The Importance of Parameters in Database Queries

Martin Grohe
RWTH Aachen University, Germany

Benny Kimelfeld
Technion – Israel Institute of Technology, Haifa, Israel

Peter Lindner
École Polytechnique Fédérale de Lausanne, Switzerland

Christoph Standke
RWTH Aachen University, Germany

Abstract

We propose and study a framework for quantifying the importance of the choices of parameter values to the result of a query over a database. These parameters occur as constants in logical queries, such as conjunctive queries. In our framework, the importance of a parameter is its SHAP score. This score is a popular instantiation of the game-theoretic Shapley value to measuring the importance of feature values in machine learning models. We make the case for the rationale of using this score by explaining the intuition behind SHAP, and by showing that we arrive at this score in two different, apparently opposing, approaches to quantifying the contribution of a parameter.

The application of the SHAP score requires two components in addition to the query and the database: (a) a probability distribution over the combinations of parameter values, and (b) a utility function that measures the similarity between the result for the original parameters and the result for hypothetical parameters. The main question addressed in the paper is the complexity of calculating the SHAP score for different distributions and similarity measures. We first address the case of probabilistically independent parameters. The problem is hard if we consider a fragment of queries that is hard to evaluate (as one would expect), and even for the fragment of acyclic conjunctive queries. In some cases, though, one can efficiently list all relevant parameter combinations, and then the SHAP score can be computed in polynomial time under reasonable general conditions. Also tractable is the case of full acyclic conjunctive queries for certain (natural) similarity functions. We extend our results to conjunctive queries with inequalities between variables and parameters. Finally, we discuss a simple approximation technique for the case of correlated parameters.

2012 ACM Subject Classification Theory of computation → Database query languages (principles)

Keywords and phrases SHAP score, query parameters, Shapley value

Digital Object Identifier 10.4230/LIPIcs.ICDT.2024.14

Related Version *Full Version*: `http://arxiv.org/abs/2401.04606` [11]

Funding The work of Martin Grohe, Benny Kimelfeld, and Christoph Standke was supported by the German Research Foundation (DFG) grants GR 1492/16-1 and KI 2348/1-1 (DIP Program). The work of Martin Grohe and Christoph Standke was supported by the German Research Foundation (DFG) grant GRK 2236/2 (UnRAVeL).

1 Introduction

The parameters of a database query may affect the result in a way that misrepresents the importance of the parameters, or the arbitrariness in their chosen values. For example, when searching for products in commercial applications (for clothing, travel, real estate, etc.), we may fill out a complex form of parameters that produce too few answers or overly expensive ones; what is the responsibility of our input values to this deficient outcome? We may phrase a database query to select candidates for awards for job interviews; to what extent is the choice of parameters affecting people's fate?

Considerable effort has been invested in exploring the impact of parameters on query outcomes. In the *empty-answer problem*, the goal is typically to explore a space of small changes to the query that would yield a nonempty result [14, 22]. In that vein, reasoning about small parameter changes, or *perturbations*, has been applied to providing explanations to *non-answers*, that is, tuples that are missing from the result [5, 31]. From a different angle, the analysis of sensitivity to parameters has been applied to *fact checking*, and particularly, the detection of statements that are *cherry picked* in the sense that they lead to conclusions that overly rely on allegedly arbitrary parameter values. This may come in over-restriction to a database fragment that serves the intended claim [35], or over-generalization that masks the situation in substantial subgroups that oppose the claim [16].

In this work, we aim to establish a principled quantitative measure for the importance of individual parameter values to the result $Q(D)$ of a query Q over a database D. To this end, we begin with the basic idea of observing how the result changes when we randomly change the parameter of interest. Alternatively, we can observe the change in the result when the parameter keeps its value while all others randomly change. Yet, these definitions ignore dependencies among parameters; changing a parameter may have no impact in the presence of other parameter values (e.g., the number of connecting flights does not matter if we restrict the travel duration), or it may lead to an overestimation of the value's importance (e.g., changing the number of semesters empties the result since we restrict the admission year). This can be viewed as a special case of a challenge that has been studied for decades in game theory: *How to attribute individual contributions to the gains of a team?* Specifically, we can view the parameter values as players of a cooperative game where each coalition (set of parameter values in our case) has a utility, and we wish to quantify the contribution of each parameter value to the overall utility. We then adopt a conventional formula for contribution attribution, namely the Shapley value [29], as done in many domains, including economics, law, bioinformatics, crime analysis, network analysis, machine learning, and more (see, e.g., the *Handbook of the Shapley value* [1]). The Shapley value is theoretically justified in the sense that it is unique under several elementary axioms of rationality for profit sharing [29]. In the context of databases, this value has been studied recently for measuring the contribution of individual tuples to query answers [17, 8, 2] and to database inconsistency [18], as well as the contribution of constraints to the decisions of cleaning systems [7].

Our challenge then boils down to one central question: What game are we playing? In other words, what is the utility of a set J of parameter values? Following up on the two basic ideas discussed above, we can think of two analogous ways. In the first way, we measure the change in the query result when we randomly change the values of the parameters in J; the parameter values in J are deemed important if we observe a large change. This change is random, so we take the *expected* change. (We later discuss the way that we measure the change in the result.) In the second way, we again measure the change in the result, but now we do so when we fix the values of the parameters in J and randomly change the rest; now, however, the values in J are deemed important if we observe a *small* change, indicating that the other parameters have little impact once we use the values of J. This second way is known as the Shap score [20, 19] in machine learning, and it is one of the prominent score-attribution methods for features (in addition to alternatives such as LIME [25] and Anchor [26]).[1] This score quantifies the impact of each feature value on the decision for a specific given instance. Interestingly, we show that the first way described above also coincides

[1] For background on attribution scores see textbooks on explanations in machine learning, e.g., Molnar [21].

with the Shap score, so the two ways actually define the *same measure* (Theorem 3.4). We prove it in a general setting and, hence, this equivalence is of independent interest as it shows an alternative, apparently different way of arriving at Shap in machine learning.

To materialize the framework in the context of a query Q and a database D, one needs to provide some necessary mechanisms for reasoning about $Q(D)$ and $Q'(D)$, where Q' is the same as Q up to the parameters: it uses the same values as Q for the parameters of J, but the remaining parameter values are selected randomly. Specifically, to this aim, we need two mechanisms:

1. A *probability distribution* Γ of possible parameterizations of the query;
2. A *similarity function* $\mathfrak{s}$ between relations to quantify how close $Q'(J)$ is to $Q(J)$.

The distribution Γ may include any feasible combination of parameter values, and they can be either probabilistically independent or correlated. For $\mathfrak{s}$, one can use any similarity between sets (see, e.g., surveys on similarity measures such as [15, 28]) or measures that account for the distance between attributes values (e.g., as done in the context of database repairs [4]). We give examples in Section 3.

A central challenge in the framework is the computational complexity, since the direct definition of the Shap score (like the general Shapley value) involves summation over an exponential space of coalitions. Indeed, the calculation can be a hard computational problem, #P-hard to be precise, even for simple adaptations of the Shap score [3] and the Shapley value [10, 6, 17]. Hence, instantiations of our framework require specialized complexity analyses and nontrivial algorithms that bypass the exponential time of the naïve computation.

We begin the complexity analysis of the framework by establishing some general insights for finite fully factorized distributions, i.e., where the parameters are probabilistically independent and each is given as an explicit collection of value-probability pairs. First, the Shap score can be computed in polynomial time if we can evaluate the query, compute the similarity measure, and enumerate all parameter combinations in polynomial time. We prove this using a recent general result by Van den Broeck et al. [32] (from which we borrow some notation) showing that, under tractability assumptions, the Shap score is reducible to the computation of the expected value under random parameter values. Second, under reasonable assumptions, the computation of the Shap score is at least as hard as testing for the emptiness of the query, for *every* nontrivial similarity function; this is expected, as the definition of the Shap scores requires, conceptually, many applications of the query.

Next, we focus on the class of conjunctive queries, where the parameters are constants in query atoms. Put differently, we consider Select-Project-Join queries where each selection predicate has the form $x = p$ where x is an attribute and p is a parameter. It follows from the above general results that this case is tractable under data complexity. Hence, we focus on combined complexity. As the emptiness problem is intractable, we consider the tractable fragment of acyclic queries, and show that Shap scores can be #P-hard even there.

We then focus on the class of *full* acyclic queries and establish that the Shap score can be computed in polynomial time for three natural, set-based similarity functions between $Q(D)$ and $Q'(D)$. Interestingly, this gives us nontrivial cases where the Shap score can be computed in polynomial time even if $Q(D)$ and $Q'(D)$ can be exponential in the size of the input, and hence, it is intractable to materialize them.

We then extend our results to conjunctive queries with inequalities, that is, built-in atoms of the form $x \leq p$ where x is a variable and p is a parameter. We show that this addition can make the Shap score intractable to compute, even if there is a single relational atom in addition to the inequality atoms. Nevertheless, we identify cases in which tractability properties are retained when adding inequalities to classes of parameterized queries (e.g., full acyclic conjunctive queries), relying on structural assumptions on the use of inequalities.

Given that the computation of the exact SHAP score is often intractable, we also study the complexity of approximate evaluation. We show that using sampling, we can obtain an efficient approximation scheme (FPRAS) with additive guarantees. Moreover, the tractability of approximation generalizes to allow for parameters that are correlated through Bayesian networks (and actually any distribution) that provide(s) polynomial-time sampling while conditioning on assignments to arbitrary subsets of the random variables.

Details omitted due to space limitations, including the full proofs of the proof sketches, can be found in the full version of the paper [11].

2 Preliminaries

We write 2^S for the power set of S. Vectors, tuples and sequences are denoted by boldface letters. If $\boldsymbol{x} = (x_i)_{i\in I}$ and $J \subseteq I$, then $\boldsymbol{x}_J = (x_j)_{j\in J}$. Moreover, $|\boldsymbol{x}| = |I|$ is the number of entries of $\boldsymbol{x}$. We let $[n] = \{1, \dots, n\}$. Truth values (true/false) are denoted by tt and ff.

A (discrete) *probability distribution* is a function $\Gamma \colon S \to [0, 1]$, where S is a non-empty countable set, and $\sum_{s\in S} \Gamma(s) = 1$. The *support* of Γ is $\mathsf{supp}(\Gamma) = \{s \in S \colon \Gamma(s) > 0\}$. We use $\Pr$ for generic probability distributions and, in particular, $\Pr_{X\sim\Gamma}$ to refer to probabilities for a random variable or random vector X being drawn from the distribution Γ. Likewise, $\mathbb{E}_{X\sim\Gamma}$ refers to the expectation operator with respect to the distribution Γ of X.

Whenever x is an input to a computational problem, we write $\|x\|$ to refer to the encoding length of x as it is represented in the input.

2.1 Schemas and Databases

A *relation schema* is a sequence $\boldsymbol{A} = (A_1, \dots, A_k)$ of distinct *attributes* A_i, each with an associated *domain* $\mathsf{dom}(A_i)$ of values. We call k the *arity* of $\boldsymbol{A}$. If $\boldsymbol{A} = (A_1, \dots, A_k)$ is a relation schema and $S = \{i_1, \dots, i_\ell\} \subseteq [k]$ with $i_1 < i_2 < \dots < i_\ell$, then $\boldsymbol{A}_S$ denotes the relation schema $(A_{i_1}, \dots, A_{i_\ell})$.

A *tuple* over $\boldsymbol{A}$ is an element $\boldsymbol{a} = (a_1, \dots, a_k) \in \mathsf{dom}(A_1) \times \dots \times \mathsf{dom}(A_k)$. The set of tuples over $\boldsymbol{A}$ is denoted by $\mathrm{Tup}[\boldsymbol{A}]$. A *relation* over $\boldsymbol{A}$ is a finite set of tuples over $\boldsymbol{A}$. We denote the space of relations over $\boldsymbol{A}$ by $\mathrm{Rel}[\boldsymbol{A}]$. We typically denote relations by T or T_1, T_2, and so on. *By default, we assume that all attribute domains are countably infinite.*

A *database schema* is a finite set of *relation symbols*, where every relation symbol is associated with a relation schema. For example, if R is a relation symbol with associated relation schema $\boldsymbol{A} = (A_1, \dots, A_k)$, we may refer to R by $R(\boldsymbol{A})$ or $R(A_1, \dots, A_k)$, and call k the *arity* of R. A *fact* f over a database schema $\boldsymbol{S}$ is an expression of the shape $R(\boldsymbol{a})$ with $R = R(\boldsymbol{A})$, where $\boldsymbol{a}$ is a tuple over $\boldsymbol{A}$. A *database* over schema $\boldsymbol{S}$ is a finite set of facts over $\boldsymbol{S}$. We denote the space of databases over some schema $\boldsymbol{S}$ by $\mathrm{DB}[\boldsymbol{S}]$.

2.2 Queries and Parameters

A *relational query* (or just *query*) is an isomorphism invariant function that maps databases to relations. More precisely, a query q has a database schema $\mathsf{dom}(q)$ for its valid input databases (called the *domain schema*), and a relation schema $\mathsf{range}(q)$ for its output relation (called the *range schema*). Then q is a function that maps databases over $\mathsf{dom}(q)$ to relations over $\mathsf{range}(q)$. We write $q = q(\boldsymbol{R})$ to denote that $\mathsf{range}(q) = \boldsymbol{R}$.

We focus on queries that are expressed in variants of first-order logic without equality. In this case, $\mathsf{range}(q)$ is the domain of the free variables of q. *Whenever we discuss queries in this paper, it is assumed that they are first-order queries, unless explicitly stated* *otherwise.* To emphasize that q has free variables $\boldsymbol{x} = (x_1, \dots, x_n)$, we write $q = q(\boldsymbol{x})$. If $\boldsymbol{a} = (a_1, \dots, a_n) \in \mathrm{Tup}[\boldsymbol{R}]$, then $q(\boldsymbol{a})$ denotes the Boolean query obtained from q by substituting every occurrence of x_i with a_i, $i \in [n]$. If D is a database over $\mathsf{dom}(q)$, then $q(D) \coloneqq \{\boldsymbol{a} \in \mathrm{Tup}[\boldsymbol{R}] \colon D \models q(\boldsymbol{a})\}$.

A *parameterized query* Q is a relational query in which some of the free variables are distinguished *parameters*. We write $Q(\boldsymbol{x}; \boldsymbol{y})$ to denote that $\boldsymbol{x}$ are the non-parameter (free) variables of Q, and that $\boldsymbol{y}$ are the parameters of Q. A parameterized query Q has a *parameter schema* $\mathsf{param}(Q)$, such that $\mathrm{Tup}[\mathsf{param}(Q)]$ is the set of valid tuples of parameter values for Q. If $Q = Q(\boldsymbol{x}; \boldsymbol{y})$ is a parameterized query, and $\boldsymbol{p} \in \mathrm{Tup}[\mathsf{param}(Q)]$, then $Q_{\boldsymbol{p}}(\boldsymbol{x}) \coloneqq Q(\boldsymbol{x}; \boldsymbol{p})$ is the relational query obtained from Q by substituting the parameters $\boldsymbol{y}$ with the constants $\boldsymbol{p}$. The *domain* and *range schemas* of Q coincide with $\mathsf{dom}(Q_{\boldsymbol{p}})$ and $\mathsf{range}(Q_{\boldsymbol{p}})$, respectively, which is independent of the choice of $\boldsymbol{p} \in \mathrm{Tup}[\mathsf{param}(Q)]$. We write $Q = Q(\boldsymbol{R}; \boldsymbol{P})$ to denote that $\mathsf{range}(Q) = \boldsymbol{R}$ and $\mathsf{param}(Q) = \boldsymbol{P}$. If $Q = Q(\boldsymbol{R}; \boldsymbol{P})$, then for every database D over $\mathsf{dom}(Q)$ and every $\boldsymbol{p} \in \mathrm{Tup}[\boldsymbol{P}]$ we have

$$Q_{\boldsymbol{p}}(D) \coloneqq \big\{\boldsymbol{a} \in \mathrm{Tup}[\boldsymbol{R}] \colon D \models Q_{\boldsymbol{p}}(\boldsymbol{a})\big\} \in \mathrm{Rel}[\boldsymbol{R}].$$

▶ **Example 2.1.** For illustration, we consider a parameterized query over flights data, assuming access to a database with relations $\mathsf{Flight}(\mathsf{Id}, \mathsf{Date}, \mathsf{AirlineName}, \mathsf{From}, \mathsf{To}, \mathsf{Departure}, \mathsf{Arrival})$ and $\mathsf{Airline}(\mathsf{Name}, \mathsf{CountryOfOrigin})$. Then

$$Q(f, t_{\mathsf{dep}}, t_{\mathsf{arr}}; d, c) = \exists a \colon \mathsf{Flights}(f, d, a, \mathtt{CDG}, \mathtt{JFK}, t_{\mathsf{dep}}, t_{\mathsf{arr}}) \wedge \mathsf{Airline}(a, c)$$

is a parameterized query. For parameters (d, c) it asks for departure and arrival times of flights from Paris to New York City on date d operated by an airline from country c. ⌟

Parameterized queries inherit structural properties from the underlying relational query. For example, if $Q(\boldsymbol{x}; \boldsymbol{y})$ is a parameterized query with free variables $\boldsymbol{x}$, then Q is called an (acyclic) conjunctive query, if the relational query $Q(\boldsymbol{x}, \boldsymbol{y})$ with variables $\boldsymbol{x}$, $\boldsymbol{y}$ is an (acyclic) conjunctive query. We emphasize that for the notion of acyclicity, the parameters act as additional free variables. However, a parameterized query Q is called *Boolean*, if all (non-parameter) variables in Q are quantified, i.e., if $Q_{\boldsymbol{p}}$ is Boolean for all $\boldsymbol{p}$.

2.3 The Shapley Value

A *cooperative game* is a pair (I, ν) where I is a non-empty set of players, and $\nu \colon 2^I \to \mathbb{R}$ is a *utility function* that assigns a "joint wealth" $\nu(J)$ to every *coalition* $\nu(J) \subseteq I$. The *Shapley value* of player $i \in I$ in (I, ν) is defined as

$$\mathrm{Shapley}(I, \nu, i) \coloneqq \frac{1}{|I|!} \sum_{\sigma \in S_I} \big(\nu(\sigma_i \cup \{i\}) - \nu(\sigma_i)\big) = \mathop{\mathbb{E}}_{\sigma \sim S_I} \big[\nu(\sigma_i \cup \{i\}) - \nu(\sigma_i)\big], \tag{1}$$

where S_I is the set of permutations of I, and σ_i is the set of players appearing before i in the permutation $\sigma \in S_I$ (and $\sigma \sim S_I$ refers to drawing from the uniform distribution on S_I). Intuitively, $\mathrm{Shapley}(I, \nu, i)$ measures the importance (or, contribution) of i among coalitions. An equivalent, alternative expression is

$$\mathrm{Shapley}(I, \nu, i) = \sum_{J \subseteq I \setminus \{i\}} \Pi_i(J) \cdot \big(\nu(J \cup \{i\}) - \nu(J)\big) = \mathop{\mathbb{E}}_{J \sim \Pi_i} \big[\nu(J \cup \{i\}) - \nu(J)\big], \tag{2}$$

where Π_i is the probability distribution on $I \setminus \{i\}$ with

$$\Pi_i(J) = \frac{|J|! \cdot (|I| - |J| - 1)!}{|I|!} = \frac{1}{|I| \cdot \binom{|I|-1}{|J|}}. \tag{3}$$

(For further details, see textbooks on the Shapley value, e.g., [1].) In the next section, we will discuss the intuition behind using the Shapley value in the context of our paper.

3 The SHAP Score of Query Parameters

In this section, we shall give our exact definition of the SHAP score of a query parameter, and we will argue that it measures the contribution of the parameter to the outcome of the query in a natural way. We start with a database D over $\mathsf{dom}(Q)$; a parameterized query $Q(\boldsymbol{R};\boldsymbol{P})$; and a *reference parameter* $\boldsymbol{p}^* = (p_1^*, \ldots, p_\ell^*)$ over $\boldsymbol{P}$. We want to quantify the contribution of each parameter p_i^* to the query answer $Q_{\boldsymbol{p}}(D)$.

3.1 Intuition

A basic idea is to see what happens to the answer if we either modify p_i^* and keep all other parameters fixed or, conversely, keep p_i^* fixed but modify the other parameters. The following example shows that neither of these two approaches is sufficient.

▶ **Example 3.1.** We let $Q(x; y_1, y_2, y_3) \coloneqq R(y_1, y_2, y_3, x)$ for a relation $R(B_1, B_2, B_3, A)$. Let $N \coloneqq [n] = \{1, \ldots, n\}$, and D be the database where the tuple set of R is

$$\big((\{1\} \times N) \cup (N \times \{1\})\big) \times N \times N.$$

Let $\boldsymbol{p}^* = (p_1^*, p_2^*, p_3^*) = (1, 1, 1)$. Then $Q_{\boldsymbol{p}^*}(D) = N$.

Let us try to understand the contribution of each parameter p_i^*. Intuitively, the parameter p_3^* is completely irrelevant. Whatever value it takes in any configuration of the other two parameters, the query answer remains the same. Parameters p_1^* and p_2^* are important, though. If we change both of their values to values $p_1, p_2 \in N \setminus \{1\}$, the query answer becomes empty. However, for this change to take effect, *we need to change both values at the same time.* If we change only p_1 keeping p_2^* and p_3^* fixed, the query answer does not change. If we keep p_1^* fixed and modify p_2 and p_3, then again the query answer does not change. The same holds for the second parameter. ⌟

What this example shows is that even if we just want to understand the contribution of the individual parameters, we need to look at the contributions of *sets* of parameters. This will immediately put us in the realm of *cooperative game theory* with the parameters acting as players in a coalitional game.

Assume for a moment that we have a function $\nu : 2^{[\ell]} \to \mathbb{R}$ such that, for $J \subseteq [\ell]$, the function value $\nu(J)$ quantifies the combined contribution of the parameters with indices in J. We will discuss later how to obtain such a function ν.

Given ν, we now quantify the contribution of every *individual* parameter $j \in [\ell]$. This is exactly what the Shapley value $\mathrm{Shapley}([\ell], \nu, j)$ does. And not only that: as Shapley [29] proved, subject to a few natural assumptions ("axioms") and the technical requirement that ν is zero on the empty set, it is the *only* way of doing this. Let us briefly review these axioms (for details and a proof of Shapley's theorem, we refer to the literature, e.g., [29, 27, 1]). The first axiom, *symmetry*, says that if any two parameters i and j (or rather parameter indices) have the same contribution to any coalition $J \subseteq [\ell] \setminus \{i, j\}$, then $\mathrm{Shapley}([\ell], \nu, i) = \mathrm{Shapley}([\ell], \nu, j)$. The second axiom, *efficiency*, says that the sum of $\mathrm{Shapley}([\ell], \nu, j)$ for all parameters $j \in [\ell]$ equals the contribution of the "grand coalition" $[\ell]$. Thus, the Shapley values "distribute" $\nu([\ell])$ among the individual parameters. The third axiom, *null player*, says that if a parameter makes no contribution to any coalition, then $\mathrm{Shapley}([\ell], \nu, j) = 0$. The fourth axiom, *additivity*, says that if we have two different functions $\nu_1, \nu_2 : 2^{[\ell]} \to \mathbb{R}$ measuring the value of a coalition of parameters, then $\mathrm{Shapley}([\ell], \nu_1 + \nu_2, j) = \mathrm{Shapley}([\ell], \nu_1, j) + \mathrm{Shapley}([\ell], \nu_2, j)$. All of the axioms strike us as completely natural also for evaluating the contribution of parameters to a query answer.

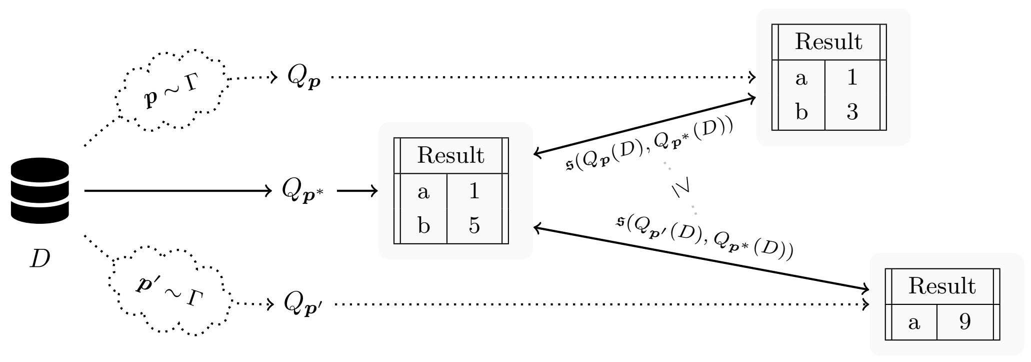

Figure 1 On a given database D, we evaluate a parameterized query Q with respect to a reference parameter $\boldsymbol{p}^*$. The result is compared (using a similarity function $\mathfrak{s}$) against different parameterizations of Q, according to a probabilistic model Γ.

Concluding the discussion so far, to understand the contribution of individual parameters, we need to look at sets of parameters, and the method of choice to retrieve this contribution from a valuation for sets is the Shapley value.

3.2 The Utility Function

What remains to be done is constructing a natural valuation for sets of parameters, that is, a function $\nu : 2^{[\ell]} \to \mathbb{R}$ such that, for $J \subseteq [\ell]$ the value $\nu(J)$ quantifies the combined contribution of the parameters p_j^*, for $j \in J$, to the query answer. The first thing we need is a way of measuring the similarity between query answers, that is, a function $\mathfrak{s}$ that takes two relations of the schema of our query answer and assigns a similarity value to them. Formally, a *similarity function* is just an arbitrary function $\mathfrak{s} : \mathrm{DB}[\boldsymbol{R}] \times \mathrm{DB}[\boldsymbol{R}] \to \mathbb{R}$. We make no restrictions on this function, similarity can have different meanings depending on the application. Let us just note that the intended use of this function is always to quantify similarity: the higher $\mathfrak{s}(Q, Q^*)$, the more similar Q and Q^* are.

▶ **Example 3.2.**

(a) Prime examples of similarity functions are set-similarity measures such as the *Jaccard index*, the *Sørensen index*, and *Tverski's index*, which attempt to capture the degree of similarity between two sets [23]. If Q and Q^* are two relations of the same relation schema $\boldsymbol{R}$, then, e.g., their Jaccard index is given by

$$\texttt{Jaccard}(Q, Q^*) := \frac{|Q \cap Q^*|}{|Q \cup Q^*|} = 1 - \frac{|Q \triangle Q^*|}{|Q \cup Q^*|},$$

and $\texttt{Jaccard}(\emptyset, \emptyset) = 0$ by convention.

(b) In the same spirit, but somewhat simpler, are similarities based on cardinalities and basic set operations, for example:

$$\begin{aligned} \texttt{Int}(Q, Q^*) &:= |Q \cap Q^*|, && \text{(size of the interesection)} \\ \texttt{NegSymDiff}(Q, Q^*) &:= -|Q \triangle Q^*|, && \text{(negative symmetric difference)} \\ \texttt{NegSymCDiff}(Q, Q^*) &:= -\big||Q| - |Q^*|\big|. && \text{(negative symmetric cardinality difference)} \end{aligned}$$

Note that we do not require similarity to be non-negative, thus `NegSymDiff` and `NegSymCDiff` are well-defined. Intuitively, the negative sign makes sense because two relations get more similar as their symmetric difference gets smaller.

(c) In our applications, Q and Q^* have different roles: Q^* is always a fixed "reference relation", and Q is a modification of Q^*, obtained by changing some parameters in the query. In some situations, it can be useful to use similarity measures that are not symmetric. For example, we may only care about not losing any tuples from Q^*, whereas additional tuples may be irrelevant. This leads to an asymmetric similarity measure:

$$\texttt{NegDiff}(Q, Q^*) \coloneqq -|Q^* \setminus Q|. \qquad \text{(negative difference)}$$

We can also use similarity functions that are specialized to the application at hand. For example, we can use a similarity function that accounts for differences of attribute values, such as arrival time minus departure time:

$$\texttt{MinDiff}_{A,B}(Q, Q^*) \coloneqq -\big|\min(Q.A - Q.B) - \min(Q^*.A - Q^*.B)\big|.$$

where $Q.A$ is the vector obtained from numerical A column of Q (hence, $Q.A - Q.B$ is the vector of differences between the A and B attributes). Alternatively, we can use

$$\texttt{ExpMinDiff}_{A,B}(Q, Q^*) \coloneqq \exp\big(\min(Q.A - Q.B) - \min(Q^*.A - Q^*.B)\big).$$

which uses the exponent function instead of the absolute value. ⌟

Suppose now that we have a subset $J \subseteq [\ell]$. We need a value $\nu(J)$ quantifying how important the subtuple $\boldsymbol{p}^*_J = (p_j)_{j \in J}$ is to obtain a query answer similar to $Q_{\boldsymbol{p}^*}(D)$. The crucial idea of the Shap score in a very similar setting in machine learning [20] was to see what happens if we fix the parameters $\boldsymbol{p}^*_J$ and change the other parameters: the value of J is high if changing the other parameters has no big effect on the query answer. In other words, the value of J is large if for parameter tuples $\boldsymbol{p}$ with $\boldsymbol{p}_J = \boldsymbol{p}^*_J$ the relations $Q_{\boldsymbol{p}}(D)$ and $Q_{\boldsymbol{p}^*}(D)$ are similar, that is, $\mathfrak{s}(\boldsymbol{p}, \boldsymbol{p}^*)$ is large.[2] But of course this depends on how exactly we change the other parameters. The idea is to change them randomly and take the expected value, as illustrated in Figure 1. So we assume that we have a probability distribution Γ on the space of all possible parameter values, and we define

$$\nu(J) \coloneqq \mathop{\mathbb{E}}_{\boldsymbol{p} \sim \Gamma}\big[\mathfrak{s}(\boldsymbol{p}, \boldsymbol{p}^*) \bigm| \boldsymbol{p}_J = \boldsymbol{p}^*_J\big] = \mathop{\mathbb{E}}_{\boldsymbol{p} \sim \Gamma}\big[\mathfrak{s}(Q_{\boldsymbol{p}}(D), Q_{\boldsymbol{p}^*}(D)) \bigm| \boldsymbol{p}_J = \boldsymbol{p}^*_J\big].$$

▶ Remark 3.3. Officially, we should "normalize" to $\nu(\emptyset) = 0$ by subtracting $\mathbb{E}_{\boldsymbol{p}}\big[\mathfrak{s}(\boldsymbol{p}, \boldsymbol{p}^*)\big]$. We chose not to do so, because it leads to the same Shapley values due to linearity. ⌟

With the above, we can quantify the contribution of any individual parameter i as Shapley $\big([\ell], \nu, i\big)$. In using different probability distributions, we can incorporate various assumptions about the usage of parameters. For example, Γ may apply to only a subset of interesting parameters and be deterministic (fixed) on the others; it can also be concentrated on certain values (e.g., locations of a reasonable distance to the original value) or describe any other statistical model on the parameter space.

There is a different, in some way complementary approach to quantifying the contribution of a set J of parameters: instead of measuring what happens if we fix $\boldsymbol{p}^*_J$ and randomly change the other parameters, we can also fix the other parameters and randomly change the parameters in J. Then the contribution of J is the higher, the more the query answer changes. Thus, now instead of our similarity measure $\mathfrak{s}$ we need a *dissimilarity measure* $\bar{\mathfrak{s}}$. We simply let $\bar{\mathfrak{s}}(\boldsymbol{p}, \boldsymbol{p}') \coloneqq c - \mathfrak{s}(\boldsymbol{p}, \boldsymbol{p}')$, where $c \in \mathbb{R}$ is an arbitrary constant. (We will see that this

[2] We use $\mathfrak{s}(\boldsymbol{p}, \boldsymbol{p}^*)$ as shorthand for $\mathfrak{s}(Q_{\boldsymbol{p}}(D), Q_{\boldsymbol{p}^*}(D))$ in case Q and D are clear from the context.

constant is completely irrelevant, but it may have some intuitive meaning. For example, if our similarity measure is normalized to take values in the interval $[0,1]$, as Jaccard similarity, then it would be natural to take $c = 1$.) Letting $\overline{J} := [\ell] \setminus J$ for every $J \subseteq [\ell]$, our new value functions for sets of indices is

$$\overline{\nu}(J) := \mathop{\mathbb{E}}_{\boldsymbol{p}\sim\Gamma}\left[\overline{\mathfrak{s}}(\boldsymbol{p},\boldsymbol{p}^*) \,\middle|\, \boldsymbol{p}_{\overline{J}} = \boldsymbol{p}^*_{\overline{J}}\right].$$

We could have started our treatment with introducing $\overline{\nu}$ instead of ν; indeed, the only reason that we did not is that the analogue of ν is what is used in machine learning. Both ν and $\overline{\nu}$ strike us as completely natural value functions for sets of parameters, and we see no justification for preferring one over the other. Fortunately, we do not have to, because they both lead to exactly the same Shapley values for the individual parameters.

▶ **Theorem 3.4.** *For all $i \in [\ell]$ we have* $\mathrm{Shapley}\big([\ell],\nu,i\big) = \mathrm{Shapley}\big([\ell],\overline{\nu},i\big)$.

Proof sketch. Linearity of expectation entails $\nu(J) = c - \overline{\nu}(\overline{J})$. The main observation is then on the relationship between J and $\overline{J \cup \{i\}}$, for $J \subseteq [\ell] \setminus \{i\}$: Both sets have the same probability under Π_i and, moreover, there is a one-to-one correspondence between them. Changing the summation accordingly shows that the SHAP scores coincide. ◀

When $Q, D, \boldsymbol{p}^*, \mathfrak{s}, \Gamma$ are clear from the context, we write $\textsc{Shap}(i) := \mathrm{Shapley}([\ell],\nu,i)$; hence:

$$\textsc{Shap}(i) = \mathop{\mathbb{E}}_{J\sim\Pi_i}\left[\mathop{\mathbb{E}}_{\boldsymbol{p}\sim\Gamma}\left[\mathfrak{s}(Q_{\boldsymbol{p}}(D),Q_{\boldsymbol{p}^*}(D)) \,\middle|\, \boldsymbol{p}_{J\cup\{i\}} = \boldsymbol{p}^*_{J\cup\{i\}}\right] - \mathop{\mathbb{E}}_{\boldsymbol{p}\sim\Gamma}\left[\mathfrak{s}(Q_{\boldsymbol{p}}(D),Q_{\boldsymbol{p}^*}(D)) \,\middle|\, \boldsymbol{p}_J = \boldsymbol{p}^*_J\right]\right].$$

We illustrate the framework using our running example from Section 2.

▶ **Example 3.5.** In Example 2.1, we considered a flights database D and a query Q, parameterized with a date d and an airline country c, asking for flights between Paris and New York City. We assess the importance of Q's parameters with respect to the reference parameter $d = \texttt{02/24/2024}$ and $c = \texttt{USA}$.

In specifying the parameter distribution Γ, we can tune the exploration of the parameter space. For example, we may choose to only take local parameter perturbations into account. For this, we can specify Γ to be a product distribution whose support is restricted to dates in the vicinity of `02/24/2024`, and to the countries `USA` and `France`. If the similarity function `NegSymCDiff` is used, e.g., then it is measured how close $Q_{\boldsymbol{p}}$ and $Q_{\boldsymbol{p}^*}$ are in terms of the number of given flight options. If `NegDiff` is used, it is measured how many of $Q_{\boldsymbol{p}^*}$'s flight options are lost when the parameters are changed. With $\texttt{MinDiff}_{\textsf{Arrival,Departure}}$, the difference in duration of the shortest flight options is assessed. ⌟

3.3 Formal Problem Statement

We now have all the ingredients to formulate the computation of SHAP scores as a formal computational problem. It can be phrased concisely as follows.

▶ **Problem 3.6.** $\texttt{SHAP}(\mathcal{Q},\texttt{PR},\mathfrak{s})$ is the following computational problem:
On input $(Q,\boldsymbol{p}^*,D,\Gamma)$, compute $\textsc{Shap}(i)$ for all $i \in [\ell]$. ⌟

Table 1 presents an overview of the problem parameters and inputs. For technical reasons (which we explain next), the problem is parameterized with a class of parameterized queries $\mathcal{Q}$, a class of parameter distributions PR, and a (class of) similarity function(s) $\mathfrak{s}$.

Table 1 Problem parameters and inputs for Shap score computation.

Problem parameters		**Problem inputs**	
$\mathcal{Q}$	class of parameterized queries	$Q(\boldsymbol{R};\boldsymbol{P})$	parameterized query
$\mathtt{PR}$	class of parameter distributions	$D \in \mathrm{DB}[\mathsf{dom}(Q)]$	database
$\mathfrak{s}$	similarity function	$\boldsymbol{p}^* = (p_1^*, \dots, p_\ell^*) \in \boldsymbol{P}$	reference parameter
		$\Gamma \in \mathtt{PR}_{\boldsymbol{P}}$, $\Gamma(\boldsymbol{p}^*) > 0$	parameter distribution

Parameter distributions. For simplicity, we only consider parameter distributions with finite support and rational probabilities. A class of parameter distributions will always mean a class $\mathtt{PR} = \bigcup_{\boldsymbol{P}} \mathtt{PR}_{\boldsymbol{P}}$ where $\boldsymbol{P}$ are relation schemas, and $\mathtt{PR}_{\boldsymbol{P}}$ is a set of functions $\Gamma : \mathrm{Tup}[\boldsymbol{P}] \to [0,1]$ such that $\sum_{\boldsymbol{p}} \Gamma(\boldsymbol{p}) = 1$.

▶ **Example 3.7.** One of the simplest classes of parameter distributions is the class IND of *fully factorized distributions*, where all parameters are stochastically independent. A probability distribution Γ over $\mathrm{Tup}[\boldsymbol{P}]$ is called *fully factorized* if

$$\Gamma(p_1, \dots, p_\ell) = \Gamma_1(p_1) \cdots \Gamma_\ell(p_\ell)$$

for all $(p_1, \dots, p_\ell) \in \mathrm{Tup}[\boldsymbol{P}]$ where Γ_i is the marginal distribution of p_i under Γ. To represent Γ, it suffices to provide the ℓ lists of all pairs $(p_i, \Gamma_i(p_i))$ where $\Gamma_i(p_i) > 0$. ⌟

Technically, PR is a class of *representations* of probability distributions in the SHAP problem. For convenience, we do not distinguish between a distribution and (one of) its representation. The only assumption we *globally* make about classes of parameterized distributions is that the encoding length of parameters is polynomially bounded in $\|\Gamma\|$.

Similarity functions. It would feel natural to consider $\mathfrak{s}$ as an input to the problem. Discussing computational complexity in terms of an encoding of such functions is, however, beyond the scope of this paper. We therefore assume, that the similarity function (or rather, a class $(\mathfrak{s}_{\boldsymbol{R}})_{\boldsymbol{R}}$ of versions of the same abstract similarity function $\mathfrak{s}$, one per query range schema) is a parameter to the problem. We abuse notation, and call this class $\mathfrak{s}$ too. Moreover, we write $\mathfrak{s}(T_1, T_2)$ for $\mathfrak{s}_{\boldsymbol{R}}(T_1, T_2)$ when $T_1, T_2 \in \mathrm{DB}[\boldsymbol{R}]$, and $\boldsymbol{R}$ is clear from the context.

Parameterized queries. The parameterized queries we consider are formulated in first-order logic, unless explicitly stated otherwise. The encoding size of $Q_{\boldsymbol{p}}$ can be larger than that of the parameterized query Q. By our assumptions on Γ, this blow-up is at most polynomial in $\|Q\|$ and $\|\Gamma\|$ if $\boldsymbol{p} \in \mathsf{supp}(\Gamma)$.

Complexity. We abuse notation and use $\mathfrak{s} \circ \mathcal{Q}$ to denote the class of functions mapping $(Q, \boldsymbol{p}, \boldsymbol{p}^*, D)$ to $\mathfrak{s}(Q_{\boldsymbol{p}}(D), Q_{\boldsymbol{p}^*}(D))$ where $Q = Q(\boldsymbol{R};\boldsymbol{P}) \in \mathcal{Q}$, where $D \in \mathrm{DB}[\mathsf{dom}(Q)]$, and where $\boldsymbol{p}, \boldsymbol{p}^* \in \mathrm{Tup}[\boldsymbol{P}]$.

▶ **Definition 3.8.** *Let $\mathcal{Q}$ be a class of parameterized queries, $\mathfrak{s}$ a similarity function. We call*
- $\mathcal{Q}$ tractable, *if $(Q, \boldsymbol{p}, D) \mapsto Q_{\boldsymbol{p}}(D)$ can be computed in polynomial time.*
- $\mathfrak{s}$ tractable, *if $(T_1, T_2) \mapsto \mathfrak{s}(T_1, T_2)$, can be computed in polynomial time.*
- $\mathfrak{s} \circ \mathcal{Q}$ tractable, *if $(Q, \boldsymbol{p}, \boldsymbol{p}^*, D) \mapsto \mathfrak{s}(Q_{\boldsymbol{p}}(D), Q_{\boldsymbol{p}^*}(D))$ can be computed in polynomial time.*

Tractability of both $\mathfrak{s}$ and $\mathcal{Q}$ imply tractability of $\mathfrak{s} \circ \mathcal{Q}$, but the converse is not true in general (see [12]). For illustration of this phenomenon, consider the query answering problem for full ACQs: The query answer may be exponentially large, but the answer *count* can be computed efficiently [24, Theorem 1].

▶ **Definition 3.9.** *Let $\mathcal{Q}$ be a class of parameterized queries,* $\texttt{PR}$ *a class of parameter distributions, and $\mathfrak{s}$ a similarity function, and let C be a complexity class.*

- *If the complexity of solving $\texttt{SHAP}(\mathcal{Q}, \texttt{PR}, \mathfrak{s})$, as a function of $\|Q\| + \|\boldsymbol{p}^*\| + \|D\| + \|\Gamma\|$, is in C, then we say that $\texttt{SHAP}(\mathcal{Q}, \texttt{PR}, \mathfrak{s})$ is in C in* combined complexity*.*
- *If for every fixed $Q \in \mathcal{Q}$, the complexity of solving $\texttt{SHAP}(\{Q\}, \texttt{PR}, \mathfrak{s})$, as a function of $\|\boldsymbol{p}^*\| + \|D\| + \|\Gamma\|$, is in C, then we say that $\texttt{SHAP}(\mathcal{Q}, \texttt{PR}, \mathfrak{s})$ is in C in* data complexity*.*

That is, we discuss the computational complexity in terms of variants of the classical combined, and data complexity [34]. If not indicated otherwise, our statements about the complexity of $\texttt{SHAP}(\mathcal{Q}, \texttt{PR}, \mathfrak{s})$ always refer to *combined* complexity.

4 General Insights for Fully Factorized Distributions

We first discuss fully factorized distributions. We start by showing that for tractable similarity functions, tractability of $\texttt{SHAP}(\mathcal{Q}, \texttt{IND}, \mathfrak{s})$ in terms of data complexity is guided by the data complexity of $\mathcal{Q}$. This contrasts other query evaluation settings involving probabilities, like query evaluation in probabilistic databases [30, 33]. The intuitive reason is that here, the probability distributions are tied to the parameterized query instead of the database.

▶ **Proposition 4.1.** *Let $\mathfrak{s}$ be a tractable similarity function and let $Q(\boldsymbol{R}; \boldsymbol{P})$ be a* fixed *parameterized query such that $Q_{\boldsymbol{p}}(D)$ can be computed in polynomial time in $\|D\|$ for all $\boldsymbol{p} \in \mathrm{Tup}[\boldsymbol{P}]$. Then $\texttt{SHAP}(\{Q\}, \texttt{IND}, \mathfrak{s})$ can be solved in polynomial time.*

Proof sketch. When the parameterized query is fixed, ℓ is constant. In particular, the probability spaces Π_i are of constant size. The problem can thus be solved by brute force, i.e., by evaluating the explicit formula for the Shap score. ◀

For this reason, in the remainder of the paper, we will focus on the *combined complexity* of the computation of $\texttt{SHAP}(\mathcal{Q}, \texttt{IND}, \mathfrak{s})$.

Next, we point out two relationships with other computational problems. First, we see that $\texttt{SHAP}(\mathcal{Q}, \texttt{IND}, \mathfrak{s})$ is at least as hard as deciding whether the output of a parameterized query is non-empty. To be precise, let $\mathcal{Q}$ be a class of parameterized queries and let $\texttt{NONEMPTY}(\mathcal{Q})$ be the problem that takes inputs $(Q, \boldsymbol{p}^*, D)$, and asks to decide whether $Q_{\boldsymbol{p}^*}(D) \neq \emptyset$. In order to establish a relationship to the $\texttt{SHAP}(\mathcal{Q}, \texttt{IND}, \mathfrak{s})$ problem, we need two more ingredients.

▶ **Definition 4.2.** *A similarity function $\mathfrak{s}$* strongly depends on its first argument*, if for all possible range schemas $\boldsymbol{R}$, one of the following is satisfied:*

(s1) *for all non-empty $T \in \mathrm{Rel}[\boldsymbol{R}]$ we have $\mathfrak{s}_{\boldsymbol{R}}(\emptyset, \emptyset) \neq \mathfrak{s}_{\boldsymbol{R}}(T, \emptyset)$; or*

(s2) *for all non-empty $T \in \mathrm{Rel}[\boldsymbol{R}]$ we have $\mathfrak{s}_{\boldsymbol{R}}(\emptyset, T) \neq \mathfrak{s}_{\boldsymbol{R}}(T, T)$.*

Both are highly natural conditions for meaningful similarity functions. Moreover, they entail that $\mathfrak{s}$ can, to a certain extent, distinguish whether its first argument is empty or not. For example, the similarity functions from Example 3.2(a)–(c) all satisfy (s2), whereas a version of $\texttt{NegDiff}$ that swaps the roles of Q and Q^* would always satisfy (s1).

▶ **Theorem 4.3.** *Let $\mathfrak{s}$ strongly depend on its first argument. Let $\mathcal{Q}$ be a class of parameterized queries such that for all $Q(\boldsymbol{R}; \boldsymbol{P}) \in \mathcal{Q}$ there exists $i_0 \in [\ell]$ (which we can find in polynomial time) such that, for all $D \in \mathrm{DB}[\mathsf{dom}(Q)]$ and all $\boldsymbol{p} = (p_1, \dots, p_\ell) \in \mathrm{Tup}[\boldsymbol{P}]$ with $p_{i_0} \notin \mathsf{adom}(D)$, we have $Q_{\boldsymbol{p}}(D) = \emptyset$. Then $\texttt{NONEMPTY}(\mathcal{Q}) \leq^{\mathsf{P}}_{\mathsf{T}} \texttt{SHAP}(\mathcal{Q}, \texttt{IND}, \mathfrak{s})$.*

Proof sketch. Our probability distribution Γ is just a coin flip between two parameter values $\boldsymbol{p}^*$ and $\boldsymbol{p}^a$, where $\boldsymbol{p}^a$ coincides with $\boldsymbol{p}^*$ on all parameters except for $p^*_{i_0}$ which is replaced with a value $a \neq p^*_{i_0}$ outside of the active domain of D. Calculation shows that the $\textsc{Shap}(i_0) = 0$ if and only if $Q_{\boldsymbol{p}^*}(D) = \emptyset$. (We reduce to the $\mathtt{SHAP}$ instance $(Q, \boldsymbol{p}^a, D, \Gamma)$ or $(Q, \boldsymbol{p}^*, D, \Gamma)$, depending on whether $\mathfrak{s}$ satisfies (s1) or (s2)). ◀

The restriction we impose on $\mathcal{Q}$ expresses that one of the parameters immediately renders the query result on D empty, if it is chosen outside the active domain of D. This is fulfilled, for example, for unions of conjunctive queries for which there exists a parameter that appears in each of its conjunctive queries, and can be extended to Datalog queries for which there exists a parameter that appears in the body of every rule.

Van den Broeck et al. [32] have shown a reduction from computing $\textsc{Shap}$ to computing the expected similarity. The latter problem, $\mathtt{ESIM}(\mathcal{Q}, \mathtt{PR}, \mathfrak{s})$, takes the same parameters and inputs as $\mathtt{SHAP}(\mathcal{Q}, \mathtt{PR}, \mathfrak{s})$, but asks for $\mathbb{E}_{\boldsymbol{p}\sim\Gamma}[\mathfrak{s}(\boldsymbol{p}, \boldsymbol{p}^*)]$ instead of $\textsc{Shap}(i)$.

▶ **Theorem 4.4** (Cf. [32, Theorem 2]). *For any class of parameterized queries $\mathcal{Q}$ and any similarity function $\mathfrak{s}$ such that $\mathfrak{s} \circ \mathcal{Q}$ is tractable, we have* $\mathtt{SHAP}(\mathcal{Q}, \mathtt{IND}, \mathfrak{s}) \equiv^{\mathsf{P}}_{\mathsf{T}} \mathtt{ESIM}(\mathcal{Q}, \mathtt{IND}, \mathfrak{s})$.

▶ Remark 4.5. The paper [32] discusses a more general setting, in which $\textsc{Shap}$ scores are computed for tractable functions F over ℓ-ary domains. They provide an algorithm that computes $\textsc{Shap}$ scores of F using oracle calls to the expected value of F with different parameter distributions $\Gamma \in \mathtt{IND}$. This algorithm runs in polynomial time in $\|\Gamma\|$ and is independent of the choice of F (apart from the oracle, of course). For our version of the theorem, we use this algorithm on functions $F_{Q,\boldsymbol{p}^*,D}(\boldsymbol{p}) = \mathfrak{s}(Q_{\boldsymbol{p}}(D), Q_{\boldsymbol{p}^*}(D))$. ⌟

The key property we needed for Proposition 4.1 was the manageable size of the parameter distributions. This can be formulated as a property of a class of queries.

▶ **Definition 4.6.** *Let $Q(\boldsymbol{R}; \boldsymbol{P})$ be a parameterized query and $D \in \mathrm{DB}[\mathsf{dom}(Q)]$, and denote* $\mathsf{psupp}(Q, D) := \{\boldsymbol{p} \in \mathrm{Tup}[\boldsymbol{P}] \colon Q_{\boldsymbol{p}}(D) \neq \emptyset\}$. *A class $\mathcal{Q}$ of parameterized queries has* polynomially computable parameter support *if there exists a polynomial time algorithm (in $\|Q\| + \|D\|$) which, on input $Q(\boldsymbol{R}; \boldsymbol{P}) \in \mathcal{Q}$ and $D \in \mathrm{DB}[\mathsf{dom}(Q)]$, outputs a set $P(Q, D) \supseteq \mathsf{psupp}(Q, D)$.*

There are simple, yet relevant classes with this property, e.g., parameterized CQs with a bounded number of joins, or where parameters only appear in a bounded number of atoms. This property allows the efficient computation of expected similarities.

▶ **Proposition 4.7.** *Let $\mathcal{Q}$ have polynomially computable parameter support, and suppose $\mathfrak{s} \circ \mathcal{Q}$ is tractable. Then $\mathtt{SHAP}(\mathcal{Q}, \mathtt{IND}, \mathfrak{s})$ can be solved in polynomial time.*

Proof sketch. We solve $\mathtt{ESIM}(\mathcal{Q}, \mathtt{IND}, \mathfrak{s})$ efficiently, by first computing some $P \supseteq \mathsf{psupp}(Q, D)$ and then evaluating the formula for $\mathbb{E}_{\boldsymbol{p}}[\mathfrak{s}(\boldsymbol{p}, \boldsymbol{p}^*)]$. For this, we explicitly compute the terms for $\boldsymbol{p} \in P$, and bundle those for $\boldsymbol{p} \notin P$. By Theorem 4.4, this yields tractability of $\mathtt{SHAP}(\mathcal{Q}, \mathtt{IND}, \mathfrak{s})$. ◀

5 Parameterized Conjunctive Queries with Independent Parameters

In this section, we study the complexity of the $\textsc{Shap}$ score for the parameters of acyclic conjunctive queries. Later in the section, we also investigate the addition of inequalities to such queries.

5.1 Acyclic Conjunctive Queries

We first focus on parameterized *acyclic* conjunctive queries (pACQs). In particular, we will also consider classes of Boolean parameterized queries. If $\boldsymbol{R}$ is a Boolean relation schema, then the only possible inputs to a similarity function $\mathfrak{s}_{\boldsymbol{R}}$ are pairs of $\mathtt{tt}$ and ff. In particular, the properties from Definition 4.2 become

(s1′) $\mathfrak{s}_{\boldsymbol{R}}(\mathsf{ff},\mathsf{ff}) \neq \mathfrak{s}_{\boldsymbol{R}}(\mathtt{tt},\mathsf{ff})$ for (s1);

(s2′) $\mathfrak{s}_{\boldsymbol{R}}(\mathsf{ff},\mathtt{tt}) \neq \mathfrak{s}_{\boldsymbol{R}}(\mathtt{tt},\mathtt{tt})$ for (s2).

If $\mathfrak{s}$ takes Boolean inputs and is strongly dependent on its first argument, then $\mathfrak{s}$ satisfies one of (s1′), (s2′). Such $\mathfrak{s}$ is also always tractable, because it has only four possible inputs: the pairs of the constants $\mathtt{tt}$ and ff.

▶ **Proposition 5.1.** *Let $\mathcal{Q}$ be the class of Boolean parameterized queries of the shape*

$$Q(;y_1,\dots,y_\ell) = \exists x\colon R_1(x,y_1) \wedge \dots \wedge R_\ell(x,y_\ell) \tag{4}$$

and let $\mathfrak{s}$ be strongly dependent on its first argument. Then $\mathtt{SHAP}(\mathcal{Q},\mathtt{IND},\mathfrak{s})$ *is* #P*-hard.*

Proof sketch. It can easily be checked that $\mathcal{Q}$ is tractable (although it does not have polynomially computable parameter support). Therefore, so is $\mathfrak{s}\circ\mathcal{Q}$. We use Theorem 4.4 and show that $\mathtt{ESIM}(\mathcal{Q},\mathtt{IND},\mathfrak{s})$ is #P-hard by reduction from #`posDNF`. The main idea is as follows. Let φ be a positive DNF formula with variables X_i and disjuncts φ_j. Construct a database D in which the tuples $R_i(j,y_i)$ represent the possible truth assignments for X_i to satisfy φ_j: If X_i occurs in φ_j, then y_i should be $\mathtt{tt}$; otherwise, X_i does not matter in φ_j and y_i can take both values $\mathtt{tt}$ or ff. With this interpretation, Q, on D, expresses the existence of a disjunct φ_j, in which all occurring variables are set to $\mathtt{tt}$. The expected similarity under the uniform distribution can be used to recover $\#\varphi$. ◀

▶ Remark 5.2. Our hardness results are stated for $\mathtt{PR} = \mathtt{IND}$. Inspection of our proofs reveals that they already hold for the subclass $\mathtt{PR} = \mathtt{UNIF}$ of parameter distributions that are uniform on their support. ⌟

Next, we show a tractability result for parameterized *full* ACQs. In contrast, by Proposition 5.1, even a single existential quantifier can make the problem difficult. A similar observation has been made for the (weighted) counting for ACQ answers [24, 9] (which we use to establish tractability here).

▶ **Proposition 5.3.** *Let $\mathcal{Q}$ be the class of full pACQs, and let $\mathfrak{s}$ be any of* `Int`*,* `NegSymDiff`*, or* `NegDiff`*. Then* $\mathtt{SHAP}(\mathcal{Q},\mathtt{IND},\mathfrak{s})$ *can be solved in polynomial time.*

Proof sketch. The main idea of the proof is to introduce an artificial "similarity" function `Count` with $\mathtt{Count}(T_1,T_2) = |T_1|$. Tractability of $\mathtt{Count}\circ\mathcal{Q}$ is given, since counting the answers to full ACQs is tractable [24]. Thus, we can use Theorem 4.4 again and discuss $\mathtt{ESIM}(\mathcal{Q},\mathtt{IND},\mathtt{Count})$ first. This problem can be reduced to the weighted answer counting problem for ACQs, which is also tractable for full ACQs by [9]. A simple construction allows transferring tractability to $\mathtt{SHAP}(\mathcal{Q},\mathtt{IND},\mathtt{Int})$. The similarities `NegSymDiff` and `NegDiff` are linear combinations in `Int` and `Count`. Therefore, we can compute the Shap scores for the former two efficiently by computing the Shap scores for the latter two. ◀

▶ Remark 5.4. In light of [9, Proposition 5], our tractability result for `Int` may seem surprising. The construction in our proof relies on the fact that in our case, the intersection concerns different parameterizations *of the same query*, one of them being fixed. If it were performed for two *arbitrary* full ACQs, acyclicity can be lost. ⌟

5.2 Conjunctive Queries with Inequalities

In this section, we investigate parameterized conjunctive queries Q with inequalities of the shape $x_i \leq y_j$, where x_i is a variable (which also appears in the inequality-free part of Q) and y_j is a parameter. We refer to such queries as p$^{\leq}$CQs. For simplicity, we assume that all attribute domains are numerical.

If $\mathcal{Q}$ is a class of pCQs (i.e., without inequalities), then for $t = 0, 1, 2, \ldots$ we let $\mathcal{Q}^{(\leq,t)}$ denote the class of p$^{\leq}$CQs obtained by adding at most t inequalities of the above shape to the body of a query from $\mathcal{Q}$. We let $\mathcal{Q}^{\leq} = \bigcup_{t=0}^{\infty} \mathcal{Q}^{(\leq,t)}$.

▶ Remark 5.5. In practice, such inequalities would typically be used for attributes with continuous domains, like $\mathbb{R}$. Continuous parameter values would require a treatment of continuous parameter distributions. Our framework can handle continuous parameter distributions as follows: On the technical side, the definition of the Shap score needs to be changed to avoid conditional probabilities. For fully factorized distributions, this can be done easily. On the algorithmic side, we can discretize the continuous distribution into intervals defined by the values in the database to obtain a discrete distribution with finite support that yields the same output. ⌟

▶ **Proposition 5.6.** *Let $\mathcal{Q}$ be the class of* p$^{\leq}$CQ*s of the shape*

$$Q(;y_1,\ldots,y_\ell) = \exists x_1,\ldots,x_\ell \colon R(x_1,\ldots,x_\ell) \wedge (x_1 \leq y_1) \wedge \cdots \wedge (x_\ell \leq y_\ell). \tag{5}$$

and let $\mathfrak{s}$ be strongly dependent on its first argument. Then $\mathtt{SHAP}(\mathcal{Q}, \mathtt{IND}, \mathfrak{s})$ *is* #P*-hard.*

Proof sketch. This proof works very similar to that of Proposition 5.1. First, we observe that $\mathcal{Q}$ is tractable (but again not with polynomially parameter support). Then $\mathfrak{s} \circ \mathcal{Q}$ is tractable too. In the remainder of the proof, we prove hardness of $\mathtt{ESIM}(\mathcal{Q}, \mathtt{IND}, \mathfrak{s})$ by reduction from #`posDNF`, similar to the proof of Proposition 5.1. ◀

The simple shape of (5) indicates that structural restrictions on the usage of inequalities are needed to establish tractability. If $\mathcal{Q}$ is a class of pCQs (i.e., without $\leq$), then we call a query $Q \in \mathcal{Q}^{\leq}$ *acyclic* (a p$^{\leq}$ACQ), if it becomes a pACQ, when $\leq$ is interpreted as a relation symbol. It can be shown that Proposition 5.3 extends to this setting:

▶ **Proposition 5.7.** *Let $\mathcal{Q}$ be the class of full* pACQ*s (i.e., without $\leq$), and let $\mathcal{Q}'$ be a class of full* p$^{\leq}$ACQ*s such that $\mathcal{Q}' \subseteq \mathcal{Q}^{\leq}$. Moreover, let $\mathfrak{s}$ be any of* `Int`*,* `NegSymDiff`*, or* `NegDiff`*. Then* $\mathtt{SHAP}(\mathcal{Q}', \mathtt{IND}, \mathfrak{s})$ *can be computed in polynomial time.*

Proof sketch. We replace inequalities $x_i \leq y_j$ with atoms $R_{\leq}(x_i, y_j)$ and hard-code the *relevant* $R_{\leq}$ tuples in the database D. Then, Proposition 5.3 can be applied. ◀

6 Correlated Parameters and Approximability

In this section, we allow classes `PR` of parameter distributions with correlations. We only need to make the following tractability assumptions (which are trivially satisfied by `IND`).

1. For every fixed ℓ, and every $\Gamma \in \mathtt{PR}_{\boldsymbol{P}}$ with $|\boldsymbol{P}| = \ell$, the support $\mathsf{supp}(\Gamma)$ can be computed in polynomial time in $\|\Gamma\|$.
2. For all $\boldsymbol{p}$ and J, we can compute $\Pr_{\boldsymbol{p}' \sim \Gamma}(\boldsymbol{p}'_J = \boldsymbol{p}_J)$ in polynomial time in $\|\Gamma\|$.

For example, the first property holds for distributions encoded by Bayesian networks. The second one holds for structurally restricted classes of Bayesian networks, like polytrees [13]. The following result states that given these assumptions, the data complexity of the `SHAP` problem remains in polynomial time even for parameter distributions with correlations. The proof is similar to the proof of Proposition 4.1.

▶ **Proposition 6.1.** *Let $\mathfrak{s}$ be a tractable similarity function and let $Q(\boldsymbol{R};\boldsymbol{P})$ be a* fixed *parameterized query such that $Q_{\boldsymbol{p}}(D)$ can be computed in polynomial time in $\|D\|$ for all $\boldsymbol{p} \in \mathrm{Tup}[\boldsymbol{P}]$. Moreover, let $\mathtt{PR}$ be a class of distributions as described above. Then $\mathtt{SHAP}(\{Q\}, \mathtt{PR}, \mathfrak{s})$ can be solved in polynomial time.*

We conclude this section by explaining how SHAP can be approximated via sampling. Consider an input $(Q, \boldsymbol{p}^*, D, \Gamma)$ of $\mathtt{SHAP}(\mathcal{Q}, \mathtt{PR}, \mathfrak{s})$, where $Q = Q(\boldsymbol{R};\boldsymbol{P})$ and $|\boldsymbol{p}^*| = \ell$. We rewrite $\textsc{Shap}(i)$ as an expectation in a single probability space, instead of nested expectations in different spaces. Let $\{i\}^0 = \emptyset$ and $\{i\}^1 = \{i\}$. Consider the following two-step random process, for $b \in \{0,1\}$:

1. Draw $J \subseteq [\ell] \setminus i$ according to Π_i.
2. Draw $\boldsymbol{p}$ according to Γ, conditioned on having $\boldsymbol{p}$ agree with $\boldsymbol{p}^*$ on J.

This defines a joint probability distribution on pairs $(\boldsymbol{p}, J)$. By $\Gamma^{i,1}$ and $\Gamma^{i,0}$, we denote the corresponding marginal distributions over parameter tuples $\boldsymbol{p}$. A simple calculation shows that the following equality holds:

$$\textsc{Shap}(i) = \mathop{\mathbb{E}}_{\boldsymbol{p}\sim\Gamma^{i,1}}\big[\mathfrak{s}(\boldsymbol{p},\boldsymbol{p}^*)\big] - \mathop{\mathbb{E}}_{\boldsymbol{p}\sim\Gamma^{i,0}}\big[\mathfrak{s}(\boldsymbol{p},\boldsymbol{p}^*)\big]. \tag{6}$$

We give a proof of this statement in the full version of the paper [11].

We say that PR *admits efficient conditional sampling* if for all $\Gamma \in \mathtt{PR}$, all $\boldsymbol{p}^* \in \mathsf{supp}(\Gamma)$, and all $J \subseteq [\ell]$, the conditional distribution of Γ subject to $\boldsymbol{p}_J = \boldsymbol{p}^*_J$ can be sampled in polynomial time in $\|\Gamma\| + \|\boldsymbol{p}^*\|$. This is again the case, for example, for structurally restricted classes of Bayesian networks. By the structure of the two-step process defining $\Gamma^{i,b}$, we can efficiently sample from $\Gamma^{i,b}$ if we can efficiently sample from conditional distributions of Γ. A proof of this can be found in the full version of the paper [11].

A similarity function $\mathfrak{s}$ is *bounded* for a class of parameterized queries $\mathcal{Q}$ if there are $a \le b$ such that for all $(Q, \boldsymbol{p}^1, \boldsymbol{p}^2)$, $Q \in \mathcal{Q}$, we have $a \le \mathfrak{s}(\boldsymbol{p}^1, \boldsymbol{p}^2) \le b$. For example, similarity measures, like `Jaccard`, are usually $[0,1]$-valued. The following theorem states that, under the assumptions of efficient conditional sampling and boundedness, we have an additive FPRAS for the Shap score of a parameter.

▶ **Theorem 6.2.** *Let $\mathcal{Q}$ be a class of tractable parameterized queries, let $\mathfrak{s}$ a tractable similarity function, and let $\mathtt{PR}$ be a class of parameter distributions such that*

(a) *Every $\Gamma \in \mathtt{PR}$ admits efficient conditional sampling.*

(b) *The value range of $\mathfrak{s}$ is bounded for $\mathcal{Q}$.*

Then, for all inputs $(Q, \boldsymbol{p}^*, D, \Gamma)$ and all $i \in [\ell]$, we can compute a value S satisfying $\Pr(|S - \textsc{Shap}(i)| < \varepsilon) \ge 1 - \delta$ in time polynomial in $\frac{1}{\varepsilon}$, $\log\frac{1}{\delta}$, and the size of the input.

Proof sketch. The proof uses a simple sampling procedure and approximates the two terms in Equation (6) from the means of the samples. The guarantee is then given by a variant of Hoeffding's inequality (requiring boundedness of $\mathfrak{s}$ to be applicable). ◀

7 Conclusions

We proposed a framework for measuring the responsibility of parameters to the result of a query using the Shap score. We studied the computational problem of calculating the Shap score of a given parameter value. We gave general complexity lower and upper bounds, and presented a complexity analysis for the restricted case of conjunctive queries and independent parameters. We also discussed the complexity of approximate calculation and correlated parameters. The rich framework we introduced here offers many opportunities for future research. Especially important is the direction of aggregate queries, where the similarity between results accounts for the numerical values such as sum, average, median, and so on. For such queries, it is important to study numerical parameter distributions, which are typically continuous probability measures. It is also important to identify general tractability conditions for similarity measures and parameter distributions in order to generalize the upper bounds beyond the special cases that we covered here. Finally, we plan to explore the applicability of SHAP to measuring parameters in various queries and datasets, such as those studied in the context of fact checking [5, 31].

References

1 Encarnación Algaba, Vito Fragnelli, and Joaquín Sánchez-Soriano, editors. *Handbook of the Shapley Value.* CRC Press, 2019. doi:10.1201/9781351241410.

2 Dana Arad, Daniel Deutch, and Nave Frost. LearnShapley: Learning to predict rankings of facts contribution based on query logs. In *CIKM*, pages 4788–4792. ACM, 2022. doi:10.1145/3511808.3557204.

3 Marcelo Arenas, Pablo Barceló, Leopoldo E. Bertossi, and Mikaël Monet. The tractability of shap-score-based explanations for classification over deterministic and decomposable boolean circuits. In *AAAI*, pages 6670–6678. AAAI Press, 2021. doi:10.1609/aaai.v35i8.16825.

4 Leopoldo E. Bertossi, Loreto Bravo, Enrico Franconi, and Andrei Lopatenko. The complexity and approximation of fixing numerical attributes in databases under integrity constraints. *Inf. Syst.*, 33(4-5):407–434, 2008. doi:10.1016/j.is.2008.01.005.

5 Adriane Chapman and H. V. Jagadish. Why not? In *SIGMOD Conference*, pages 523–534. ACM, 2009. doi:10.1145/1559845.1559901.

6 Xiaotie Deng and Christos H. Papadimitriou. On the complexity of cooperative solution concepts. *Math. Oper. Res.*, 19(2):257–266, 1994. doi:10.1287/moor.19.2.257.

7 Daniel Deutch, Nave Frost, Amir Gilad, and Oren Sheffer. Explanations for data repair through shapley values. In *CIKM*, pages 362–371. ACM, 2021. doi:10.1145/3459637.3482341.

8 Daniel Deutch, Nave Frost, Benny Kimelfeld, and Mikaël Monet. Computing the shapley value of facts in query answering. In *SIGMOD Conference*, pages 1570–1583. ACM, 2022. doi:10.1145/3514221.3517912.

9 Arnaud Durand and Stefan Mengel. The complexity of weighted counting for acyclic conjunctive queries. *J. Comput. Syst. Sci.*, 80(1):277–296, 2014. doi:10.1016/j.jcss.2013.08.001.

10 U. Faigle and W. Kern. The shapley value for cooperative games under precedence constraints. *Int. J. Game Theory*, 21(3):249–266, sep 1992. doi:10.1007/BF01258278.

11 Martin Grohe, Benny Kimelfeld, Peter Lindner, and Christoph Standke. The importance of parameters in database queries, 2024. arXiv:2401.04606 [cs.DB]. doi:10.48550/arXiv.2401.04606.

12 Yuri Gurevich and Saharon Shelah. Time polynomial in input or output. *J. Symb. Log.*, 54(3):1083–1088, 1989. doi:10.2307/2274767.

13 Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques.* Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA, 2009.

14 Nick Koudas, Chen Li, Anthony K. H. Tung, and Rares Vernica. Relaxing join and selection queries. In *VLDB*, pages 199–210. ACM, 2006. URL: http://dl.acm.org/citation.cfm?id=1164146.

15 Marie-Jeanne Lesot, Maria Rifqi, and Hamid Benhadda. Similarity measures for binary and numerical data: a survey. *International Journal of Knowledge Engineering and Soft Data Paradigms*, 1(1):63–84, dec 2008. doi:10.1504/ijkesdp.2009.021985.

16 Yin Lin, Brit Youngmann, Yuval Moskovitch, H. V. Jagadish, and Tova Milo. On detecting cherry-picked generalizations. *Proc. VLDB Endow.*, 15(1):59–71, 2021. doi:10.14778/3485450.3485457.

17 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, 2020. `doi:10.4230/LIPIcs.ICDT.2020.20`.

18 Ester Livshits and Benny Kimelfeld. The shapley value of inconsistency measures for functional dependencies. *Log. Methods Comput. Sci.*, 18(2), 2022. `doi:10.46298/lmcs-18(2:20)2022`.

19 Scott M. Lundberg, Gabriel G. Erion, Hugh Chen, Alex J. 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. *Nat. Mach. Intell.*, 2(1):56–67, 2020. `doi:10.1038/s42256-019-0138-9`.

20 Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *NIPS*, pages 4765–4774, 2017. URL: `https://proceedings.neurips.cc/paper/2017/hash/8a20a8621978632d76c43dfd28b67767-Abstract.html`.

21 Christoph Molnar. Interpretable machine learning: A guide for making black box models explainable, 2023. Version 2023-08-21. URL: `https://christophm.github.io/interpretable-ml-book`.

22 Davide Mottin, Alice Marascu, Senjuti Basu Roy, Gautam Das, Themis Palpanas, and Yannis Velegrakis. A probabilistic optimization framework for the empty-answer problem. *Proc. VLDB Endow.*, 6(14):1762–1773, 2013. `doi:10.14778/2556549.2556560`.

23 Santiago Ontañón. An overview of distance and similarity functions for structured data. *Artif. Intell. Rev.*, 53(7):5309–5351, 2020. `doi:10.1007/s10462-020-09821-w`.

24 Reinhard Pichler and Sebastian Skritek. Tractable counting of the answers to conjunctive queries. *J. Comput. Syst. Sci.*, 79(6):984–1001, 2013. `doi:10.1016/j.jcss.2013.01.012`.

25 Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why should I trust you?": Explaining the predictions of any classifier. In *KDD*, pages 1135–1144. ACM, 2016. `doi:10.1145/2939672.2939778`.

26 Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018. `doi:10.1609/aaai.v32i1.11491`.

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

28 B. Sathiya and T. V. Geetha. A review on semantic similarity measures for ontology. *J. Intell. Fuzzy Syst.*, 36(4):3045–3059, 2019. `doi:10.3233/JIFS-18120`.

29 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.

30 Dan Suciu, Dan Olteanu, Christopher Ré, and Christoph Koch. *Probabilistic Databases*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011. `doi:10.2200/S00362ED1V01Y201105DTM016`.

31 Quoc Trung Tran and Chee-Yong Chan. How to conquer why-not questions. In *SIGMOD Conference*, pages 15–26. ACM, 2010. `doi:10.1145/1807167.1807172`.

32 Guy Van den Broeck, Anton Lykov, Maximilian Schleich, and Dan Suciu. On the tractability of SHAP explanations. *Journal of Artificial Intelligence Research*, 74:851–886, jun 2022. `doi:10.1613/jair.1.13283`.

33 Guy Van den Broeck and Dan Suciu. Query processing on probabilistic data: A survey. *Found. Trends Databases*, 7(3-4):197–341, 2017. `doi:10.1561/1900000052`.

34 Moshe Y. Vardi. The complexity of relational query languages (extended abstract). In Harry R. Lewis, Barbara B. Simons, Walter A. Burkhard, and Lawrence H. Landweber, editors, *Proceedings of the 14th Annual ACM Symposium on Theory of Computing, May 5-7, 1982, San Francisco, California, USA*, pages 137–146. ACM, 1982. `doi:10.1145/800070.802186`.

35 You Wu, Pankaj K. Agarwal, Chengkai Li, Jun Yang, and Cong Yu. Computational fact checking through query perturbations. *ACM Trans. Database Syst.*, 42(1):4:1–4:41, 2017. `doi:10.1145/2996453`.