self-joins for which AggCQs with α admit a polynomial-time computation of the Shapley value. For each box, we also provide an example of a CQ that belongs to the class but not to the more restrictive class that directly contains it. The class of $\exists$-hierarchical CQs consists of those CQs for which the Shapley value of a database tuple, with respect to a given answer tuple in the result of the CQ, can be computed in polynomial time (and we refer to this result as *membership* in Figure 1); this class is also the tractability frontier for the sum and count aggregate functions [35]. For count-distinct, min, and max, the tractability frontier is the more restricted class of all-hierarchical CQs. Strictly harder are the cases of average and quantile (for every threshold), where the tractability frontier is the class of q-hierarchical CQs. The aggregate function has-duplicates is the hardest we consider, as its tractability frontier is the most restricted—the class of sq-hierarchical CQs.

The intuition behind the separation is as follows. The $\exists$-hierarchical property enables the decomposition of the database into smaller sub-databases, each corresponding to a distinct answer. These sub-databases may overlap, which is acceptable for the sum and count aggregates due to the linearity of expectation [35]. However, this is insufficient for other aggregate functions. In contrast, the all-hierarchical property allows us to apply the standard algorithm for hierarchical CQs [17, 18] that (conceptually) operates by repeatedly partitioning the database into disjoint subsets. This approach is suitable for min and max (though a different argument is used for count-distinct). Nonetheless, these disjoint subsets can still yield overlapping sets of answers, which poses a problem to the Shapley computation for the average and quantile aggregates. The q-hierarchical property addresses this by ensuring that the answer sets are also disjoint. For the aggregation has-duplicates, even this property is insufficient, since the Shapley computation requires not only disjoint answer sets but also disjoint sets of *values* extracted from the answers for aggregation. This stricter requirement can be accommodated when using sq-hierarchical CQs.

Organization. The rest of the manuscript is organized as follows. We first introduce the general framework, including the problem definition (Section 2). We then establish general machinery that applies to all (or most) of the aggregate functions we study in the manuscript, including hardness under a general condition and an algorithmic template for hierarchical CQs (Section 3). We study the aggregate functions count-distinct, min, and max (Section 4), the aggregate functions average and quantile (Section 5), and then the function has-duplicates (Section 6). Finally, we address the standing challenge of extending the classification results of this paper to incorporate the variability of the value function (Section 7) and conclude (Section 8). Due to space constraints, some of the proofs are omitted from this manuscript and can be found in the arXiv version of this paper [52].

[1]This aggregate operation is non-conventional in traditional database systems since it can be implemented easily via grouping and the count aggregation; yet, it is supported in the pandas library via the property has_duplicates of the Index object of a DataFrame.

2 Preliminaries

We begin by introducing the formal framework of the manuscript, including the problem definition.

Sets and bags. Let X be a set and k be a number. A *k-subset* of X is a set $Y \subseteq X$ with $|Y| = k$. By $\binom{X}{k}$ we denote the set of all k-subsets of X. Recall that a *bag* is a pair (X, μ) where X is a set and $\mu : X \to \mathbb{N}$ is a function that maps every element $x \in X$ to a *multiplicity*. For a bag $B = (X, \mu)$ we write $x \in B$ for $\mu(x) > 0$. The size of a bag $B = (X, \mu)$ is $|B| = \sum_{x \in X} \mu(x)$. We denote by $\mathcal{B}_{\mathsf{fin}}(X)$ the set of all finite bags over X. If $f : X \to Y$ is a function, then we use the conventional notation of $\{\!\{f(x) \mid x \in X\}\!\}$ to denote the bag (Y, μ) where $\mu(y) = |\{x \in X \mid f(x) = y\}|$, that is, the bag that is obtained from X by replacing each element x with the element $f(x)$ while keeping duplicates.

The Shapley value. A *cooperative game* is a pair (P, ν) where P is a set of players and $\nu : 2^P \to \mathbb{R}$ is a *utility function* that associates every coalition $C \subseteq P$ with a value $\nu(C)$, so that $\nu(\emptyset) = 0$.

Consider a situation when we gather the players of P by selecting players iteratively uniformly at random without replacement, starting with the empty set. The Shapley value of a player $p \in P$ is the expected increase in utility when adding p [48, 51]. Using the law of conditional expectation via conditioning over the set of players chosen before p, it can be computed by the following formula.

$$\mathrm{Shapley}(p, \nu) = \sum_{C \subseteq P \setminus \{p\}} q_{|C|} \cdot (\nu(C \cup \{p\}) - \nu(C)) \tag{1}$$

where $q_{|C|} := \frac{|C|!(|P|-|C|-1)!}{|P|!}$ is the probability that the iterative process selects C first, then p, and then the remaining $|P| - |C| - 1$ players.

Relational databases. A *relation schema* has the form R/k, where R is a *relation name* and $k \in \mathbb{N}$ is an *arity*. We assume an infinite domain Const of *constants* that occur in database tuples. A *relation* over the relation schema R/k is a finite subset of Const^k. A *database schema* (or just *schema* for short) is a finite set $\mathcal{S}$ of relation schemas with distinct names. We denote by $ar(R)$ the arity k associated with the relation name R (assuming that $\mathcal{S}$ is known from the context). A *database* D over the schema $\mathcal{S}$ assigns to each relation schema R/k a relation over R/k, and we denote this relation by R^D. A *fact* over $\mathcal{S}$ is an expression of the form $R(a_1, \ldots, a_k)$ where $R/k \in \mathcal{S}$ and each a_i is in Const. We identify a database D over a schema $\mathcal{S}$ as the set of its facts over $\mathcal{S}$. For example, $D_1 \subseteq D_2$ means that each relation of D_2 contains the corresponding relation of D_1.

Conjunctive queries. Let $\mathcal{S}$ be a schema. A *relational query* Q is associated with an arity, denoted $ar(Q)$, and it is abstractly a function that maps a given database D over $\mathcal{S}$ into a finite k-ary relation $Q(D) \subseteq \mathsf{Const}^k$, where $k = ar(Q)$.

We will focus on the special case where the query is a *conjunctive query* (CQ), which is an expression of the form

$$Q(\vec{x}) \leftarrow R_1(\vec{z}_1), \ldots, R_q(\vec{z}_q)$$

where $\vec{x}$ is a sequence of variables, each R_j is the name of a relation in $\mathcal{S}$, and each $\vec{z}_j$ is a sequence of length $ar(R_j)$ consisting of variables and constants. We denote by $vars(Q)$ the set of all variables of Q (i.e., those that occur in either $\vec{x}$ or some $\vec{z}_j$), by $vars_{\mathsf{F}}(Q)$ the set of *free* variables of Q, that is, those in $\vec{x}$, and by $vars_{\exists}(Q)$ the set of *existential* variables of Q, that is, those in $vars(Q)$ that are not in $\vec{x}$. Hence, $vars(Q) = vars_{\mathsf{F}}(Q) \cup vars_{\exists}(Q)$. In addition, we denote by $\mathrm{atoms}(Q)$ the set of atomic formulas $R_j(\vec{z}_j)$ of Q, and by $\mathrm{atoms}(Q, x)$ the set of atoms of Q where the variable x occurs. By a *self-join* we refer to a repeated relation symbol in the body of Q. Throughout the manuscript, we always assume that the involved CQs have no self-joins.

We adopt the usual semantics of CQs, defined as follows. A *homomorphism* h from the CQ Q to a database D is a function $h : vars(Q) \to \mathsf{Const}$ that maps every variable y of Q to a constant $h(y)$,

so that for every atom $R(\vec{z})$ of Q, the fact $R(h(\vec{z}))$ is in D; here, $h(\vec{z})$ denotes the sequence obtained from $\vec{z}$ by replacing every variable y with the constant $h(y)$. We denote by $Hom(Q, D)$ the set of all homomorphisms from Q to D. Then $Q(D) := \{h(\vec{x}) \mid h \in Hom(Q, D)\}$.

Hierarchical CQs. We refine the well-known concept of a *hierarchical* CQ, as follows. Let Q be a CQ. We say that Q is *hierarchical* with respect to (w.r.t.) a set V of variables if, for all variables x and y in V, it holds that (a) $\text{atoms}(Q, x) \subseteq \text{atoms}(Q, y)$, or (b) $\text{atoms}(Q, y) \subseteq \text{atoms}(Q, x)$, or (c) $\text{atoms}(Q, x)$ and $\text{atoms}(Q, y)$ are disjoint. We say that Q is $\exists$-*hierarchical* if Q is hierarchical w.r.t. the set $vars_{\exists}(Q)$ of existential variables, and that Q is *all-hierarchical* if Q is hierarchical w.r.t. the entire set $vars(Q)$ of all variables of Q. The CQ Q is *q-hierarchical* if it is all-hierarchical and for all variables x and y, if $\text{atoms}(Q, y) \subsetneq \text{atoms}(Q, x)$ and y is free, then x is free as well; in other words, there are no existential x and free y such that $\text{atoms}(Q, y) \subsetneq \text{atoms}(Q, x)$.

Note that an all-hierarchical CQ is also $\exists$-hierarchical, but not necessarily vice versa, and that every q-hierarchical CQ is all-hierarchical. Hence, we have the following entailment chain:

$$\text{q-hierarchical} \rightarrow \text{ all-hierarchical} \rightarrow \text{ } \exists\text{-hierarchical}$$

Figure 1 includes examples of CQs of the different classes. For every class (represented by a box), its CQ is not in any class contained in it.

Remark 2.1. All three classes of hierarchical CQs coincide when the CQ is Boolean—the distinction is meaningful only in the presence of free variables, which are essential for the aggregation. The concept of a hierarchical CQ was originally coined by Dalvi and Suciu [17]. The relaxation to $\exists$-hierarchical was adopted later on (under different terminology) by Fink and Olteanu [23]. The restriction to q-hierarchical CQs was introduced soon after by Berkholz, Keppeler and Schweikardt [7] in the context of fine-grained complexity of query maintenance through database updates. ◇

Aggregate queries. Let $\mathcal{S}$ be a schema. In this work, an *aggregate query* A consists of three components: *(1)* a relational query Q; *(2)* a *value function* $\tau : \text{Const}^k \rightarrow \mathbb{R}$ where $k = ar(Q)$; *and (3)* an *aggregation function* $\alpha : \mathcal{B}_{\text{fin}}(\mathbb{R}) \rightarrow \mathbb{R}$. We will make the convenient assumption that α is zero on the empty set, that is, $\alpha(\emptyset) = 0$. The semantics of $A = \alpha \circ \tau \circ Q$ is fairly straightforward: given a database D, the result $A(D)$ of applying A to D is the number

$$\alpha(\{\!\{\tau(\vec{t}) \mid \vec{t} \in Q(D)\}\!\}) .$$

We will discuss general value functions τ and make the assumption that $\tau(\vec{t})$ is computable in polynomial time. Another important assumption we make in most of our results is that τ is *localized* in the sense that it is determined by one of the atoms in the CQ. Formally, we say that τ is *localized* on an atom $R(\vec{z})$ of $Q(\vec{x})$ if for all databases D_1 and D_2 and homomorphisms $h_1 \in Hom(Q, D_1)$ and $h_2 \in Hom(Q, D_2)$, if $h_1(\vec{z}) = h_2(\vec{z})$, then $\tau(h_1(\vec{x})) = \tau(h_2(\vec{x}))$; that is, τ is completely determined by the assignment to the variables of R. In this case, we may write $\tau(h(\vec{z}))$ instead of $\tau(h(\vec{x}))$. Furthermore, we write $\tau \equiv c$ if τ is the constant function mapping each tuple in $\text{Const}^{ar(Q)}$ to the value $c \in \mathbb{R}$.

Throughout the manuscript, we will refer to the following specific value functions, assuming that the relevant Const value is a number:

$$\tau^i_{\text{id}}(a_1, \ldots, a_k) := a_i \tag{2}$$

$$\tau^i_{>b}(a_1, \ldots, a_k) := 1 \text{ if } a_i > b, \text{ and } 0 \text{ otherwise} \tag{3}$$

$$\tau^i_{\text{ReLU}}(a_1, \ldots, a_k) := a_i \text{ if } a_i > 0, \text{ and } 0 \text{ otherwise} \tag{4}$$

Here, the parameter b can be any fixed number. Note that each of the three is localized on some atom, since it is determined by one entry of the tuple. In the case where $k = 1$, we may omit the superscript and write just τ_{id}, $\tau_{>b}$, and τ_{ReLU} (where "ReLU" stands for Rectified Linear Unit).

For α, we will focus on several common functions: sum (Sum), count-distinct (CDist), average (Avg), minimum (Min), maximum (Max), median (Med), and more generally q-quantile (Qnt_q). As conventional, we set $\mathsf{Qnt}_q(B) = \frac{1}{2}(x_{\lceil q|B| \rceil} + x_{\lfloor q|B|+1 \rfloor})$ where x_i is the i-th element of B when the elements are sorted by increasing order. For our exploration, we will also examine the aggregate function *has-duplicates* (Dup) that evaluates to 1 if the multiplicity of at least one element is two or more, and 0 otherwise. In notation:

$$\text{For } B = (X, \mu), \text{ we have: } \mathsf{Dup}(B) = \begin{cases} 1 & \text{if there is } a \in X \text{ with } \mu(a) \geq 2 \\ 0 & \text{otherwise.} \end{cases}$$

Example 2.2. The following schema $\mathcal{S}$ is used for the database of an educational institute offering individual courses: Earns(person, salary), Course(name, number), Took(person, course). The following AggCQ returns the average salary of people who took courses in the institution.

$$A \;=\; \mathsf{Avg} \circ s \circ \big(Q(p, s) \leftarrow \textsc{Earns}(p, s), \textsc{Took}(p, c), \textsc{Course}(\_, c)\big) \tag{5}$$

Here, $\alpha = \mathsf{Avg}$ and $\tau(p, s)$ is simply s; in particular, τ is localized (on the relation Earns). Note that a person may take many courses, but she is counted only once for the average due to the projection of Q. An example of a non-localized τ is in the following AggCQ over the schema with Cargo(number, weight), Truck(number, weight), and Carries(truck, cargo).

$$A' \;=\; \mathsf{Max} \circ (w_c + w_t) \circ \big(Q(c, t, w_c, w_t) \leftarrow \textsc{Cargo}(c, w_c), \textsc{Carries}(t, c), \textsc{Truck}(t, w_t)\big)$$

This query asks for the maximal weight of a truck loaded with cargo. The value function τ is non-localized since its determination requires attributes from both Cargo and Truck. ◇

Shapley value for AggCQs. We study the computational complexity of measuring the contribution of a fact f to the result of an aggregate query A over a database D. We model this contribution as the Shapley value of f in the cooperative game (P, ν), where P is the database D (which is a set of facts, each being a player) and the utility ν is defined by $\nu(C) := A(C)$ for every $C \subseteq P$. Moreover, following the common modelling of the responsibility for database queries [35, 40], we assume that D consists of *endogenous* facts and *exogenous* facts. Facts of the latter kind are taken for granted and do not participate in the game; hence, the set P of players is restricted to the endogenous facts.

Formally, we assume that a given database D is the union of two disjoint databases (over the same schema $\mathcal{S}$): the database D^{n} contains the endogenous facts, and the database D^{x} contains the exogenous facts. For a fixed schema $\mathcal{S}$ and aggregate query A, the computational problem is defined as follows. Given a database D and a fact $f \in D^{\mathsf{n}}$, calculate the value $\mathsf{Shapley}(f, A)[D]$ as defined in Equation (1), where:

- The set P of players is D^{n}.
- The utility function ν is defined by $\nu(C) := A(C \cup D^{\mathsf{x}}) - A(D^{\mathsf{x}})$.

To simplify notation, we drop D when it is clear from context.

Example 2.3. Consider again the AggCQ of Equation (5). Suppose that we wish to measure the contribution of every course to the average salary (or the median if α is $\frac{1}{2}$-Qnt) of the past graduates. Then we make all Course facts endogenous and all other facts exogenous. Given a specific course represented as a fact $f = \textsc{Course}(c, n)$, its contribution would be quantified as $\mathsf{Shapley}(f, A)$. ◇

Remark 2.4. Previous work in database management has adopted the Shapley value largely because it is a conventional measure across disciplines, including other areas of data science [16, 31,

49]. This makes it an attractive candidate for a well-defined, application-independent attribution measure that can be offered as part of a database system service. In certain situations, the axiomatic properties that characterize the Shapley value are indeed compelling. For example, suppose that endogenous tuples represent employees or teams, and attribution serves to determine their reward for success. Furthermore, suppose that success is defined by an aggregate query Q that can be expressed as the sum of other aggregate queries Q' (e.g., the number of distinct new customers in the US equals the sum of the numbers of distinct new customers across the states). In such a setting, it is arguably desirable that the reward remains the same whether it is based on Q directly or it is given for each Q' separately (linearity). Similarly, it may appear fair that two facts indistinguishable in their impact on the query receive the same reward (symmetry), and that facts with no impact receive none (null player). Finally, to determine whether an attribution score is considered high or low, it is useful that the scores are calibrated so that the contributions of all endogenous tuples sum up to the actual result of the aggregate query (efficiency). We further discuss the rationality of the Shapley value for AggCQs in Section 8. ◇

We focus on the *data complexity* of computing $\mathrm{Shapley}(f, A)$. This means that the AggCQ A is assumed to be fixed, and the input consists of the database D and the fact $f \in D$. When we prove $\mathrm{FP}^{\#\mathrm{P}}$-completeness, we show only $\mathrm{FP}^{\#\mathrm{P}}$-hardness; completeness is done using standard techniques for showing membership of probability computation in $\mathrm{FP}^{\#\mathrm{P}}$ [21, 26].

3 General Machinery

We first present general machinery that we use throughout the manuscript, both for establishing hardness results on the Shapley value and for algorithms to compute this value.

3.1 General Hardness Result

We begin with a general hardness result. Our starting point is the following result of Livshits et al. [35], showing the hardness of the Shapley value for aggregate queries whenever the CQ is non-∃-hierarchical and without self-joins.

Theorem 3.1. [35, Theorem 4.8] *Let $A = \alpha \circ \tau \circ Q$ be an AggCQ. Suppose that Q has no self-joins. If Q is not ∃-hierarchical, then it is $\mathrm{FP}^{\#\mathrm{P}}$-complete to compute $\mathrm{Shapley}(f, A)$, unless A is a constant query (namely, 0, since we assume $\alpha(\emptyset) = 0$).*

From Theorem 3.1 we conclude that, for every nontrivial AggCQ, a necessary condition for polynomial-time Shapley computation is that the underlying CQ is ∃-hierarchical. The same work also showed that, for Sum (and Count), this condition is also sufficient. The next argument establishes that this is no longer true for the aggregate functions that we study in this work.

An aggregate function α is *constant per singleton* if for all $a \in \mathbb{R}$ and nonempty bags B and B' over the singleton $\{a\}$ it holds that $\alpha(B) = \alpha(B')$. For example, each of Min, Max, CDist, Avg, and Qnt_q is constant per singleton.

Proposition 3.2. *Let $A = \alpha \circ \tau \circ Q$ be an aggregate query such that τ is a constant function $\tau \equiv c$ and α is constant per singleton. For all databases D and facts $f \in D$ it holds that*

$$\mathrm{Shapley}(f, A) = \alpha(\{\!\{c\}\!\}) \cdot \mathrm{Shapley}(f, Q_{\mathrm{bool}})$$

where Q_{bool} is the Boolean CQ obtained from Q by viewing every free variable as existential.

Under the assumption that $\alpha(\{\!\{c\}\!\})$ is nonzero, Proposition 3.2 immediately yields a Turing reduction from computing $\mathrm{Shapley}(f, Q_{\mathrm{bool}})$ to computing $\mathrm{Shapley}(f, A)$ via division by $\alpha(\{\!\{c\}\!\})$. This shows that:

Theorem 3.3. *Let $A = \alpha \circ \tau \circ Q$ be an AggCQ such that τ is a constant function $\tau \equiv c$ and α is constant per singleton with $\alpha(\{\!\{c\}\!\}) \neq 0$. Suppose that Q has no self-joins. If Q is not all-hierarchical, then it is $\mathrm{FP}^{\#\mathrm{P}}$-complete to compute* $\mathrm{Shapley}(f, A)$. *In particular, this is the case for the aggregate functions* Min, Max, CDist, Avg, *and* Qnt_q *for all* $q \in (0, 1)$.

Hence, for the AggCQs we study in this work, the class of tractable underlying CQs is contained in the class of all-hierarchical CQs. In particular, this class is strictly narrower than the class of tractable CQs for sum (and count), namely that of the ∃-hierarchical CQs [36].

3.2 Algorithmic Template for Hierarchical CQs

We now describe a generic template that we apply for computing $\mathrm{Shapley}(f, A)$ when the underlying CQ Q is all-hierarchical. We begin with a folklore technique for reducing the computation to an expectation computation. Our goal is to compute $\mathrm{Shapley}(f, A)$ on a database D. In the following equations, we denote by F the database that is identical to D, except that f is exogenous rather than endogenous. We also denote by G the database $D \setminus f$. Note the following (referring to Equation (1)):

$$\begin{aligned}\mathrm{Shapley}(f, A) &= \sum_{E \subseteq D^{\mathsf{n}} \setminus \{f\}} q_{|E|} \cdot (A(E \cup \{f\} \cup D^{\mathsf{x}}) - A(E \cup D^{\mathsf{x}})) \\ &= \sum_{k=0}^{|D^{\mathsf{n}}|-1} q_k \left(\left(\sum_{E \in \binom{F^{\mathsf{n}}}{k}} A(E \cup F^{\mathsf{x}}) \right) - \left(\sum_{E \in \binom{G^{\mathsf{n}}}{k}} A(E \cup G^{\mathsf{x}}) \right) \right)\end{aligned}$$

Therefore, it suffices to be able to compute the value $\sum_{E \in \binom{D^{\mathsf{n}}}{k}} (A(D^{\mathsf{x}} \cup E))$, that is, the sum of the results of A over all of the databases that are obtained from D by removing all but k endogenous facts. Most of the algorithms we devise will compute this value, which we denote simply by $\mathsf{sum}_k(A, D)$:

$$\mathsf{sum}_k(A, D) \coloneqq \sum_{E \in \binom{D^{\mathsf{n}}}{k}} A(D^{\mathsf{x}} \cup E) \tag{6}$$

Notably, whenever we present an algorithm for the Shapley value by calculating $\mathsf{sum}_k(A, D)$, it applies to the class of the *Shapley-like scores* [33] that includes the Shapley value, as well as other contribution measures such as the *Banzhaf score* [6] that was studied in the context of membership in CQ answers, either knowingly [1] or unawarely [50]. Unfortunately, other algorithms (that rely, e.g., on the linearity of expectation) are not known to generalize to Shapley-like scores, and so is the case for the lower bounds – specialized reductions should be constructed for these (and possibly algorithms for classes that are considered intractable in our Shapley setting).

In general, the computation may not produce $\mathsf{sum}_k(A, D)$ directly, but rather a value or function $\mathcal{P}[Q, D]$ from which $\mathsf{sum}_k(A, D)$ is easily computable. The computation of $\mathcal{P}[Q, D]$ is done via the dynamic programming of Figure 2, which entails the computation of $\mathcal{P}[Q', D']$ for "smaller" Q' and D'. This program follows the general structure of typical algorithms for Boolean hierarchical CQs, such as the ones used for computing the probability of Q in a probabilistic database [18] and computing the Shapley value of Q [35]. Note, however, that we do not assume that Q is Boolean.

3.2.1 Notation. In the algorithm, we use the following terminology. A *root* variable of the CQ Q is a variable x that appears in every atom. The values that x *can take* are the values of D that occur in every column that corresponds to a position where x occurs in Q. If a is such a value, then a fact f is consistent with $x \mapsto a$ if f can be obtained from an atom $R_j(\vec{z}_j)$ of Q by replacing x with a and replacing every other variable with arbitrary constants. The *subset of D consistent with $x \mapsto a$* is the database that consists of all facts of D that are consistent with some atom of Q. In addition, we denote by $Q_{x \mapsto a}$ that CQ that is obtained by Q by replacing each body occurrence of x with a; in addition, if x is a head variable, then x is removed from the head. Hence, $vars(Q_{x \mapsto a}) = vars(Q) \setminus \{x\}$.

We say that the CQ Q is a *Cartesian product* of the CQs Q_1 and Q_2 if Q_1 and Q_2 have disjoint sets of relation symbols and disjoint sets of variables, the set of atoms of Q is the union of the sets

Generic dynamic programming for all-hierarchical CQs
Input: Database D, all-hierarchical CQ Q, and $\mathcal{P}[Q', D']$ for all relevant smaller Q' and D'
Goal: Compute $\mathcal{P}[Q, D]$

if Q has no variables **then** ▷ Constant number of relevant facts, compute and terminate
  Compute $\mathcal{P}[Q, D]$ directly and **return**
end if
if Q has a root variable x **then**
  Let $\{a_1, \dots, a_n\}$ be the values that x can take
  $D_1 \leftarrow$ the subset of D consistent with $x \mapsto a_1$
  $D_{\text{tmp}} \leftarrow D_1$
  $\mathcal{P}[Q, D_{\text{tmp}}] \leftarrow \mathcal{P}[Q_{x \mapsto a_1}, D_1]$
  for $i = 2$ to n **do**
    $D_i \leftarrow$ the subset of D consistent with $x \mapsto a_i$
    $\mathcal{P}[Q, D_i] \leftarrow \mathcal{P}[Q_{x \mapsto a_i}, D_i]$
    $\mathcal{P}[Q, D_{\text{tmp}} \cup D_i] \leftarrow \text{combine}_{\cup}(\mathcal{P}[Q, D_{\text{tmp}}], \mathcal{P}[Q, D_i])$ ▷ $\mathcal{P}[Q, D_i] = \mathcal{P}[Q_{x \mapsto a_i}, D_i]$
    $D_{\text{tmp}} \leftarrow D_{\text{tmp}} \cup D_i$
  end for ▷ Last $D_{\text{tmp}} \cup D_i$ is D, so $\mathcal{P}[Q, D]$ is computed
else ▷ Q is a cross product
  Suppose that Q is a cross product of Q_1 and Q_2
  Let D_1 and D_2 be the subsets of D with the relations of Q_1 and Q_2, respectively
  $\mathcal{P}[Q, D] \leftarrow \text{combine}_{\times}(\mathcal{P}[Q_1, D_1], \mathcal{P}[Q_2, D_2])$
end if

Fig. 2. Generic algorithmic scheme for hierarchical CQs

of atoms of Q_1 and Q_2, and the set of head variables of Q is the union of the set of head variables of Q_1 and Q_2. As an example, the CQ $Q(x, y) \leftarrow R(x), S(y, z), T(z)$ is the Cartesian product of the CQs $Q_1(x) \leftarrow R(x)$ and $Q_2(y) \leftarrow S(y, z), T(z)$.

3.2.2 Algorithm. The above notation covers all terminology used in the generic algorithm of Figure 2, except for two subroutines that need to be devised for each specific $\mathcal{P}$:

- The subroutine $\text{combine}_{\cup}(\mathcal{P}[Q, D'], \mathcal{P}[Q, D''])$ computes $\mathcal{P}[Q, D' \cup D'']$, assuming that D' and D'' are disjoint and $Q(D' \cup D'') = Q(D') \cup D(D'')$.
- The subroutine $\text{combine}_{\times}(\mathcal{P}[Q', D'], \mathcal{P}[Q'', D''])$ computes $\mathcal{P}[Q, D' \cup D'']$ assuming that Q is the Cartesian product of Q' and Q''.

Importantly, we need to clarify how τ is defined on the generated CQs. Recall that τ is localized on a single relation R. When we break Q as a Cartesian product of Q' and Q'', the relation R belongs to just one of the two, say Q'. The semantics of $\mathcal{P}$ on Q'' will not relate to τ. However, we assume that τ is defined on Q', since τ is localized on R and, by definition, it is a function of the assignment to the free variables of R.

Formally speaking, τ is defined over the answers to Q, but *not* over the answers to $Q_{x \mapsto a}$, since the latter may lack the values assigned to x (if x is a free variable). Nevertheless, since the missing value is always a in the answers to $Q_{x \mapsto a}$, we can assume that τ is defined on the answers of $Q_{x \mapsto a}$ by implicitly replacing τ with τ' defined by $\tau'(\vec{t}) = \tau(\vec{t}_a)$ where $\vec{t}_a$ is obtained from $\vec{t}$ by inserting a to every position where x is in the head of Q.

In summary, the following describes a recipe to instantiate the generic algorithm of Figure 2:

Instantiating the generic algorithm for an aggregate function α:

Let $A = \alpha \circ \tau \circ Q$ be an aggregate query, let D be an input database, and let f be a given fact.

(1) Define the content of the data structure $\mathcal{P}$.
(2) Show how $\mathsf{sum}_k(A, D)$ is computable from $\mathcal{P}[Q, D]$ for all $k = 0, \ldots, |D^{\mathsf{n}}|$.
(3) Devise an algorithm for implementing $\mathsf{combine}_{\cup}$
(4) Devise an algorithm for implementing $\mathsf{combine}_{\times}$.

4 Count Distinct, Min, Max

We start our analysis by considering the aggregation functions $\alpha \in \{\mathsf{CDist}, \mathsf{Min}, \mathsf{Max}\}$. First, we observe that all of them are constant per singleton. By Theorem 3.3, there is a value function τ such that Shapley value computation of $\alpha \circ \tau \circ Q$ is intractable for all non-all-hierarchical CQs Q. The following theorem states that being non-all-hierarchical is not only sufficient for the existence for such τ, but also necessary.

Theorem 4.1. *Let Q be a CQ without self-joins and $\alpha \in \{\mathsf{Min}, \mathsf{Max}, \mathsf{CDist}\}$. Denote by A_τ the aggregate query for $\alpha \circ \tau \circ Q$.*

(1) *If Q is all-hierarchical, then $\mathrm{Shapley}(f, A_\tau)$ is computable in polynomial time, given a database D and a fact f, for every localized τ.*
(2) *Otherwise, there is a localized τ such that computing $\mathrm{Shapley}(f, A_\tau)$, given a database D and a fact f, is $\mathrm{FP}^{\#\mathrm{P}}$-complete.*

In the next subsections, we will present polynomial-time algorithms for all-hierarchical queries.

4.1 Count Distinct

To get an intuition for the Shapley values for CDist-aggregation, we first state a closed formula for simplest case of a single relational query. In that case, the Shapley value matches the intuitive contribution of a fact: one over the number of all facts with the same τ-value.

Proposition 4.2. *Let $Q(\vec{x}) \leftarrow R(\vec{x})$ and assume that all facts in R are endogenous. Then*

$$\mathrm{Shapley}(R(\vec{t}), \mathsf{CDist} \circ \tau \circ Q) = \frac{1}{|\{R(\vec{t}') \mid \tau(\vec{t}') = \tau(\vec{t})\}|}.$$

Now, we present an algorithm for general all-hierarchical CQs that uses a direct reduction to the Boolean case. It is based on the insight that we can express CDist as a sum over indicator variables χ_a with $\chi_a(B) = 1$ if $a \in B$ and 0 otherwise. Since the indicator variables can be computed using Boolean queries, we obtain the following lemma.

Lemma 4.3. *Let Q be a CQ without self-joins, τ be localized on the atom $R(\vec{z})$, and D be a database. For $a \in \mathbb{R}$, let D_a be the database obtained from D by removing all facts $R(\vec{b})$ with $\tau(\vec{b}) \neq a$. Furthermore, let $Q_{\mathrm{bool}} \leftarrow Q(\vec{x})$. Let $A = \mathsf{CDist} \circ \tau \circ Q$. Then, for each fact $f \in D^{\mathsf{n}}$, we have*

$$\mathrm{Shapley}(f, A)[D] = \sum\nolimits_{a \in (\tau \circ Q)(D)} \mathrm{Shapley}(f, Q_{\mathrm{bool}})[D_a],$$

with the convention that $\mathrm{Shapley}(f, Q_{\mathrm{bool}})[D_a] = 0$ for $f \notin D_a$.

This lemma yields a polynomial-time algorithm and shows Theorem 4.1 for $\alpha = \mathsf{CDist}$: If Q is all-hierarchical, then Q_{bool} is a Boolean hierarchical CQ and we can use the polynomial-time algorithm from [35] to compute $\mathrm{Shapley}(f, Q_{\mathrm{bool}})[D_a]$ for the at most $|D|^{ar(Q)}$ values $a \in (\tau \circ Q)(D)$.

4.2 Min and Max

Next, we present a polynomial-time algorithm for computing the Shapley value for Max-aggregation on all-hierarchical queries. For Min-aggregation, we can use the same algorithm by first negating all values in the database D and then negating all obtained Shapley values. For intuition, we again start with a closed formula for the simplest case of a single-atom query.

Proposition 4.4. *Let $Q(\vec{x}) \leftarrow R(\vec{x})$. Assume all facts are endogenous. Let q_k as in Equation (1) and for $a \in \mathbb{R}$, let $m[\leq a]$ and $m[<a]$ denote the number of facts $R(\vec{t})$ with $\tau(\vec{t}) \leq a$ or $< a$, respectively.*

$$\mathrm{Shapley}(R(\vec{t}), \mathsf{Max} \circ \tau \circ Q) = \frac{\tau(\vec{t})}{|D|} + \sum_{\substack{a \in (\tau \circ Q)(D) \\ a < \tau(\vec{t})}} (\tau(\vec{t}) - a) \sum_{k=1}^{n-1} q_k \left(\binom{m[\leq a]}{k} - \binom{m[<a]}{k} \right).$$

The proposition illustrates, for instance, that even facts with small τ-values contribute to the Max aggregation.

Our algorithm for general all-hierarchical CQs is based on the generic algorithm shown in Figure 2. Let $A = \mathsf{Max} \circ \tau \circ Q$ be such that τ is localized on an atom $R(\vec{z})$. We define the data structure $\mathcal{P}[Q', D']$ as follows:

- If D' contains the relation R that determines τ, then $\mathcal{P}[Q', D']$ is a function that maps every pair (a, k) with $a \in (\tau \circ Q')(D')$ and $0 \leq k \leq |D'^{\mathsf{n}}|$ to the number of subsets $E \in \binom{D'^{\mathsf{n}}}{k}$ with $\max((\tau \circ Q')(E \cup D'^{\mathsf{x}})) = a$. (Recall that τ is defined on Q', as explained at the end of Section 3.2.)
- Otherwise, $\mathcal{P}[Q', D']$ maps each number $k = 0, 1, \ldots, |D'^{\mathsf{n}}|$ to the number of subsets $E \in \binom{D'^{\mathsf{n}}}{k}$ such that $Q(E \cup D'^{\mathsf{x}})$ is nonempty.

With this data structure at hand, we can simply compute $\mathsf{sum}_k(A, D)$ via

$$\mathsf{sum}_k(A, D) = \sum_{E \in \binom{D^{\mathsf{n}}}{k}} (\mathsf{Max} \circ \tau \circ Q)(E \cup D^{\mathsf{x}}) = \sum_{a \in (\tau \circ Q)(D)} a \cdot \mathcal{P}[Q, D](a, k).$$

We next describe the algorithms for $\mathsf{combine}_\cup$ and $\mathsf{combine}_\times$ for this data structure. More details on the computation can be found in the arXiv version of this paper [52].

Let us first discuss the algorithm for $\mathsf{combine}_\cup(\mathcal{P}[Q', D_1], \mathcal{P}[Q', D_2])$ and let first D_1 and D_2 contain R. Let $E \in \binom{D_1^{\mathsf{n}} \cup D_2^{\mathsf{n}}}{k}$ and $E_i = E \cap D_i^{\mathsf{n}}$. Since $(\tau \circ Q')(E \cup D_1^{\mathsf{x}} \cup D_2^{\mathsf{x}})$ is the union of the sets $(\tau \circ Q')(E_1 \cup D_1^{\mathsf{x}})$ and $(\tau \circ Q')(E_2 \cup D_2^{\mathsf{x}})$, its maximal value is equal to a if and only if the maximal values of $(\tau \circ Q')(E_i \cup D_i^{\mathsf{x}})$ for $i = 1, 2$ are less than or equal to a and equality holds for at least one i. Hence, $\mathsf{combine}_\cup(\mathcal{P}[Q', D_1], \mathcal{P}[Q', D_2])(a, k)$ can be computed by summing up $\mathcal{P}[Q', D_1](a_1, k_1) \cdot \mathcal{P}[Q', D_2](a_2, k_2)$ over all combinations with $k_1 + k_2 = k$ and $\max(a_1, a_2) = a$.

If D_1 and D_2 do not contain R, we can argue about the complement $\binom{|D'|}{k} - \mathcal{P}[Q', D']$ as follows. $Q'(E \cup D_1^{\mathsf{x}} \cup D_2^{\mathsf{x}})$ is empty if and only if both $Q'(E_1 \cup D_1^{\mathsf{x}})$ and $Q'(E_2 \cup D_2^{\mathsf{x}})$ are empty. Hence, we sum up the product of the complements of $\mathcal{P}[Q', D_1](k_1)$ and $\mathcal{P}[Q', D_2](k_2)$ over all $k_1 + k_2 = k$.

The algorithm for $\mathsf{combine}_\times(\mathcal{P}[Q_1, D_1], \mathcal{P}[Q_2, D_2])$ is straightforward: For a Cartesian product to be nonempty, we need both sides to be nonempty and if D_1 contains R (and D_2 does not), we need to attain a as the maximum of $(\tau \circ Q_1)(E_1 \cup D_1^{\mathsf{x}})$ and have $Q_2(E_2 \cup D_2^{\mathsf{x}})$ nonempty.

Together, these algorithms show how to compute Shapley values for Max-aggregation over all-hierarchical CQs in polynomial time and, hence, completes the proof of Theorem 4.1.

Remark 4.5. A polynomial-time algorithm for all-hierarchical queries with Min and Max aggregation follows from the work of Abramovich et al. [2], who established, independently of our work, polynomial-time algorithms for Min and Max aggregation over d-trees. Such d-trees can, in turn,

be obtained in polynomial time from a database and an all-hierarchical CQ, as shown in the context of knowledge compilation for query evaluation over probabilistic databases [22, 43]. ◇

5 Average and Quantile

In the previous section, we established that the tractability frontier for the aggregate functions Min, Max, and CDist coincides with the class of all-hierarchical CQs. In this section, we turn our attention to two other aggregate functions—Avg (average) and Qnt (quantile)—and demonstrate that their Shapley value is inherently harder, as their tractability frontier is strictly narrower:

Theorem 5.1. *Let Q be a CQ without self-joins, $q \in (0,1)_{\mathbb{Q}}$ fixed, and $\alpha \in \{\mathsf{Avg}, \mathsf{Qnt}_q\}$. Denote by A_τ the aggregate query $\alpha \circ \tau \circ Q$.*

1. *If Q is q-hierarchical, then $\mathrm{Shapley}(f, A_\tau)$ is computable in polynomial time, given a database D and a fact f, for every localized τ.*
2. *Otherwise, there is a localized τ such that computing $\mathrm{Shapley}(f, A_\tau)$, given a database D and a fact f, is $\mathrm{FP}^{\#\mathrm{P}}$-complete.*

The notion of q-hierarchical was introduced by Berkholz, Keppeler, and Schweikardt [7], and has subsequently been used to characterize the fine-grained complexity of query-evaluation tasks in several studies (e.g., [7, 8, 30, 42, 54]). To the best of our knowledge, Theorem 5.1 is the first to show that the q-hierarchical property also delineates a boundary of coarse-grained complexity. In the remainder of this section, we prove Theorem 5.1. We begin with the positive side.

5.1 An Algorithm for Q-Hierarchical Queries

We first give a closed formula for Shapley values for Avg-aggregation over the simplest case of a single relational query. It is tempting to think that the Shapley value of a fact $R(x)$ is $\frac{\tau(\vec{x})}{n}$, but this is not the case, since the denominator of the average changes with the size of the subset considered.

Proposition 5.2. *Let $Q(\vec{x}) \leftarrow R(\vec{x})$ and assume that all facts in R are endogenous. Let $n = |D|$ and $H(n) = \sum_{k=1}^{n} \frac{1}{k}$ denote the n-th partial sum of the harmonic series. Then*

$$\mathrm{Shapley}(R(\vec{t}), \mathsf{Avg} \circ \tau \circ Q) = \frac{H(n)}{n} \cdot \tau(\vec{t}) + \frac{H(n)-1}{n(n-1)} \sum_{R(\vec{t}') \neq R(\vec{t})} \tau(\vec{t}')\,.$$

Unlike the previous aggregation functions we considered, here we omit the closed formula for Shapley values for Qnt for a singleton CQ $Q(x) \leftarrow R(x)$ since we do not find it considerably simpler than the algorithm for general q-hierarchical CQs. Instead, we turn our attention to the general algorithm by instantiating the generic algorithm from Figure 2 for q-hierarchical CQs Q.

To take full advantage of the structure of q-hierarchical CQs, we need to specify the following: In step 4, if Q' has several root variables, then the algorithm chooses a *free* root variable if there is one. If there are only existentially quantified root variables, then Q' needs to be Boolean by the q-hierarchical property.

We next present a data structure $\mathcal{P}[Q', D']$ that allows us to compute $\mathsf{sum}_k(\alpha \circ \tau \circ Q, D)$ for $\alpha = \mathsf{Avg}$ as well as for $\alpha = \mathsf{Qnt}_q$. It is defined as follows.

- If D' contains the relation R that determines τ, then $\mathcal{P}[Q', D']$ is a function that maps every quintuple $(a, k, \ell_<, \ell_=, \ell_>)$ with $a \in (\tau \circ Q')(D')$, $0 \le k \le |D^{\mathsf{n}}|$, and $0 \le \ell_< + \ell_= + \ell_> \le |Q'(D')|$ to the number of subsets $E \in \binom{D'^{\mathsf{n}}}{k}$ with the property that $(\tau \circ Q')(E \cup D'^{\mathsf{x}})$ contains the element a exactly $\ell_=$ times, exactly $\ell_<$ elements b with $b < a$, and exactly $\ell_>$ elements c with $c > a$. (Recall that τ is defined on Q', as explained at the beginning of Section 3.2.)
- Otherwise, $\mathcal{P}[Q', D']$ maps each pair (k, ℓ) with $0 \le k \le |D'^{\mathsf{n}}|$ and $0 \le \ell \le |Q'(D')|$ to the number of subsets $E \in \binom{D'^{\mathsf{n}}}{k}$ such that $|Q'(E \cup D'^{\mathsf{x}})| = \ell$.

To compute $\mathsf{sum}_k(\mathsf{Avg} \circ \tau \circ Q, D)$, we write the sum over a bag B as the sum over all elements of the set underlying B times their multiplicity in B. Using this in the formula for the average yields

$$\mathsf{sum}_k(\mathsf{Avg} \circ \tau \circ Q, D) = \sum_{a \in (\tau \circ Q)(D)} \sum_{0 \leq \ell_< + \ell_= + \ell_> \leq |Q(D)|} \frac{a \cdot \ell_=}{\ell_< + \ell_= + \ell_>} \mathcal{P}[Q, D](a, k, \ell_<, \ell_=, \ell_>) \,.$$

To compute $\mathsf{sum}_k(\mathsf{Qnt}_q \circ \tau \circ Q, D)$, we make the following trivial observation. The value a is the i-th smallest element of a bag B if B contains fewer than i elements smaller than a and at least i elements smaller than or equal to a. We further decompose $\mathsf{Qnt}_q(B)$ into a sum over all elements of the set underlying B times their contribution (either 0, 1 or $\frac{1}{2}$) to $\mathsf{Qnt}_q(B)$. Applying this to $B = (\tau \circ Q)(E \cup D^{\mathsf{x}})$ allows us to compute $\mathsf{sum}_k(\mathsf{Qnt}_q \circ \tau \circ Q, D)$ via

$$\mathsf{sum}_k(\mathsf{Qnt}_q \circ \tau \circ Q, D) = \sum_{a \in (\tau \circ Q)(D)} \sum_{0 \leq \ell_< + \ell_= + \ell_> \leq |Q(D)|} a \cdot f_q(\ell_<, \ell_=, \ell_>) \cdot \mathcal{P}[Q, D](a, k, \ell_<, \ell_=, \ell_>) \,.$$

where we denote the indicator function of a condition C by $\chi(C)$ to define $f_q(\ell_<, \ell_=, \ell_>)$ as

$$f_q(\ell_<, \ell_=, \ell_>) := \frac{1}{2}\Big(\chi\big(\ell_< < \lceil q(\ell_< + \ell_= + \ell_>)\rceil\big) \cdot \chi\big(\ell_< + \ell_= \geq \lceil q(\ell_< + \ell_= + \ell_>)\rceil\big) + \chi\big(\ell_< < \lfloor q(\ell_< + \ell_= + \ell_>) + 1\rfloor\big) \cdot \chi\big(\ell_< + \ell_= \geq \lfloor q(\ell_< + \ell_= + \ell_>) + 1\rfloor\big)\Big) \,.$$

We show how to compute $\mathsf{combine}_\cup$ and $\mathsf{combine}_\times$ in polynomial time in the arXiv version of this paper [52]. The algorithms are similar to those in the previous section with the addition that they use the fact that for non-Boolean Q', the sets $Q'(D_1)$ and $Q'(D_2)$ are disjoint.

5.2 Hardness

We first show a reduction from the case of the following CQ, which is the simplest CQ that is all-hierarchical but not q-hierarchical:

$$Q_{xyy}(x) \leftarrow R(x, y), S(y) \tag{7}$$

The following lemma gives a general reduction from Q_{xyy}.

Lemma 5.3. *Consider an AggCQ of the form $\alpha \circ \tau \circ Q_{xyy}$, and let Q_0 be a CQ without self-joins, such that Q_0 is all-hierarchical but not q-hierarchical. There is a value function τ_0, having the form $\tau \circ \tau^i_{\mathsf{id}}$, and a polynomial-time algorithm that, given an input database D for Q_{xyy}, constructs an input database D_0 for Q_0 and a function $h : D^{\mathsf{n}} \to D_0^{\mathsf{n}}$ such that*

$$\mathrm{Shapley}(f, \alpha \circ \tau \circ Q_{xyy}) = \mathrm{Shapley}(h(f), \alpha \circ \tau_0 \circ Q_0)$$

for every endogenous fact $f \in D^{\mathsf{n}}$.

Hence, it suffices to prove the following, and we do so in the arXiv version of this paper [52].

Lemma 5.4. *For $A = \mathsf{Avg} \circ \tau_{\mathsf{ReLU}} \circ Q_{xyy}$ and $A = \mathsf{Qnt}_q \circ \tau_{>0} \circ Q_{xyy}$ for fixed $q \in (0, 1)_{\mathbb{Q}}$, computing $\mathrm{Shapley}(f, A)$, given a database D and a fact f, is $\mathrm{FP}^{\#\mathrm{P}}$-complete.*

(Recall that the value functions $\tau_{>b}$ and τ_{ReLU} are defined in Equations (2) and (4), respectively.)

Proof (sketch). For Avg, the proof is by a reduction from the *#Set-Cover* problem: given a set $\mathcal{X}$ of elements, a collection $\mathcal{Y}$ of subsets of $\mathcal{X}$, determine the number of collections $C \subseteq \mathcal{Y}$ that cover all of $\mathcal{X}$ (i.e., $\mathcal{X} = \cup C$). This problem is known to be #P-complete [46]. We construct an equation system over the variables $Z_{i,j}$ that, when solved, hold number of subsets of $\mathcal{Y}$ with j members, covering precisely i elements from $\mathcal{X}$. To construct this equation system, we define several databases D' and use their $\mathrm{Shapley}(f, A)$ to construct one equation (similarly to past proofs of hardness of the Shapley value, e.g., [5]). To crux of the proof is to show that the resulting matrix

is invertible; we show it by establishing that the matrix is the Kronecker product of the Hilbert matrix ($M_{i,j} = \frac{1}{i+j-1}$) and the Hankel matrix with factorial entries ($M_{i,j} = (i+j)!$).

For Qnt, we show how to construct a database D so that $A(D)$ simulates (in an intuitive sense) the Set-Cover game (where the goal of a coalition is to cover all elements) for $A = \mathsf{Qnt}_q \circ \tau_{>0} \circ Q_{xyy}$. Hardness for this game was established in [24]. □

6 Q-Hierarchy might not be Enough

Given the results of the previous section, one might wonder whether the restriction of being q-hierarchical suffices for the tractability of every (natural) aggregate function. More specifically, suppose that α is an aggregate function such that Shapley(f, A) is computable in polynomial time whenever the underlying CQ of A has a single relation; does it imply the tractability of Shapley(f, A) when the underlying CQ is q-hierarchical without self-joins?

This is not the case. In this section, we establish the negative answer by studying the aggregate function Dup (has-duplicates) and showing that the tractability frontier of the Shapley value is a class stricter than that of the q-hierarchical CQs.

The stricter class of CQs is that of the *strongly q-hierarchical* CQs, or *sq-hierarchical* for short. Let Q be a CQ. We say that Q is sq-hierarchical if Q is all-hierarchical and Q has no head variable whose set of atoms is strictly contained in that of another variable; that is, for all variables x and y of Q, if $\mathrm{atoms}(Q, x) \subsetneq \mathrm{atoms}(Q, y)$, then x is existential. Note that an sq-hierarchical CQ is necessarily q-hierarchical. For illustration, the following CQs are sq-hierarchical:

$$Q_1(x) \leftarrow R(x,y), S(x) \quad | \quad Q_2(x,y) \leftarrow R(x,y), S(x,y,z) \quad | \quad Q_3(x,z) \leftarrow R(x,y), S(x), T(z)$$

In contrast, the q-hierarchical CQ $Q_4(x,y) \leftarrow R(x,y), S(x)$ is *not* sq-hierarchical, since $\mathrm{atoms}(y) \subsetneq \mathrm{atoms}(x)$ and, yet, y is a head variable. In this section, we show that:

Theorem 6.1. *Let Q be a CQ without self-joins.*

(1) *If Q is sq-hierarchical, then* Shapley(f, A) *is computable in polynomial time for $A = \mathsf{Dup} \circ \tau \circ Q$ on every localized τ.*
(2) *Otherwise, there is a localized τ such that computing* Shapley(f, A)*, given a database D and a fact f, is $\mathrm{FP}^{\#\mathrm{P}}$-complete.*

Proof (sketch). Tractability is shown by an adaptation of the generic algorithm (Figure 2). We prove hardness as follows. Theorem 3.1 covers all non-∃-hierarchical CQs. We extend Theorem 3.3 to the Dup function, hence cover all CQs that are ∃-hierarchical but not all-hierarchical. We then prove hardness for the following two AggCQs, through a reduction from the problem of calculating the permanent of a 0/1-matrix, which is known to be #P-hard due to Valiant's Theorem [53]:

(1) $\mathsf{Dup} \circ \tau_{\mathsf{ReLU}} \circ Q_{xyy}$ for $Q_{xyy}(x) \leftarrow R(x,y), S(y)$.
(2) $\mathsf{Dup} \circ \tau^1_{\mathsf{id}} \circ Q^{\mathrm{full}}_{xyy}$ for $Q^{\mathrm{full}}_{xyy}(x,y) \leftarrow R(x,y), S(y)$.

From hardness (1) and Lemma 5.3, we conclude hardness for every CQ that is all-hierarchical but not q-hierarchical. Next, we prove a general reduction similar to Lemma 5.3, stating that if Q is q-hierarchical but not sq-hierarchical, then the computation of Shapley$(f, \mathsf{Dup} \circ \tau_{\mathsf{ReLU}} \circ Q^{\mathrm{full}}_{xyy})$ reduces to Shapley$(f, \mathsf{Dup} \circ \tau \circ Q)$ for some localized value function τ; hence, hardness (2) and the new general reduction jointly yield hardness for every CQ that is q-hierarchical but not sq-hierarchical, completing the story. The details are given in the arXiv version of this paper [52]. □

7 Discussion: Towards Finer Classifications

The dichotomy theorems established in the previous sections (namely, Theorems 4.1, 5.1 and 6.1) classify the AggCQs for each aggregation function α into two categories: the *tractable* ones, which

are tractable for *every* localized value function τ, and the *intractable* ones, which are $\mathrm{FP}^{\#\mathrm{P}}$-complete for *at least one* value function τ, which is localized on some predefined atom $R(\vec{z})$ (or a constant, as a special case). This classification, however, does not rule out the possibility that an AggCQ that is intractable under one value function may become tractable under a different value function τ, and even a natural one such as τ^i_{id}. Thus, there is potential for finer classifications that take τ into account. In this section, we explore this challenge and offer preliminary insights.

7.1 Choice of a Value Function

In the case of Sum and Count (Theorem 3.1), the classification applies to every function τ, as long as the resulting AggCQ is not constant [36]. For other aggregation functions, the choice of τ can definitely make a difference. For illustration, we inspect what happens when we replace the value function τ with the natural copying function τ_{id}.

We have seen in the proof sketch of Theorem 6.1 the hardness of $\mathsf{Dup} \circ \tau_{\mathsf{ReLU}} \circ Q_{xyy}$. Replacing τ_{ReLU} with τ_{id} would make the query constantly zero, and the Shapley value trivial, for every relational query Q, since $Q(D)$ cannot include duplicates by definition. As another example, consider the $\exists$-hierarchical CQ $Q(x) \leftarrow R(x), S(x,y), T(y)$ and the AggCQ $\mathsf{CDist} \circ \tau \circ Q$ for a constant value function. Computing the Shapley value is intractable, by Theorem 3.3. However, changing τ to τ_{id} makes it solvable in polynomial time: since τ_{id} is injective, this AggCQ is equivalent to $\mathsf{Count} \circ \tau_{>0} \circ Q$, where the Shapley value is tractable (since Count is tractable for every $\exists$-hierarchical CQ without self-joins [35]).

Contrasting the above examples, we next show that our hardness results for the other aggregation functions are robust to replacing the value function with τ_{id}. More generally, we show that every τ can be replaced by τ^i_{id}, as long as τ is a monotonically increasing (i. e. non-decreasing) function over τ^i_{id}, which applies $\tau^i_{>b}$, τ^i_{ReLU} and also $\tau \equiv c$.

Theorem 7.1. *Let $\alpha \circ \tau \circ Q$ be an AggCQ where $\alpha \in \{\mathsf{Min}, \mathsf{Max}, \mathsf{Avg}, \mathsf{Qnt}_q\}$ and Q has no self-joins. Suppose that $\tau = \gamma_{\mathsf{mon}} \circ \tau^i_{\mathrm{id}}$ for some number i and a monotonically increasing function $\gamma_{\mathsf{mon}} : \mathbb{R} \to \mathbb{R}$. If $\mathrm{Shapley}(f, \alpha \circ \tau \circ Q)$ is $\mathrm{FP}^{\#\mathrm{P}}$-complete to compute, then so is $\mathrm{Shapley}(f, \alpha \circ \tau^i_{\mathrm{id}} \circ Q)$.*

Hence, From Theorem 7.1 and the hardness results of the previous sections, we conclude that:

Corollary 7.2. *Let $A = \alpha \circ \tau^i_{\mathrm{id}} \circ Q$ be an AggCQ where $\alpha \in \{\mathsf{Min}, \mathsf{Max}, \mathsf{Avg}, \mathsf{Qnt}_q\}$. If Q has no self-joins and Q is not all-hierarchical, then computing $\mathrm{Shapley}(f, A)$ is $\mathrm{FP}^{\#\mathrm{P}}$-complete.*

7.2 Localization of a Value Function

Next, we ask whether the choice of the atom $R(\vec{z})$ on which τ is localized matters; that is, can it be the case that the hard AggCQ becomes tractable if τ is assumed to be localized on a specific R?

For the aggregation functions Sum and Count, the hardness of Theorem 3.1 is robust to the localization (as long as $\alpha \circ \tau$ is not constant). For Min, Max and CDist, hardness is covered by Theorem 3.3, which is also robust to the localization, in a trivial sense (as the constant value function is localized on every atom). It continues to hold (in a non-trivial sense) for the generalization of Corollary 7.2. Next, we show that our hardness results for Avg, Qnt_q, and Dup do not possess this robustness—they may become false if we replace the localization atom $R(\vec{z})$.

Consider the following AggCQs:

(1) $\mathsf{Avg} \circ \tau^1_{\mathsf{ReLU}} \circ Q_{xyyz}$ for $Q_{xyyz}(x,z) \leftarrow R(x,y), S(y), T(z)$.
(2) $\mathsf{Med} \circ \tau^1_{>0} \circ Q_{xyyz}$.
(3) $\mathsf{Dup} \circ \tau^1_{\mathrm{id}} \circ Q^{\mathrm{full}}_{xyy}$ for $Q^{\mathrm{full}}_{xyy}(x,y) \leftarrow R(x,y), S(y)$.

In each of the three, the computation of the Shapley value is $\mathrm{FP}^{\#\mathrm{P}}$-complete. For the first two, it can be shown by immediate reductions from $\mathsf{Avg} \circ \tau_{\mathsf{ReLU}} \circ Q_{xyy}$ and $\mathsf{Med} \circ \tau_{>0} \circ Q_{xyy}$, respectively,

(eliminating the effect of $T(z)$ by assigning to T a relation with a single, exogenous tuple) that were established to be hard in Lemma 5.4. The third AggCQ was shown to be hard in the proof sketch of Theorem 6.1. In contrast, for each of the three, the Shapley value is computable in polynomial time if we change the localization assumption from the first atom to the last atom:

Proposition 7.3. *For each of the following AggCQs, there is a polynomial-time algorithm for computing* Shapley(f, A)*, given a database D and a fact f.*

(1) $\mathsf{Avg} \circ \tau^2_{\mathsf{ReLU}} \circ Q_{xyyz}$ *for* $Q_{xyyz}(x, z) \leftarrow R(x, y), S(y), T(z)$.
(2) $\mathsf{Med} \circ \tau^2_{>0} \circ Q_{xyyz}$.
(3) $\mathsf{Dup} \circ \tau^2_{\mathsf{id}} \circ Q^{\mathsf{full}}_{xyy}$ *for* $Q^{\mathsf{full}}_{xyy}(x, y) \leftarrow R(x, y), S(y)$.

Hence, a valuable challenge for future research is to establish finer classifications that account for the identity of the atom on which localization is assumed.

7.3 Beyond Localized Value Functions

The polynomial-time algorithms presented so far crucially rely on the locality of the value function τ. In contrast, the case of non-localized value functions remains largely unexplored and is left as an open direction for future research. In this section, we outline the main challenges that arise when attempting to extend our results to this more general setting.

The algorithm of Livshits et al. [35] for Sum aggregation over ∃-hierarchical CQs applies to general (not necessarily localized) value functions τ, as long as τ can be computed in polynomial time. The reason for that is simple—the algorithm considers one answer at a time, independently of the others (due to the linearity of expectation), so it can simply compute τ upfront based on the considered answer, regardless of the number of atoms that determine τ.

However, when going beyond Sum, things get more complicated. For illustration, let us consider the case of Min and Max. The algorithm of Section 4 for Min/Max over all-hierarchical CQs extends to a non-localized τ, as long as it is a *monotonic monoid*. More precisely, if we assume that τ is the result $x_1 \otimes \cdots \otimes x_\ell$ of applying a non-decreasing monoid $\otimes : \mathbb{R}^2 \to \mathbb{R}$ on a sequence $(x_1, \ldots, x_\ell)$ of numeric free variables, then we can extend the algorithm for Shapley(f, A) to handle $\mathsf{Max} \circ \tau$. For example, this applies to $\mathsf{Max}(x_1 + x_2)$, $\mathsf{Max}(\max(x_1, x_2))$, and so on. (Similarly, we can support Min with a non-increasing monoid.) The extension is as follows. For each subquery Q' and relevant number a, we store in $\mathcal{P}$ the number of subsets E of D^{n} where the monoid, restricted to the variables of Q', has a as the maximal value. Due to the monotonicity, we can then realize the $\mathsf{combine}_\times$ function efficiently by considering all combinations of numbers a' and a'' with $a' \otimes a'' = a$. (We will give the complete argument in the extended version of this manuscript.)

Importantly, it is necessary to restrict τ (e.g., being a monotonic monoid) to extend our upper bound for min and max; it is insufficient to assume only that τ is computable in polynomial time (as in the case of sum). To see why, consider the Boolean CQ $Q_0() \leftarrow R_0(x), S_0(x, y), T_0(y)$. Suppose that all facts of S_0 are exogenous. In this case, computing Shapley(Q_0, f) is $\mathrm{FP}^{\#\mathrm{P}}$-hard [35]. Now, consider the AggCQ

$$A := \mathsf{Max} \circ \tau \circ Q \text{ for } Q(x, y) \leftarrow R(x), T(y)\,.$$

Define τ as follows. Assume that $\tau(x, y)$ is equal to 1 if x divides y and 0 otherwise. Then A can *simulate* the Boolean CQ Q_0 for the computation of the Shapley value with the following database D that we construct for A. Given a database D_0 for Q_0, choose a distinct prime p_c for each $c \in D_0^{R_0}$ and add it to D^R. In D^T, we store for each $d \in D_0^{T_0}$ the product over all p_c with $(c, d) \in D_0^{S_0}$ (or a power of another prime p' if no such c exists). This construction can be performed in polynomial time, since the nth prime is in $O(n \log n)$ [28]. Then, for each subset E_0 of D_0^{n}, it holds that $Q(D_0^{\mathsf{x}} \cup E_0)$ is

true if and only if $A(D^{\mathsf{x}} \cup E) = 1$, where E is the set of facts of D that corresponds to E_0. Therefore, each fact f of D_0 has the same Shapley value as its corresponding fact of D.

We conclude that there are polynomial-time computable functions τ for which the Shapley value for the min/max is intractable even for the Cartesian-product CQ (which is all-hierarchical and even sq-hierarchical). We conjecture that a similar situation applies to every aggregate function that we considered in this manuscript, with the exception of sum and count. Consequently, future extensions of this work for non-local τ will need to make some assumptions on τ.

8 Concluding Remarks

We investigated the complexity of computing the Shapley value of a tuple for AggCQs $\alpha \circ \tau \circ Q$ where the underlying CQ Q has no self-joins and the value function τ is localized. While the cases of sum and count have been resolved in prior work [35], our results shed light on the tractability frontiers of other common aggregate functions, including count-distinct, min, max, average, quantile, and has-duplicates. In particular, we showed that these functions capture different classes of hierarchical CQs, as shown in Figure 1.

Many avenues remain for future continuation of this research. Following the discussion on Section 7, an important direction is to pursue classification with respect to general classes of value functions τ. Another important direction is to extend the class of relational queries in the considered aggregate queries, namely the class of CQs without self-joins. While handling CQs with self-joins is traditionally a daunting challenge in the case of the Shapley value (and other complexity classifications, e.g., [15, 19, 44]), we can also explore the extension to the classes of the *constant-free unions of connected CQs*, and *connected homomorphism-closed graph queries*, where dichotomies for the Boolean Shapley value have been established by Bienvenu, Figueira, and Lafourcade [12].

In addition to the directions of extending Q and τ, we are interested in generalizing our study from specific (though commonly used) aggregate functions α to broad classes of aggregate functions that are identified by general properties (as exemplified in Theorem 3.3) that cover the aggregate functions of standard SQL and possibly many others. Specifically, an important future direction is to extend our results into full classifications that cover aggregate functions from wide classes of functions defined by intuitive properties.

Mirroring past applications of the Shapley value to databases [27, 35, 36], an important direction is to study the complexity of *approximating* the Shapley value with error guarantees. This is highly relevant to the pragmatics of calculating the Shapley value in reality, as well as the usage of SAT-solvers and circuits that were shown to provide practical solutions for both ordinary CQs [4, 20] and AggCQs for certain aggregation functions [2].

There are also directions that concern different database models. The work of Karmakar et al. [33] gives rise to a variant of the problem involving *probabilistic databases*: they studied the expected contribution score (for the general class of Shapley-like scores) of an endogenous fact when tuples are probabilistic (in the *tuple-independent model* [17]), and this problem generalizes naturally to the expected contribution to an aggregate query. In this vein, another direction regards the computation of the Shapley value for queries over ontologies (known as ontology-mediated query answering), where ontological axioms together with the database induce a set of possible worlds, as done recently by Bienvenu et al. [11]. In their work, the utility function for a Boolean query is the certainty of the query. For aggregate queries, we can adopt known semantics from the literature, such as those proposed by Calvanese et al. [14] or the range semantics by Afrati and Kolaitis [3].

Finally, it is also important to explore the rationality of using the Shapley value for aggregate queries over databases, in continuation of the discussion in Remark 2.4. Different application scenarios may call for rationality criteria other than those of the Shapley value, and it is therefore important for a database system to support a broad collection of attribution measures. On this

matter, Bienvenu et al. [13] argued that, in the case of Boolean conjunctive queries (or unions thereof), attribution should reflect rationality criteria based on the set of homomorphisms that include a given fact. They proposed the weighted sums of minimal supports (WSMS) measure, which satisfies these criteria. We leave as an important avenue for future work the study of rationality criteria for AggCQs (and other types of numerical aggregate queries); this includes understanding whether and how the axiomatic properties advocated by Bienvenu et al. should extend or be adapted to such queries, and developing corresponding attribution measures and algorithms.

Acknowledgments

We are grateful to the PODS referees for providing insightful feedback and contributing excellent suggestions to this paper. This work was supported by the German Research Foundation (DFG) grants GR 1492/16-1 and KI 2348/1-1 (DIP Program). Christoph Standke was funded by the European Union (ERC, SymSim, 101054974) and by the German Research Foundation (GRK 2236, UnRAVeL). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

References

[1] Omer Abramovich, Daniel Deutch, Nave Frost, Ahmet Kara, and Dan Olteanu. 2024. Banzhaf Values for Facts in Query Answering. *Proc. ACM Manag. Data* 2, 3 (2024), 123.
[2] Omer Abramovich, Daniel Deutch, Nave Frost, Ahmet Kara, and Dan Olteanu. 2025. Advancing Fact Attribution for Query Answering: Aggregate Queries and Novel Algorithms. *CoRR* abs/2506.16923 (2025).
[3] Foto N. Afrati and Phokion G. Kolaitis. 2008. Answering aggregate queries in data exchange. In *PODS*. ACM, 129–138.
[4] Antoine Amarilli and Florent Capelli. 2024. Tractable Circuits in Database Theory. *SIGMOD Rec.* 53, 2 (2024), 6–20.
[5] Haris Aziz and Bart de Keijzer. 2014. Shapley meets Shapley. In *STACS (LIPIcs, Vol. 25)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 99–111.
[6] J.F. Banzhaf. 1965. Weighted voting doesn't work: A mathematical analysis. *Rutgers Law Review* 19, 2 (1965), 317–343.
[7] Christoph Berkholz, Jens Keppeler, and Nicole Schweikardt. 2017. Answering Conjunctive Queries under Updates. In *PODS*. ACM, 303–318.
[8] Christoph Berkholz and Maximilian Merz. 2021. Probabilistic Databases under Updates: Boolean Query Evaluation and Ranked Enumeration. In *PODS*. ACM, 402–415.
[9] Leopoldo E. Bertossi and Floris Geerts. 2020. Data Quality and Explainable AI. *ACM J. Data Inf. Qual.* 12, 2 (2020), 11:1–11:9.
[10] Leopoldo E. Bertossi, Benny Kimelfeld, Ester Livshits, and Mikaël Monet. 2023. The Shapley Value in Database Management. *SIGMOD Rec.* 52, 2 (2023), 6–17.
[11] Meghyn Bienvenu, Diego Figueira, and Pierre Lafourcade. 2024. Shapley Value Computation in Ontology-Mediated Query Answering. In *KR*. 156–166.
[12] Meghyn Bienvenu, Diego Figueira, and Pierre Lafourcade. 2024. When is Shapley Value Computation a Matter of Counting? *Proc. ACM Manag. Data* 2, 2 (2024), 105.
[13] Meghyn Bienvenu, Diego Figueira, and Pierre Lafourcade. 2025. Shapley Revisited: Tractable Responsibility Measures for Query Answers. *Proc. ACM Manag. Data* 3, 2 (2025), 112:1–112:26.
[14] Diego Calvanese, Evgeny Kharlamov, Werner Nutt, and Camilo Thorne. 2008. Aggregate queries over ontologies. In *ONISW*. ACM, 97–104.
[15] Nofar Carmeli and Luc Segoufin. 2023. Conjunctive Queries With Self-Joins, Towards a Fine-Grained Enumeration Complexity Analysis. In *PODS*. ACM, 277–289.
[16] Yu-Liang Chou, Catarina Moreira, Peter Bruza, Chun Ouyang, and Joaquim Jorge. 2022. Counterfactuals and causability in explainable artificial intelligence: Theory, algorithms, and applications. *Inf. Fusion* 81 (2022), 59–83.
[17] Nilesh N. Dalvi and Dan Suciu. 2004. Efficient Query Evaluation on Probabilistic Databases. In *VLDB*. Morgan Kaufmann, 864–875.
[18] Nilesh N. Dalvi and Dan Suciu. 2007. Efficient query evaluation on probabilistic databases. *VLDB J.* 16, 4 (2007), 523–544. https://doi.org/10.1007/S00778-006-0004-3
[19] Nilesh N. Dalvi and Dan Suciu. 2012. The dichotomy of probabilistic inference for unions of conjunctive queries. *J. ACM* 59, 6 (2012), 30:1–30:87.

[20] Daniel Deutch, Nave Frost, Benny Kimelfeld, and Mikaël Monet. 2022. Computing the Shapley Value of Facts in Query Answering. In *SIGMOD Conference*. ACM, 1570–1583.

[21] Ronald Fagin, Benny Kimelfeld, and Phokion G. Kolaitis. 2011. Probabilistic data exchange. *J. ACM* 58, 4 (2011), 15:1–15:55.

[22] Robert Fink, Larisa Han, and Dan Olteanu. 2012. Aggregation in Probabilistic Databases via Knowledge Compilation. *Proc. VLDB Endow.* 5, 5 (2012), 490–501.

[23] Robert Fink and Dan Olteanu. 2016. Dichotomies for Queries with Negation in Probabilistic Databases. *ACM Trans. Database Syst.* 41, 1 (2016), 4:1–4:47.

[24] Amir Gilad, Martin Grohe, Benny Kimelfeld, Peter Lindner, and Christoph Standke. 2024. The Importance of Parameters in Database Queries. arXiv:2401.04606 [cs.DB] https://arxiv.org/abs/2401.04606

[25] Boris Glavic, Alexandra Meliou, and Sudeepa Roy. 2021. Trends in Explanations: Understanding and Debugging Data-driven Systems. *Found. Trends Databases* 11, 3 (2021), 226–318.

[26] Erich Grädel, Yuri Gurevich, and Colin Hirsch. 1998. The Complexity of Query Reliability. In *PODS*. ACM, 227–234.

[27] Martin Grohe, Benny Kimelfeld, Peter Lindner, and Christoph Standke. 2024. The Importance of Parameters in Database Queries. In *ICDT (LIPIcs, Vol. 290)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 14:1–14:17.

[28] Loo-Keng Hua. 1982. *Introduction to Number Theory*. Springer-Verlag. xviii+572 pages.

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

[30] Muhammad Idris, Martín Ugarte, and Stijn Vansummeren. 2017. The Dynamic Yannakakis Algorithm: Compact and Efficient Query Processing Under Updates. In *SIGMOD Conference*. ACM, 1259–1274.

[31] Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nick Hynes, Nezihe Merve Gürel, Bo Li, Ce Zhang, Dawn Song, and Costas J. Spanos. 2019. Towards Efficient Data Valuation Based on the Shapley Value. In *AISTATS (Proceedings of Machine Learning Research, Vol. 89)*. PMLR, 1167–1176.

[32] Ahmet Kara, Dan Olteanu, and Dan Suciu. 2024. From Shapley Value to Model Counting and Back. *Proc. ACM Manag. Data* 2, 2 (2024), 79.

[33] Pratik Karmakar, Mikaël Monet, Pierre Senellart, and Stéphane Bressan. 2024. Expected Shapley-Like Scores of Boolean functions: Complexity and Applications to Probabilistic Databases. *Proc. ACM Manag. Data* 2, 2 (2024), 92.

[34] Ester Livshits, Leopoldo E. Bertossi, Benny Kimelfeld, and Moshe Sebag. 2021. Query Games in Databases. *SIGMOD Rec.* 50, 1 (2021), 78–85.

[35] Ester Livshits, Leopoldo E. Bertossi, Benny Kimelfeld, and Moshe Sebag. 2021. The Shapley Value of Tuples in Query Answering. *Log. Methods Comput. Sci.* 17, 3 (2021).

[36] Ester Livshits and Benny Kimelfeld. 2022. The Shapley Value of Inconsistency Measures for Functional Dependencies. *Log. Methods Comput. Sci.* 18, 2 (2022).

[37] Scott M Lundberg, Gabriel Erion, Hugh Chen, Alex DeGrave, Jordan M Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee. 2020. From local explanations to global understanding with explainable AI for trees. *Nature machine intelligence* 2, 1 (2020), 2522–5839.

[38] Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Advances in neural information processing systems*. 4765–4774.

[39] Xuan Luo and Jian Pei. 2024. Applications and Computation of the Shapley Value in Databases and Machine Learning. In *SIGMOD Conference Companion*. ACM, 630–635.

[40] Alexandra Meliou, Wolfgang Gatterbauer, Katherine F. Moore, and Dan Suciu. 2010. The Complexity of Causality and Responsibility for Query Answers and non-Answers. *Proc. VLDB Endow.* 4, 1 (2010), 34–45.

[41] Alexandra Meliou, Sudeepa Roy, and Dan Suciu. 2014. Causality and explanations in databases. *Proceedings of the VLDB Endowment (PVLDB)* 7, 13 (2014), 1715–1716.

[42] Thomas Muñoz, Cristian Riveros, and Stijn Vansummeren. 2024. Enumeration and Updates for Conjunctive Linear Algebra Queries Through Expressibility. In *ICDT (LIPIcs, Vol. 290)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 12:1–12:20.

[43] Dan Olteanu and Jiewen Huang. 2008. Using OBDDs for Efficient Query Evaluation on Probabilistic Databases. In *SUM (Lecture Notes in Computer Science, Vol. 5291)*. Springer, 326–340.

[44] Anantha Padmanabha, Luc Segoufin, and Cristina Sirangelo. 2024. A Dichotomy in the Complexity of Consistent Query Answering for Two Atom Queries With Self-Join. *Proc. ACM Manag. Data* 2, 2 (2024), 74.

[45] Romila Pradhan, Aditya Lahiri, Sainyam Galhotra, and Babak Salimi. 2022. Explainable AI: Foundations, Applications, Opportunities for Data Management Research. In *SIGMOD Conference*. ACM, 2452–2457.

[46] J. Scott Provan and Michael O. Ball. 1983. The Complexity of Counting Cuts and of Computing the Probability that a Graph is Connected. *SIAM J. Comput.* 12, 4 (1983), 777–788.

[47] Alon Reshef, Benny Kimelfeld, and Ester Livshits. 2020. The Impact of Negation on the Complexity of the Shapley Value in Conjunctive Queries. In *PODS*. ACM, 285–297.

[48] Alvin E Roth. 1988. *The Shapley value: essays in honor of Lloyd S. Shapley.* Cambridge University Press.
[49] Benedek Rozemberczki, Lauren Watson, Péter Bayer, Hao-Tsung Yang, Oliver Kiss, Sebastian Nilsson, and Rik Sarkar. 2022. The Shapley Value in Machine Learning. In *IJCAI*. ijcai.org, 5572–5579.
[50] Babak Salimi, Leopoldo E. Bertossi, Dan Suciu, and Guy Van den Broeck. 2016. Quantifying Causal Effects on Query Answering in Databases. In *TaPP*. USENIX Association.
[51] Lloyd S Shapley. 1953. A Value for n-Person Games. In *Contributions to the Theory of Games II*, Harold W. Kuhn and Albert W. Tucker (Eds.). Princeton University Press, Princeton, 307–317. https://doi.org/10.1515/9781400829156-012
[52] Christoph Standke and Benny Kimelfeld. 2025. Tractability Frontiers of the Shapley Value for Aggregate Conjunctive Queries. arXiv:2509.13565 [cs.DB] https://arxiv.org/abs/2509.13565
[53] Leslie G. Valiant. 1979. The Complexity of Computing the Permanent. *Theor. Comput. Sci.* 8 (1979), 189–201. https://doi.org/10.1016/0304-3975(79)90044-6
[54] Qichen Wang, Xiao Hu, Binyang Dai, and Ke Yi. 2023. Change Propagation Without Joins. *Proc. VLDB Endow.* 16, 5 (2023), 1046–1058.
[55] Feiyu Xu, Hans Uszkoreit, Yangzhou Du, Wei Fan, Dongyan Zhao, and Jun Zhu. 2019. Explainable AI: A Brief Survey on History, Research Areas, Approaches and Challenges. In *NLPCC (2) (Lecture Notes in Computer Science, Vol. 11839)*. Springer, 563–574.

Received June 2025; accepted August 2025