The Design of Arbitrage-Free Data Pricing Schemes

Shaleen Deep[1] **and Paraschos Koutris**[2]

1 **University of Wisconsin-Madison, Madison, WI**
`shaleen@cs.wisc.edu`
2 **University of Wisconsin-Madison, Madison, WI**
`paris@cs.wisc.edu`

Abstract

Motivated by a growing market that involves buying and selling data over the web, we study pricing schemes that assign value to queries issued over a database. Previous work studied pricing mechanisms that compute the price of a query by extending a data seller's explicit prices on certain queries, or investigated the properties that a pricing function should exhibit without detailing a generic construction. In this work, we present a formal framework for pricing queries over data that allows the construction of general families of pricing functions, with the main goal of avoiding arbitrage. We consider two types of pricing schemes: instance-independent schemes, where the price depends only on the structure of the query, and answer-dependent schemes, where the price also depends on the query output. Our main result is a complete characterization of the structure of pricing functions in both settings, by relating it to properties of a function over a lattice. We use our characterization, together with information-theoretic methods, to construct a variety of arbitrage-free pricing functions. Finally, we discuss various tradeoffs in the design space and present techniques for efficient computation of the proposed pricing functions.

1998 ACM Subject Classification H.2.4 [Systems]: Relational Databases

Keywords and phrases Data pricing, Determinacy, Arbitrage

1 Introduction

The commodification of data over the last decade has created many unique research challenges, among them data privacy and pricing of data. In a broad range of application areas, data today is being collected at an unprecedented scale. This phenomenon has led to a growing market for so called big data brokers, who sell this data to buyers such as financial firms, retailers and insurance companies [6, 5].

In this paper, we investigate the problem of *query-based data pricing*, where the task is to assign prices to queries over a database, such that the price captures the amount of information revealed by asking the query. Traditionally, data pricing has been done either by allowing the buyer to access only certain queries with a fixed price set by the seller, or the buyer needs to purchase the whole dataset [20]. Although such an approach is conceptually simple, defining a large set of queries that are representative of the user's needs is a tall task for the data seller. Even if this is feasible, such a pricing scheme may allow *arbitrage*, which occurs when a data buyer can potentially buy data at a price less than what is set by the seller. It can also lead to prices that exhibit undesirable behavior.

Previous work in the area of data pricing has identified a set of *arbitrage conditions* that any reasonable pricing function should avoid. The fundamental arbitrage condition is *information arbitrage*, first introduced in [16]. Intuitively, a query Q_1 that reveals a subset of the information that is revealed by another query Q_2 should be priced at most as much

20th International Conference on Database Theory (ICDT 2017).
Editors: Michael Benedikt and Giorgio Orsi; Article No. 2; pp. 2:1–2:18

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

as Q_2. If not, an arbitrage opportunity occurs: a clever buyer can pay the price of Q_2 and then use the result of Q_2 to compute Q_1 for a lower price. A second arbitrage condition is *bundle arbitrage* [20]. Intuitively, asking simultaneously for Q_1 and Q_2 (as a bundle) should cost at most the sum of asking separately for each. Both [16, 20] propose pricing functions that avoid both arbitrage conditions. However, to the best of our knowledge, there exists no framework that supports a generic construction of pricing functions, and facilitates the analysis of the various tradeoffs in design choices.

Our Contribution. We address the question of designing arbitrage-free pricing schemes that assign prices to queries over a database. Our main result is a complete characterization of the structure of pricing functions for two pricing schemes: *answer-dependent prices* (APS), and *instance-independent prices* (QPS). We use this characterization to construct a variety of pricing functions, and also discuss the various tradeoffs involved in choosing the right pricing function. We summarize below our results in more detail.

We first study APS, where the price depends both on the query Q and on the answer of the query $E = Q(D)$. To characterize such schemes, we define the *conflict set*, which is the set of databases such that $Q(D) \neq E$. We show that any arbitrage-free pricing function is equivalent to a monotone and subadditive function over the join-semilattice defined by the conflict sets (Theorems 8 and 9). Equipped with this characterization, we present several examples of arbitrage-free functions, including the weighted coverage and the weighted set cover functions. In addition, we show that an answer-dependent pricing function with no bundle arbitrage leads to unnatural behavior: any query can cost at least half the price of the whole dataset for some databases. This suggests that there is a tradeoff that any data seller must take into account when choosing a pricing function.

Second, we examine the structure of QPS, where the pricing function depends only on the query Q. We prove that any non-trivial instance-independent pricing function must have weaker arbitrage guarantees compared to an answer-dependent function. To provide a characterize of functions in QPS, we view the query Q as a *partition* over the set of possible databases: our main results is that any arbitrage-free function is equivalent to a monotone and subadditive function over the elements of the join-semilattice formed from the partitions (Theorems 24 and 25).

To design pricing functions in QPS, we apply two methods. The first method applies an appropriate aggregate function to combine the prices of an arbitrage-free answer-dependent function (Theorem 28). The second method views the database as a random variable (with some probability distribution over the possible databases), and computes the price as the *information gain* of the data buyer after the answer has been revealed (Section 4.4). This approach is parallel to work on side-channel attacks [13], and quantitative information flow [14]. By using different entropy measures, such as *Shannon entropy*, or *min-entropy*, we obtain pricing functions that we prove to be arbitrage-free using the machinery we developed.

Third, we show how the proposed pricing functions can be computed efficiently in practical settings. We discuss two different techniques. The first method restricts the computation of a pricing function to a small set of databases (instead of all possible databases). The second method uses approximation techniques to estimate the price within a small margin of error.

Organization. Section 2 presents the key concepts, terminology and notation that we use throughout the paper. In Section 3, we study the construction and properties of pricing functions for the answer-dependent case. Section 4 details the corresponding problem for instance-independent pricing schemes. Section 5 discusses techniques to compute a pricing function efficiently. We present the related work and conclude in Sections 6 and 7 respectively.

2 Notation and Framework

In this section, we set up the necessary notation and formally describe the pricing framework.

2.1 Preliminaries

We fix a relational schema $\mathbf{R} = (R_1, \ldots, R_k)$; we use D to denote a database instance that uses the schema. We will use $\mathcal{I}$ to denote the set of possible database instances. The set $\mathcal{I}$ encodes information about the database that is provided by the data seller, and is public information known to any data buyer. Further, we allow the set $\mathcal{I}$ to be infinite, but countable. For example, suppose that the schema consists of a single binary relation $R(A, B)$ and we know that the domain of both attributes is $[n] = \{1, \ldots, n\}$. Then, $\mathcal{I} = 2^{[n]\times[n]}$, which represents equivalently the set of all possible directed graphs on the vertex set $[n]$.

We will view a *query* Q from some query language $\mathcal{L}$ as a deterministic function that takes as input a database instance $D \in \mathcal{I}$ and returns an output $Q(D)$. In this paper, we do not impose any restriction on the query language $\mathcal{L}$, but in the examples we will use and in some of the design tradeoffs we assume Q is either a *conjunctive query (CQ)* or a *union of conjunctive queries (UCQ)*. A *query bundle* $\mathbf{Q} = (Q_1, \ldots, Q_n)$ is a finite set of queries that is asked simultaneously on the database. We denote by $B(\mathcal{L})$ the set of finite query bundles from the language $\mathcal{L}$. Given two query bundles $\mathbf{Q}_1, \mathbf{Q}_2$, we denote their union as $\mathbf{Q} = \mathbf{Q}_1, \mathbf{Q}_2$.

Queries as Partitions. It will be handy to provide an alternative viewpoint of a query bundle $\mathbf{Q}$ as a partition over the set of instances $\mathcal{I}$. A *partition* $\mathcal{P} = \{B_1, \ldots, B_k\}$ of $\mathcal{I}$ is a set of pairwise disjoint sets $B_i \subseteq \mathcal{I}$, which we call *blocks*, such that $\cup_{i=1}^{k} B_i = \mathcal{I}$. Given $\mathbf{Q} \in \mathcal{L}$, we denote by $\mathcal{P}_\mathbf{Q}$ the partition that is induced by the following *equivalence relation*: $D \sim D'$ iff $\mathbf{Q}(D) = \mathbf{Q}(D')$ and $\mathbf{Q} \in \mathcal{L}$. In other words, two databases belong in the same block of the partition if and only if their output for $\mathbf{Q}$ is indistinguishable. We use the standard notation $[D]_\mathbf{Q}$ to denote the equivalence class in which D belongs; in other words, $[D]_\mathbf{Q} = \{D' \in \mathcal{I} \mid \mathbf{Q}(D') = \mathbf{Q}(D)\}$. For two partitions $\mathcal{P}_1, \mathcal{P}_2$, we say that $\mathcal{P}_1$ *refines* $\mathcal{P}_2$, and write $\mathcal{P}_1 \succeq \mathcal{P}_2$, if every block of $\mathcal{P}_1$ is a subset of some block in $\mathcal{P}_2$. In other words, $\mathcal{P}_1$ is a more fine-grained partition of $\mathcal{I}$ than $\mathcal{P}_2$.

Lattices and Join-Semilattices. A *join-semilattice* $(L, \leq)$ is a partially ordered set in which every two elements in L have a unique supremum (called *join* and denoted as $\vee$). A *lattice* $(L, \leq)$ is a partially ordered set in which every two elements in L have both a unique supremum, and a unique infimum (called *meet* and denoted $\wedge$). In this paper, we will consider two different join-semilattices. The first semilattice has elements subsets of $\mathcal{I}$, which are ordered by subset inclusion $\subseteq$. The second semilattice has elements partitions of $\mathcal{I}$, which are ordered by the refinement relation $\preceq$.

Let $f : L \to \mathbb{R}$ be a function defined on the elements of the join-semilattice. We say that f is *monotone*, or *isotone*, if whenever $A \leq B$, then $f(A) \leq f(B)$. Moreover, we say that f is *subadditive* if for any two elements A, B of the semilattice we have $f(A \vee B) \leq f(A) + f(B)$.

2.2 The Pricing Framework

In our setting, a data seller offers a database instance D for sale. Data buyers can issue queries on the database in the form of query bundles $\mathbf{Q}$. For each query $\mathbf{Q}$ over the instance D, the task in hand is to assign a *price* to the query answer $\mathbf{Q}(D)$ that reflects the amount of information gained by the data buyer. When a price is assigned to a query bundle $\mathbf{Q}$, we can differentiate between three different pricing strategies, which depend on the parameters

used to compute the price. There are three possible parameters we can use to determine the price of a query: the query bundle $\mathbf{Q}$, the answer of the query on the database D, denoted $E = \mathbf{Q}(D)$, and the database D itself. The price will obviously depend on which query $\mathbf{Q}$ we issue, but there is a choice of which D, E should be further used to compute the price. This choice defines three different classes of pricing schemes:

- **Instance-independent (QPS):** the price depends only on $\mathbf{Q}$, in which case the pricing function is of the form $p(\mathbf{Q})$. The price is independent of the underlying data.
- **Answer-dependent (APS):** the price depends on the answer $E = \mathbf{Q}(D)$, so the price is of the form $p(\mathbf{Q}, E)$. In this case, the price depends on the query and the query output.
- **Data-dependent (DPS):** the price depends on the underlying database D, so the pricing function is of the form $p(\mathbf{Q}, D)$.

Any instance-independent scheme can be cast as an answer-dependent scheme, and any answer-dependent scheme as a data-dependent scheme. The distinction between APS and DPS was introduced in [20], where the authors use the terminology *delayed pricing* and *up-front pricing* respectively. Notice that both in QPS and APS the prices themselves do not leak any information about the underlying data D.[1] In contrast, a data-dependent pricing scheme can leak information about the data (for more details see [20]). For this reason, in this paper we focus on the first two types of pricing schemes: QPS and APS.

The reason we consider query bundles in our setting is that in practice a data buyer will issue over time a sequence $\mathbf{Q}_1, \ldots, \mathbf{Q}_m$ of query bundles on the database. In this case, after issuing the first i queries, the data buyer should not be charged a price of $\sum_i p(\mathbf{Q}_i, D)$, but instead $p(\mathbf{Q}_1, \ldots, \mathbf{Q}_i, D)$. Notice here that, even if a user issues only single queries, we still need to be able to price a query bundle.

2.3 Arbitrage Conditions

Assigning prices to query bundles without any restrictions can lead to the occurrence of arbitrage opportunities. In [15], the authors presented a single condition that captures arbitrage. Here, we follow [20], and consider independently two different conditions where arbitrage may occur.

Information Arbitrage. The first condition captures the intuition that the price of query bundle must capture the amount of information that an answer reveals about the actual database D. In particular, if a query bundle $\mathbf{Q}_1$ reveals a subset of information than a query bundle $\mathbf{Q}_2$ reveals, the price of $\mathbf{Q}_1$ must be less than the price of $\mathbf{Q}_2$. If this condition is not satisfied, it creates an arbitrage opportunity, since a data buyer can purchase $\mathbf{Q}_2$ instead, and use it to obtain the answer of $\mathbf{Q}_1$ for a cheaper price.

Bundle Arbitrage. The second condition regards the scenario where a data buyer that wants to obtain the answer for the bundle $\mathbf{Q} = \mathbf{Q}_1, \mathbf{Q}_2$ creates two separate accounts, and uses one to ask for $\mathbf{Q}_1$ and the other to ask for $\mathbf{Q}_2$. To avoid such an arbitrage situation, we must make sure that the price of $\mathbf{Q}$ is at most the sum of the prices for $\mathbf{Q}_1$ and $\mathbf{Q}_2$. [20] uses the terminology *separate-account arbitrage* to refer to this arbitrage condition.

We will show in the next sections how to mathematically formalize information arbitrage and bundle arbitrage for both APS and QPS.

[1] For the case of answer-dependent prices, we must make sure that we reveal the price only if we are certain that the buyer will be charged for the cost.

3 Answer-Dependent Pricing

In this section, we study the design of *answer-dependent* pricing schemes. In an APS the pricing function takes the form $p(\mathbf{Q}, E)$, where $\mathbf{Q}$ is a query bundle and $E \in \{\mathbf{Q}(D) \mid D \in \mathcal{I}\}$. Throughout the section, we assume that query bundles belong to some query language $\mathcal{L}$. We first discuss how to formalize the arbitrage conditions. To formally describe information arbitrage, we use the notion of *data-dependent determinacy*.

▶ **Definition 1.** We say that $\mathbf{Q}_2$ determines $\mathbf{Q}_1$ under database D, denoted $D \vdash \mathbf{Q}_2 \twoheadrightarrow \mathbf{Q}_1$ if for every database D' such that $\mathbf{Q}_2(D) = \mathbf{Q}_2(D')$, we also have $\mathbf{Q}_1(D') = \mathbf{Q}_1(D)$.

The above definition of determinacy is different from *query determinacy* [23, 24], since it is defined with respect to a given database D. It is also easy to see that if $D \vdash \mathbf{Q}_2 \twoheadrightarrow \mathbf{Q}_1$, we also have that $D' \vdash \mathbf{Q}_2 \twoheadrightarrow \mathbf{Q}_1$ for any database D' such that $\mathbf{Q}_2(D) = \mathbf{Q}_2(D')$.

▶ **Definition 2** (APS Information Arbitrage)**.** Let $\mathbf{Q}_1, \mathbf{Q}_2$ be two query bundles. We say that the pricing function p has no *information arbitrage* if for every database $D \in \mathcal{I}$, $D \vdash \mathbf{Q}_2 \twoheadrightarrow \mathbf{Q}_1$ implies that $p(\mathbf{Q}_2, E_2) \geq p(\mathbf{Q}_1, E_1)$, where $E_i = \mathbf{Q}_i(D)$ for $i = 1, 2$.

This definition of information arbitrage captures both *post-processing arbitrage* and *serendipitous arbitrage*, as these are defined in [20]. For the case of bundle arbitrage, we formalize it as follows.

▶ **Definition 3** (APS Bundle arbitrage)**.** Let the query bundle $\mathbf{Q} = \mathbf{Q}_1, \mathbf{Q}_2$. We say that the price function p has no *bundle arbitrage* if for every database $D \in \mathcal{I}$, we have $p(\mathbf{Q}, E) \leq p(\mathbf{Q}_1, E_1) + p(\mathbf{Q}_2, E_2)$, where $E = \mathbf{Q}(D)$ and $E_i = \mathbf{Q}_i(D)$ for $i = 1, 2$.

We say that an answer-dependent pricing function is *arbitrage-free* if it has no information arbitrage and no bundle arbitrage.

3.1 How to Find a Pricing Function

In this section, we characterize the family of answer-dependent pricing functions that satisfy both arbitrage conditions. The critical component is the notion of a *conflict set*.

3.1.1 Conflict Sets

Consider a query bundle $\mathbf{Q} \in B(\mathcal{L})$, a database $D \in \mathcal{I}$ and let $E = \mathbf{Q}(D)$. We define

$$\mathcal{S}_{\mathbf{Q}}(E) = \{D' \in \mathcal{I} \mid \mathbf{Q}(D') = E\}, \qquad \overline{\mathcal{S}}_{\mathbf{Q}}(E) = \{D' \in \mathcal{I} \mid \mathbf{Q}(D') \neq E\}$$

In other words, $\mathcal{S}_{\mathbf{Q}}(E)$ computes the set of databases that "agree" with the view extension E, and $\overline{\mathcal{S}}_{\mathbf{Q}}(E)$ contains the complement set, i.e. the set of databases that "disagree" with E. Notice that $\mathcal{S}_{\mathbf{Q}}(\mathbf{Q}(D)) = [D]_{\mathbf{Q}}$. We refer to $\overline{\mathcal{S}}_{\mathbf{Q}}(E)$ as the *conflict set* for query $\mathbf{Q}$ and extension E, while we refer to $\mathcal{S}_{\mathbf{Q}}(E)$ as the *agreement set*. It is straightforward that $\overline{\mathcal{S}}_{\mathbf{Q}}(E) = \mathcal{I} \setminus \mathcal{S}_{\mathbf{Q}}(E)$.

▶ **Example 4.** We will use the following scenario as a running example throughout this section. Suppose that we have a binary relation $R(\underline{A}, B)$, where attribute A is the key. The values of the n keys are also publicly known $\{a_1, a_2, \ldots, a_n\}$. Moreover, assume that B can take two possible values from $\{0, 1\}$. It is easy to see that $\mathcal{I}$ consists of 2^n databases. For $n = 2$, let D_{ij} denote the database $\{(a_1, i), (a_2, j)\}$. For example $D_{01} = \{(a_1, 0), (a_2, 1)\}$.

Consider now the query $Q(x) = R(a_1, x)$, which asks for value of attribute B for the tuple with key $A = a_1$. Assume that the underlying database is D_{01}. The conflict set of Q and $E = Q(D_{01})$ consists of all databases D for which $(a_1, 1) \in D$, hence $\mathcal{S}_Q(E) = \{D_{10}, D_{11}\}$.

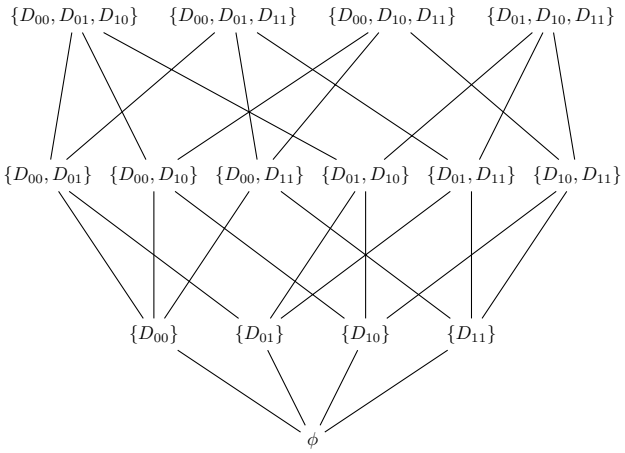

Figure 1 A simultaneous depiction of the join-semilattices for the four databases in Example 4.

If $\mathbf{Q}$ returns a constant answer for every database in $\mathcal{I}$, the conflict set will be the empty set. On the other hand, if $\mathbf{Q}$ reveals the whole database D, the conflict set will be $\mathcal{I} \setminus \{D\}$. We can now define the set of all possible conflict sets for a database D and a given language $\mathcal{L}$ as $\mathcal{S}_D^{\mathcal{L}} = \{\overline{\mathcal{S}}_{\mathbf{Q}}(\mathbf{Q}(D)) \mid \mathbf{Q} \in B(\mathcal{L})\}$. The following lemma, which we prove in [8], shows that $\mathcal{S}_D^{\mathcal{L}}$ forms a *join-semilattice* under the partial order $\subseteq$, where the join operator is set union.

▶ **Lemma 5.** *Let $\mathbf{Q} = \mathbf{Q}_1, \mathbf{Q}_2$. For a database $D \in \mathcal{I}$, let $E_1 = \mathbf{Q}_1(D)$, $E_2 = \mathbf{Q}_2(D)$, and $E = \mathbf{Q}(D)$. Then, $\overline{\mathcal{S}}_{\mathbf{Q}}(E) = \overline{\mathcal{S}}_{\mathbf{Q}_1}(E_1) \cup \overline{\mathcal{S}}_{\mathbf{Q}_2}(E_2)$.*

The diagram in Figure 1 depicts simultaneously the four join-semilattices for each of the databases in Example 4. We next show in [8] the following lemma that connects the notion of a conflict set with data-dependent determinacy.

▶ **Lemma 6.** *Let $\mathbf{Q}_1, \mathbf{Q}_2$ be two query bundles, and $D \in \mathcal{I}$ be a database. Let $E_i = \mathbf{Q}_i(D)$ for $i = 1, 2$. The following two statements are equivalent:*

1. $D \vdash \mathbf{Q}_2 \twoheadrightarrow \mathbf{Q}_1$
2. $\overline{\mathcal{S}}_{\mathbf{Q}_2}(E_2) \supseteq \overline{\mathcal{S}}_{\mathbf{Q}_1}(E_1)$

Lemma 6 and Lemma 5 demonstrate that information and bundle arbitrage can be cast as conditions on the elements of the semilattice of conflict sets.

▶ **Example 7.** Continuing Example 4, consider the queries $Q_1(x) = R(a_1, x)$ and $Q_2() = R(x, 1)$. Let D_{00} be the underlying database. It is easy to see that $D_{00} \vdash Q_2 \twoheadrightarrow Q_1$, since after asking Q_2 we learn that the database contains no 1 values for B, and thus it must have only 0 values. The conflict sets for $E_1 = Q_1(D_{00})$, $E_2 = Q_2(D_{00})$ are $\overline{\mathcal{S}}_{Q_1}(E_1) = \{D_{11}, D_{10}\}$ and $\overline{\mathcal{S}}_{Q_2}(E_2) = \{D_{01}, D_{10}, D_{11}\}$ respectively.

3.1.2 A Characterization of Arbitrage-Free APS

We can now use the notion of a conflict set to define pricing functions of the form $p(\mathbf{Q}, E) = f(\overline{\mathcal{S}}_{\mathbf{Q}}(E))$, where $f : 2^{\mathcal{I}} \setminus \{I\} \to \mathbb{R}_+$ is a *set function*. It is straightforward to see that such a pricing function is by construction in APS, since the computation depends only on $\mathbf{Q}$ and E, and not on the database D. For example, if $\mathbf{Q}$ returns a constant answer for every database in $\mathcal{I}$, $p(\mathbf{Q}, E) = f(\emptyset)$. On the other hand, if $\mathbf{Q}$ reveals the whole database

D, $p(\mathbf{Q}, E) = f(\mathcal{I} \setminus \{D\})$. We can now show a necessary and sufficient characterization of answer-dependent functions with no information arbitrage in terms of such a function f.

▶ **Theorem 8.** *Let p be an answer-dependent pricing function. The following two statements are equivalent:*

1. *p has no information arbitrage.*
2. *$p(\mathbf{Q}, E) = f(\overline{\mathcal{S}}_{\mathbf{Q}}(E))$, where f is a monotone function over every semilattice $\mathcal{S}_D^{\mathcal{L}}$.*

We have shown that in order to avoid information arbitrage it suffices to restrict the function to be monotone. We next demonstrate a similar connection of bundle arbitrage to the property of subadditivity.

▶ **Theorem 9.** *Let $p(\mathbf{Q}, E) = f(\overline{\mathcal{S}}_{\mathbf{Q}}(E))$ be a pricing function, where f is a set function. Then, the following two statements are equivalent:*

1. *p has no bundle arbitrage.*
2. *f is subadditive over every semilattice $\mathcal{S}_D^{\mathcal{L}}$.*

Both Theorem 8 and Theorem 9 are proved in the full version of the paper [8]. Observe that if a function f is monotone and subadditive over $2^{\mathcal{I}}$, it will also be monotone and subadditive over every semilattice $\mathcal{S}_D^{\mathcal{L}}$. Hence, as a corollary we can describe a general family of arbitrage-free pricing functions.

▶ **Corollary 10.** *Let f be a monotone and subadditive set function f. Then, the function $p(\mathbf{Q}, E) = f(\overline{\mathcal{S}}_{\mathbf{Q}}(E))$ is an answer-dependent pricing function that is arbitrage-free.*

3.2 Explicit Constructions of Pricing Functions

We have so far described a general class of functions that are both information and bundle arbitrage-free. Since any submodular function is also subadditive, any monotone submodular set function f will also produce a desired pricing function. We give some concrete examples of arbitrage-free pricing functions below.

▶ **Corollary 11.** *Suppose that we assign a weight of w_D to each $D \in \mathcal{I}$, such that $\sum_{D \in \mathcal{I}} w_D < \infty$. Then, the following pricing functions are arbitrage-free:*

1. *the weighted coverage function: $\sum_{D:\mathbf{Q}(D) \neq E} w_D$.*
2. *the supremum function: $\sup_{D:\mathbf{Q}(D) \neq E} w_D$.*[2]
3. *the budget-limited weighted coverage function for some $B \geq 0$: $\min\{B, \sum_{D:\mathbf{Q}(D) \neq E} w_D\}$.*

We can construct richer pricing functions by combining the weighted coverage function with a concave function g. Indeed, we can show that $p(\mathbf{Q}, E) = g(\sum_{D \in \overline{\mathcal{S}}_{\mathbf{Q}}(E)} w_D)$ is arbitrage-free for any concave function g. If $\mathcal{I}$ is finite, we can assign to each database $D \in \mathcal{I}$ an equal weight, in which case we obtain the arbitrage-free function $p(\mathbf{Q}, E) = g(|\overline{\mathcal{S}}_{\mathbf{Q}}(E)|)$.

▶ **Corollary 12.** *Suppose that we assign a weight of w_D to each $D \in \mathcal{I}$, such that $\sum_{D \in \mathcal{I}} w_D < \infty$. Then, the pricing function $p(\mathbf{Q}, E) = g(\sum_{D \in \overline{\mathcal{S}}_{\mathbf{Q}}(E)} w_D)$ is arbitrage-free for any concave function g.*

Proof. We know that if $f(A)$ is a modular set function and g is concave, then $g(f(A))$ is a submodular function. Notice that $f(A) = \sum_{i \in A} w_i$ is a modular function for any choice of weights w_i. ◀

[2] The supremum becomes equivalent to the max function if $\mathcal{I}$ is finite.

The pricing functions we have presented thus far are constructed by assigning a weight to each database in $\mathcal{I}$. Another type of construction starts by specifying a family $\mathcal{F}$ of subsets of $\mathcal{I}$. For each subset $S \in \mathcal{F}$, we assign a weight w_S. Finally, we pick some real number $B \geq \max_{S \in \mathcal{F}} w_S$. We define the *weighted set cover* function $f(A)$ as the cost of the minimum set cover for A if such a set exists, otherwise $f(A) = B$.

▶ **Lemma 13.** *The weighted set cover pricing function is arbitrage-free.*

The weighted set cover function generalizes the approach from [15], where explicit prices are specified for certain views, and the price of the query is computed as the cheapest set of views that determine the query. Indeed, if we are given explicit price points $(\mathbf{Q}_i, p_i)$ for $i = 1, \ldots, m$, we can define the following family of sets: $\mathcal{F} = \{\overline{\mathcal{S}}_{\mathbf{Q}_i}(\mathbf{Q}_i(D)) \mid i = 1, \ldots, m\}$, where each set $\overline{\mathcal{S}}_{\mathbf{Q}_i}(\mathbf{Q}_i(D))$ is assigned a weight of p_i. Since $D \vdash \mathbf{Q}_{i_1}, \ldots, \mathbf{Q}_{i_\ell} \twoheadrightarrow \mathbf{Q}$ is equivalent to saying that the union of the conflict sets of $\mathbf{Q}_{i_1}, \ldots, \mathbf{Q}_{i_\ell}$ is a superset of the conflict set of $\mathbf{Q}$, the minimum set cover for $\overline{\mathcal{S}}_{\mathbf{Q}}(E)$ corresponds to the cheapest set of views that determine $\mathbf{Q}$ under database D.

3.2.1 Information Gain as a Pricing Function

A natural mechanism for pricing is to start from a probabilistic point of view and compute the price as the reduction in uncertainty, or *information gain*, using some notion of entropy.

Formally, consider an initial probability distribution over the set $\mathcal{I}$ of possible databases: in other words, assign a probability p_D to each database $D \in \mathcal{I}$. This probability distribution may reflect public information about the database (for example some value might be more probable than some other value). Let X be a random variable such that $P(X = D) = p_D$. Given some entropy measure $H(\cdot)$ of a random variable, such as Shannon entropy or min-entropy, we can set the price as the *information gain*: the initial entropy $H(X)$ minus the entropy of the new distribution, which is now conditioned on the event $\mathbf{Q}(X) = E$. Formally, we define the price as $p(\mathbf{Q}, E) = H(X) - H(X \mid \mathbf{Q}(X) = E)$. We can now plug standard uncertainty measures to obtain a pricing function. For example, we can use the Shannon entropy $H(X) = -\sum_{D \in \mathcal{I}} p_D \log(p_D)$, or the min-entropy $H_\infty(X) = -\log(\max_D p_D)$.

▶ **Lemma 14.** *There exists a probability distribution p_D over $\mathcal{I}$ such that the answer-dependent entropy function has information-arbitrage.*

Proof. Consider two sets $B \subseteq A \subseteq \mathcal{I}$, such that $A \setminus B = \{D_0\}$. Assume that the probabilities are set as follows: for every $D \in B$ we have $p_D = \epsilon$, and $p_{D_0} = 1 - m\epsilon$, where $m = |A|$. Define now two queries $\mathbf{Q}_A$ and $\mathbf{Q}_B$ such that $\mathcal{S}_{\mathbf{Q}_A}(E) = A$ and $\mathcal{S}_{\mathbf{Q}_B}(E) = B$. In this case, we have:

$$p(\mathbf{Q}_B, E) = H(D) + \sum_{i=1}^{m} \frac{1}{m} \log(1/m) = H(D) - \log(m)$$

$$p(\mathbf{Q}_A, E) = H(D) + m\epsilon \log(\epsilon) + (1 - m\epsilon) \log(1 - m\epsilon)$$

Further, $0 < m\epsilon < 1$. To create a counterexample, we choose $m\epsilon = \frac{1}{2}$, and now we have:

$$\begin{aligned} p(\mathbf{Q}_A, E) - p(\mathbf{Q}_B, E) &= \\ &= m\epsilon \log(\epsilon) + (1 - m\epsilon) \log(1 - m\epsilon) + \log(m) \\ &= \frac{1}{2} \log(\epsilon) - \frac{1}{2} + \log(m) = \frac{1}{2} \log(m) - 1 \end{aligned}$$

By picking m large enough, we can make this quantity strictly positive, hence violating the information arbitrage condition. ◀

The intuition in the above proof is the following: the result for query $\mathbf{Q}_A$ will have a somewhat small entropy, because D_0 is much more probable than the other databases. However, by asking $\mathbf{Q}_B$ we learn that D_0 cannot be the actual database, and now the probability is equally distributed among the rest of the candidates; hence, the entropy grows!

The information gain, even though it seems a natural candidate, is not a well-behaved pricing function for APS, since it exhibits both information and bundle arbitrage (see Lemma 14 for such an example of information arbitrage). As we will see in Section 4 though, we can use information gain to construct arbitrage-free functions for QPS. In the case where the probabilities p_D are all equal, the information gain based on Shannon entropy has no information arbitrage (but can still exhibit bundle arbitrage) as shown in [8].

▶ **Lemma 15.** *If the probability distribution p_D over $\mathcal{I}$ is uniform, the information gain based on Shannon entropy has no information arbitrage.*

3.3 A Tradeoff for Arbitrage-Free APS

▶ **Example 16.** Continuing Example 4, consider the query $Q(x) = R(a, x)$ and the pricing function $p_2(\mathbf{Q}, E) = \log(|\overline{\mathcal{S}}_{\mathbf{Q}}(E)|)$. Notice that, independent of the actual database D, the conflict set has always size 2^{n-1}. In this case, $p_2(Q, E) = n - 1$. Notice that the price for learning the whole database is $\log(2^n - 1)$, which means that for learning a single tuple we pay almost as much as the whole database.

We will show here that the above example is not a random occurrence, and that the requirement that a pricing function has no bundle arbitrage gives rise to the phenomenon of assigning high prices (w.r.t. to the price of the whole dataset) to queries that reveal only a small amount of information.

▶ **Lemma 17.** *Let $p(\mathbf{Q}, E) = f(\overline{\mathcal{S}}_{\mathbf{Q}}(E))$ be an answer-dependent pricing function where f is monotone and subadditive over $2^{\mathcal{I}}$. Then, for every non-constant query $\mathbf{Q} \in B(\mathcal{L})$ there exists a database $D \in \mathcal{I}$ such that $p(\mathbf{Q}, \mathbf{Q}(D))$ is at least half the price of D.*

To see that the bundle-arbitrage requirement cause the problem, consider the function $p(\mathbf{Q}, E) = \log(|\mathcal{I}|) - \log(|\mathcal{S}_{\mathbf{Q}}(E)|)$, for which we showed that it exhibits no information arbitrage, but can still have bundle arbitrage. Continuing our example, we can see that $p(Q, E) = \log(2^n) - \log(2^{n-1}) = 1$; thus, learning about one of the n tuples is priced reasonably to $1/n$ of the price of the whole database. Our analysis demonstrates an important tradeoff in the design space of answer-dependent pricing functions: *ensuring no bundle arbitrage implies that the pricing function will charge disproportionately high prices for little information.*

It is also instructive to note that while Lemma 17 guarantees that existence of database $D \in \mathcal{I}$ that behaves badly, it does not say anything about the number of such databases. In fact, for our example we can show that for query Q at least half of the databases in $\mathcal{I}$ will exhibit this undesirable behavior.

4 Instance-Independent Pricing

We study here the structure of *instance-independent* pricing schemes. In a QPS, the pricing function is of the form $p(\mathbf{Q})$, depending only on the query. We first formalize the conditions under which the pricing function has no information arbitrage and no bundle arbitrage.

▶ **Definition 18.** We say that $\mathbf{Q}_2$ determines $\mathbf{Q}_1$, denoted $\mathbf{Q}_2 \twoheadrightarrow \mathbf{Q}_1$, if for every database database D' and D'', $\mathbf{Q}_2(D') = \mathbf{Q}_2(D'')$ implies $\mathbf{Q}_1(D') = \mathbf{Q}_1(D'')$.

In contrast to answer-dependent pricing functions, where we used a notion of determinacy that depends on the database, here we use the standard notion of *information-theoretic determinacy*.[3] We can now describe the formal definition for information arbitrage.

▶ **Definition 19** (QPS Information Arbitrage)**.** The pricing function p has no *information arbitrage* if for any two query bundles $\mathbf{Q}_1, \mathbf{Q}_2$ such that $\mathbf{Q}_2 \twoheadrightarrow \mathbf{Q}_1$, we have $p(\mathbf{Q}_2) \geq p(\mathbf{Q}_1)$.

▶ **Definition 20** (QPS Bundle arbitrage)**.** Let the query bundle $\mathbf{Q} = \mathbf{Q}_1, \mathbf{Q}_2$. We say that the pricing function p has no *bundle arbitrage* if we have $p(\mathbf{Q}) \leq p(\mathbf{Q}_1) + p(\mathbf{Q}_2)$.

4.1 Serendipitous Arbitrage

Consider two query bundles $\mathbf{Q}_1$ and $\mathbf{Q}_2$ such that $\mathbf{Q}_1 \not\twoheadrightarrow \mathbf{Q}_2$, but for some $D \in \mathcal{I}$, $D \vdash \mathbf{Q}_1 \twoheadrightarrow \mathbf{Q}_2$. For example, consider the boolean query $Q_1() = R(x, y)$ over the binary relation $R(A, B)$. Let $Q_2(x, y) = R(x, y)$. Clearly, for all databases D other than the empty database, $D \vdash Q_1 \not\twoheadrightarrow Q_2$. However, for the database $D_0 = \emptyset$, note that $D_0 \vdash Q_1 \twoheadrightarrow Q_2$. In this case, if $p(Q_1) > p(Q_2)$, the data buyer would have an arbitrage opportunity. However, this opportunity would arise by chance, since the buyer does not know the underlying database and thus does not know that asking for $\mathbf{Q}_2$ can lead to learning $\mathbf{Q}_1$ for a lower price. We call this phenomenon *serendipitous arbitrage* [20]. Our definition of QPS information arbitrage does not capture serendipitous arbitrage. The next result demonstrates a second tradeoff in the design space of pricing functions: *any non-trivial QPS will exhibit serendipitous arbitrage.* We prove in [8]

▶ **Theorem 21.** *Let $\mathcal{L} = UCQ$. If a QPS exhibits no serendipitous arbitrage, then the price of any non-constant query bundle $\mathbf{Q}$ is equal to the price of asking for the whole database.*

4.2 How to Find a Pricing Function

To characterize the structure of instance-independent pricing functions, we exploit the fact that we can equivalently view a query as a *partition* of the set of possible databases $\mathcal{I}$.

4.2.1 The Partition Lattice

Fix some query language $\mathcal{L}$. Recall that for a query bundle $\mathbf{Q} \in B(\mathcal{L})$, $\mathcal{P}_{\mathbf{Q}}$ is the partition that is induced by the following *equivalence relation*: $D \sim D'$ iff $\mathbf{Q}(D) = \mathbf{Q}(D')$.

▶ **Lemma 22.** *Let $\mathbf{Q}_1, \mathbf{Q}_2 \in \mathcal{L}$ be two query bundles. The following are equivalent:*

1. $\mathbf{Q}_1 \twoheadrightarrow \mathbf{Q}_2$
2. $\mathcal{P}_{\mathbf{Q}_1} \succeq \mathcal{P}_{\mathbf{Q}_2}$*, i.e.* $\mathcal{P}_{\mathbf{Q}_1}$ *refines* $\mathcal{P}_{\mathbf{Q}_2}$

The refinement relation defines a partial order on the set $\Pi_{\mathcal{I}}^{\mathcal{L}}$ of all partitions of $\mathcal{I}$ induced by any bundle $\mathbf{Q} \in B(\mathcal{L})$. An equivalent way to define the partial order is through the *distinction set* of a partition $dit(\mathcal{P}) = \bigcup_{B,B' \in \mathcal{P}: B \neq B'} B \times B'$. Intuitively, the distinction set contains all pairs of elements that are not in the equivalence relation. It is straightforward to see that $dit(\mathcal{P}_{\mathbf{Q}}) = \{(D', D'') \in \mathcal{I} \times \mathcal{I} \mid \mathbf{Q}(D') \neq \mathbf{Q}(D'')\}$. Furthermore, $\mathcal{P}_1 \succeq \mathcal{P}_2$ if and only if $dit(\mathcal{P}_1) \supseteq dit(\mathcal{P}_2)$ and thus one can use the inclusion of the distinction sets to define a partial order on the partitions.

[3] Here we should note that there exists a slight difference, since the databases we consider can come only from $\mathcal{I}$, and not be any database.

The partial order induced by $\succeq$ on $\Pi^{\mathcal{L}}_{\mathcal{I}}$ forms a *join-semilattice*. The bottom element of the semilattice is the partition $\{\mathcal{I}\}$, which corresponds to a query that returns a constant answer. The top element is the partition where each block is a singleton set: this corresponds to a query that informs about the whole database. The *join* $\mathcal{P}_1 \vee \mathcal{P}_2$ is a new partition whose blocks are the non-empty intersections of any two blocks from $\mathcal{P}_1, \mathcal{P}_2$. The lemma below proves that the algebraic structure we defined is indeed a semilattice.

▶ **Lemma 23.** *Let $\mathbf{Q} = \mathbf{Q}_1, \mathbf{Q}_2$, where $\mathbf{Q}_1, \mathbf{Q}_2 \in B(\mathcal{L})$. Then, $\mathcal{P}_{\mathbf{Q}} = \mathcal{P}_{\mathbf{Q}_1} \vee \mathcal{P}_{\mathbf{Q}_2}$.*

If we define the partial order as the inclusion of distinction sets, $dit(\mathcal{P}_{\mathbf{Q}}) = dit(\mathcal{P}_{\mathbf{Q}_1} \vee \mathcal{P}_{\mathbf{Q}_2}) = dit(\mathcal{P}_{\mathbf{Q}_1}) \cup dit(\mathcal{P}_{\mathbf{Q}_2})$, the join operator is simply the union of the distinction sets.

4.2.2 A Characterization of Arbitrage-Free QPS

We now consider the family of instance-independent pricing functions of the form $p(\mathbf{Q}) = f(\mathcal{P}_{\mathbf{Q}})$, where $f : \Pi^{\mathcal{L}}_{\mathcal{I}} \to \mathbb{R}_+$ is a function that maps a partition to the positive real numbers.

▶ **Theorem 24.** *Let p be an instance-independent pricing function. Then, the two statements are equivalent:*

1. *p has no information arbitrage.*
2. *$p(\mathbf{Q}) = f(\mathcal{P}_{\mathbf{Q}})$, where f is a monotone function over $\Pi^{\mathcal{L}}_{\mathcal{I}}$.*

▶ **Theorem 25.** *Let $p(\mathbf{Q}) = f(\mathcal{P}_{\mathbf{Q}})$ be an instance-independent pricing function, where f is a function over $\Pi^{\mathcal{L}}_{\mathcal{I}}$. Then, the two statements are equivalent:*

1. *p has no bundle arbitrage.*
2. *f is subadditive over $\Pi^{\mathcal{L}}_{\mathcal{I}}$.*

▶ **Corollary 26.** *Let f be a monotone and subadditive function over $\Pi^{\mathcal{L}}_{\mathcal{I}}$. Then, $p(\mathbf{Q}) = f(\mathcal{P}_{\mathbf{Q}})$ is an instance-independent pricing function that has no bundle or information arbitrage.*

Alternatively, we could also define the pricing function as $p(\mathbf{Q}) = f(dit(\mathcal{P}_{\mathbf{Q}}))$. Using the same type of arguments, we can show:

▶ **Corollary 27.** *Let f be a monotone and subadditive set function. Then, $p(\mathbf{Q}) = f(dit(\mathcal{P}_{\mathbf{Q}}))$ is an instance-independent pricing function that has no bundle or information arbitrage.*

4.3 Construction of Pricing Functions From Answer-Dependent Prices

We show first how we can design an instance-independent pricing function $p(\mathbf{Q})$ starting from an answer-dependent function $p(\mathbf{Q}, E)$. Given a query bundle $\mathbf{Q}$, the idea is to construct a vector of all prices $p(\mathbf{Q}, \mathbf{Q}(D))$ for all databases $D \in \mathcal{I}$. Formally, we define the *price vector* $\vec{p}(\mathbf{Q}) = \langle p(\mathbf{Q}, \mathbf{Q}(D)) \mid D \in \mathcal{I} \rangle$. Then we can obtain an instance-independent pricing function by computing another function $g : \mathbb{R}_+^{|\mathcal{I}|} \to \mathbb{R}_+$ over the above vector, such that $p(\mathbf{Q}) = g(\vec{p}(\mathbf{Q}))$. The next lemma describes the conditions for g under which the arbitrage-free property carries over.

▶ **Lemma 28.** *Let $p(\mathbf{Q}, E)$ be an arbitrage-free pricing function. If g is a monotone and subadditive function, then $p(\mathbf{Q}) = g(\vec{p}(\mathbf{Q}))$ is an arbitrage-free instance-independent function.*

We next present an application of Lemma 28 to obtain arbitrage-free pricing functions.

▶ **Lemma 29.** *Let f be a monotone and subadditive set function. Let w_D be a non-negative weight w_D to each $D \in \mathcal{I}$, and denote $w_B = \sum_{D \in B} w_D$. Then, the pricing functions $p_1(\mathbf{Q}) = \max_{B \in \mathcal{P}_{\mathbf{Q}}}\{f(\mathcal{I} \setminus B)\}$ and $p_2(\mathbf{Q}) = \sum_{B \in \mathcal{P}_{\mathbf{Q}}} w_B \cdot f(\mathcal{I} \setminus B)$ are arbitrage-free.*

▶ **Example 30.** Consider the function p_2 with equal weights $w_D = 1$ and the set function $f(A) = |A|$. The resulting arbitrage-free function is $p(\mathbf{Q}) = \sum_{B \in \mathcal{P}_\mathbf{Q}} |B|(|\mathcal{I}| - |B|) = |dit(\mathcal{P}_\mathbf{Q})|$, which sets the price to be the size of the distinction set.

If $\sum_D w_D = 1$ for p_2, one can interpret the weights as a probability distribution over the set of databases $\mathcal{I}$. In this case, we can write $p_2(\mathbf{Q}) = \mathbb{E}_{B \in \mathcal{P}_\mathbf{Q}}[f(\mathcal{I} \setminus B)]$, where each block B has probability w_B. In other words, the pricing function is the expected price over all answer-dependent prices. The converse of Lemma 28 does not hold: it is possible for $p(\mathbf{Q})$ to be arbitrage-free, and for some database D it may not be the case. As we will see next, this allows us to construct arbitrage-free functions that are based on measures of uncertainty.

4.4 Construction of Pricing Functions From Uncertainty Measures

In this section, we describe arbitrage-free pricing functions that do not originate from answer-dependent functions. To construct such functions, we switch to a probabilistic view of the problem and then apply information-theoretic tools that are used to measure uncertainty. For the remainder of this section, we assume that each database D is associated with a probability p_D. We denote by X the random variable such that $P(X = D) = p_D$ and let $p_E = \sum_{D:\mathbf{Q}(D)=E} p_D$. The detailed proofs in this section are presented in [8].

Shannon Entropy. The first measure of uncertainty we apply is the most commonly used form of entropy, and was proposed in [20] as a pricing function. In the answer-dependent context, we defined the price as the information gain after the output E has been revealed. Since in this setting the price is independent of the output, we define the price as the *expected information gain* over all possible outcomes. Formally:

$$p^H(\mathbf{Q}) = H(X) - \sum_E p_E \cdot H(X \mid \mathbf{Q}(X) = E) \tag{1}$$

Equivalently, we can also express the price as

$$p^H(\mathbf{Q}) = H(X) - H(X \mid \mathbf{Q}(X)) = I(X; \mathbf{Q}(X)) = H(\mathbf{Q}(X)) - H(\mathbf{Q}(X) \mid X) = H(\mathbf{Q}(X))$$

where $I(X;Y)$ is the *mutual information* between the random variables X and Y. [20] proves that p^H is both bundle and information arbitrage-free, using the subadditivity of entropy and the data-processing inequality respectively. It is instructing to write p^H as $p^H(\mathbf{Q}) = -\sum_{S \in \mathcal{P}_\mathbf{Q}} p_S \cdot \log p_S = \sum_D p_D \cdot p(\mathbf{Q}, \mathbf{Q}(D))$ where $p(\mathbf{Q}, E) = -\log(p_E)$ is now an answer-dependent pricing function. Notice that $p(\mathbf{Q}, E)$ has no information arbitrage, and thus by applying Lemma 28 we get an alternative proof that p^H is information arbitrage-free. However, $p(\mathbf{Q}, E)$ can have bundle arbitrage, and thus we cannot apply Lemma 28 to show the subadditivity property as well: entropy is subadditive only in expectation. This example demonstrates that the converse of Lemma 28 does not hold.

Tsallis Entropy. For a real number $q > 1$, the *Tsallis entropy* [27], or *q-entropy*, of a random variable X is defined as $S_q(X) = \frac{1}{q-1} \cdot \left(1 - \sum_x P(X = x)^{q-1}\right)$. Tsallis entropy is a generalization of Shannon entropy, since $\lim_{q \to 1} S_q(X) = H(X)$. We define the price as the Tsallis entropy of $\mathbf{Q}(X)$:

$$p^T(\mathbf{Q}) = S_q(\mathbf{Q}(X)) = \sum_{S \in \mathcal{P}_\mathbf{Q}} \frac{p_S}{q-1} \cdot (1 - p_S^{q-1}) \tag{2}$$

▶ **Lemma 31.** *The pricing function p^T defined in Equation (2) is arbitrage-free for $q > 1$.*

Guessing Entropy. The guessing entropy measures the average number of successive guesses required by an optimum strategy until we correctly guess the value of the random variable X (in our case the underlying database D). The guessing entropy was first introduced in [21], and subsequently used in [13] in the context of measuring leakage in side-channel attacks. To compute the guessing entropy of X, suppose that we have ordered the databases in decreasing order of their probabilities, i.e. such that $p(X = D_i) \geq p(X = D_j)$ whenever $i \leq j$. Then, we define the *guessing entropy* as $G(X) = \sum_i i \cdot p_{D_i}$. The price is now defined as the initial entropy minus the expected conditional guessing entropy $G(X \mid \mathbf{Q}(X) = E)$:

$$p^G(\mathbf{Q}) = G(X) - \sum_E p_E \cdot G(X \mid \mathbf{Q}(X) = E) \tag{3}$$

▶ **Lemma 32.** *The pricing function p^G defined in Equation (3) is arbitrage-free.*

Min-Entropy. We apply here the notion of min-entropy, as it was introduced in [26] to quantify information flow. The *min-entropy* of a random variable is $H_\infty(X) = -\log(\max_x P(X = x))$. The conditional min-entropy is defined as $H_\infty(X \mid Y) = -\log(\sum_y P(Y = y) \cdot \max_x P(X = x \mid Y = y))$. Then, we can construct the price of a query as follows:

$$p^M(\mathbf{Q}) = H_\infty(X) - H_\infty(X \mid \mathbf{Q}(X)) = -\log(\max_D p_D) + \log\left(\sum_E \max_{D:\mathbf{Q}(D)=E} p_D\right) \tag{4}$$

▶ **Lemma 33.** *The pricing function p^M defined in Equation (4) has no information arbitrage.*

The min-entropy is not in general bundle arbitrage-free, as we show in the full version of the paper [8]. However, it becomes so when the initial distribution is uniform. Let $n = |\mathcal{I}|$, in which case $p_D = 1/n$ for each database. Then, it is straightforward to see that the resulting function is the logarithm of the number of sets in the partition $\mathcal{P}_\mathbf{Q}$.

$$p^{MU}(\mathbf{Q}) = \log(n) + \log(|\mathcal{P}_\mathbf{Q}|/n) = \log(|\mathcal{P}_\mathbf{Q}|) \tag{5}$$

▶ **Lemma 34.** *The pricing function $p^{MU}(\mathbf{Q})$ defined in Equation (5) is arbitrage-free.*

β-Success Rate. This information measure, first introduced in [4], captures the expected success of guessing the database D with β tries. We will consider here only the case where the probability distribution is uniform, in which case the pricing functions becomes:

$$p^\beta(\mathbf{Q}) = \log\left(\sum_{S \in \mathcal{P}_\mathbf{Q}} \min\{\beta, |S|\}\right) \tag{6}$$

Observe that for $\beta = 1$ we have $p^\beta(\mathbf{Q}) = p^{MU}(\mathbf{Q})$, hence this generalizes uniform min-entropy.

▶ **Lemma 35.** *The pricing function $p^\beta(\mathbf{Q})$ defined in Equation (6) is arbitrage-free.*

We should finally mention that several other entropy measures have been discussed in the broader literature. The Renyi entropy [25] is a generalization of both the Shannon entropy and the min-entropy. However, it is not subadditive, and thus not applicable as an arbitrage-free pricing function. Worst-case entropy measures [13] can also be applied to measure information leakage, but they are also prone to bundle arbitrage.

5 Computing the Pricing Function

So far we have studied how to construct pricing functions for both APS and QPS. In this section, we focus on the complexity of computing a pricing function.

Shannon Entropy	$p^H(\mathbf{Q}) = \frac{1}{n}\sum_{B\in\mathcal{P}_\mathbf{Q}} \lvert B\rvert \log \lvert B\rvert$
Guessing Entropy	$p^G(\mathbf{Q}) = \frac{1}{2n}\left(n^2 - \sum_{B\in\mathcal{P}_\mathbf{Q}} \lvert B\rvert^2\right)$
Min-Entropy	$p^{MU}(\mathbf{Q}) = \log \lvert\mathcal{P}_Q\rvert$
Tsallis Entropy	$p^T(\mathbf{Q}) = \frac{1}{q-1}\left(1 - \sum_{B\in\mathcal{P}_\mathbf{Q}} (\frac{\lvert B\rvert}{n})^{q-1}\right)$
β-Success Rate	$p^\beta(\mathbf{Q}) = \log\left(\sum_{B\in\mathcal{P}_\mathbf{Q}} \min\{\beta, \lvert B\rvert\}\right)$

Table 1 The price of a query bundle $\mathbf{Q}$ according to various entropy measures for the case of uniform probability distributions. We denote $n = |\mathcal{I}|$.

5.1 Support Sets

We first start by discussing an generic approach that can construct efficiently computable arbitrage-free pricing functions for any query language $\mathcal{L}$ that can be computed efficiently. The key idea behind our construction is to define the pricing function on a smaller set $\mathcal{C} \subseteq \mathcal{I}$ of our choice, which we call *support*. The next two lemmas show that this restriction still provides arbitrage-free answer-dependent and instance-independent pricing functions.

▶ **Lemma 36.** *Let $\mathcal{C} \subseteq \mathcal{I}$. If f is a monotone and subadditive set function, the pricing function $p(\mathbf{Q}, E) = f(\overline{\mathcal{S}}_\mathbf{Q}(E) \cap \mathcal{C})$ is arbitrage-free.*

Given a partition $\mathcal{P}$ of the set $\mathcal{I}$, we define the *restriction of $\mathcal{P}$ to $\mathcal{C}$*, denoted $\mathcal{P} \cap \mathcal{C}$, as the set $\{B \cap \mathcal{C} \mid B \in \mathcal{P}, B \cap \mathcal{C} \neq \emptyset\}$.

▶ **Lemma 37.** *Let $\mathcal{C} \subseteq \mathcal{I}$. If f is a monotone and subadditive function on the partition semilattice, the pricing function $p(\mathbf{Q}) = f(\mathcal{P}_\mathbf{Q} \cap \mathcal{C})$ is arbitrage-free.*

The above results provide us with a method to design an efficient arbitrage-free pricing function for a query language $\mathcal{L}$. We start by choosing a support $\mathcal{C} \subseteq \mathcal{I}$. To compute the pricing function, we first compute $\overline{\mathcal{S}}_\mathbf{Q}(E) \cap \mathcal{C}$ for answer-dependent (or $\mathcal{P}_\mathbf{Q} \cap \mathcal{C}$ for instance-independent). The observation is that we can achieve this by evaluating the query bundle $\mathbf{Q}$ only on the databases $D \in \mathcal{C}$. Hence, the running time of computing the price does not depend on $|\mathcal{I}|$, but on $|\mathcal{C}|$ and the complexity of evaluating the query bundle $\mathbf{Q}$.

▶ **Example 38.** Consider any set $\mathcal{C} \subseteq \mathcal{I}$. Then $p(\mathbf{Q}, E) = \log |\{D \in \mathcal{C} \mid \mathbf{Q}(D) \neq E\}|$ is an arbitrage-free pricing function. Similarly, $p(\mathbf{Q}) = \log |\mathcal{P}_\mathbf{Q} \cap \mathcal{C}|$ is also arbitrage-free.

The advantage of using support sets to construct pricing functions is that they provide us with a generic method that is independent of the language $\mathcal{L}$. On the other hand, the size and choice of the support $\mathcal{C}$ is a challenging problem. We can always choose $\mathcal{C}$ to contain a single database. The evaluation of the price will be very efficient, but any query will be assigned only one of two prices, and thus the pricing function will not be very successful in measuring the value of the data. If we instead choose a very large support, this leads to expensive and impractical price computation. We leave as an open research question how to choose a good support $\mathcal{C}$ that is suitable for a practical implementation.

5.2 The Complexity of Entropy-Based Pricing

In a practical setting, the set $\mathcal{I}$ will be given implicitly. For example, $\mathcal{I}$ can be the infinite set of all databases, or the set of all subsets of a given database D_0, $\mathcal{I} = \{D \mid D \subseteq D_0\}$.

One might think that since the problem of determinacy (either query or data-dependent) is hard even for the class of conjunctive queries, computing an arbitrage-free pricing function is always hard. However, as we showed in the previous section about support sets, it is always possible to construct non-trivial pricing schemes that circumvent the computation of determinacy and thus can be computed efficiently. Here we will focus on the computational complexity for the pricing functions we introduced that are based on entropy.

The task necessary to compute an answer-dependent pricing function such as $p(\mathbf{Q}, E) = \log(|\overline{\mathcal{S}}_{\mathbf{Q}}(E)|)$, or any of the instance-independent functions in Table 1 is the following: *given a view extension E and $\mathbf{Q}$, compute $|\mathcal{S}_{\mathbf{Q}}(E)|$, which is the number of databases in $\mathcal{I}$ such that $\mathbf{Q}(D) = E$.* If $\mathcal{I}$ can be succinctly expressed as $\mathcal{I} = \{D \mid D \subseteq D_0\}$, the task relates to the area of probabilistic databases. Indeed, we can view D_0 as a tuple-independent probabilistic database where each tuple has the same probability 1/2. Then, we can write $|\mathcal{S}_{\mathbf{Q}}(E)| = P(\mathbf{Q}(D_0) = E) \cdot |\mathcal{I}|$. Unfortunately, computing the probability $P(\mathbf{Q}(D_0) = E)$ is in general a $^{\#}P$-hard problem (w.r.t. the size of D_0), even for the class of conjunctive queries [7]. However, the task is known to be in polynomial time for certain types of queries. For instance, in Example 4, where Q is a selection query over a single table, the size of the conflict set can be computed exactly in polynomial time. We should note here that the problem of checking whether $\mathcal{S}_{\mathbf{Q}}(E)$ is empty or not is equivalent to the problem of *view consistency*, which is shown to be NP-hard for the class of conjunctive queries [1] when $\mathcal{I}$ ranges over all databases.

Even if $|\mathcal{S}_{\mathbf{Q}}(E)|$ can be computed exactly, the number of blocks in the partition $\mathcal{P}_{\mathbf{Q}}$ may still be exponentially large, which would make computing the Shannon or Guessing entropy intractable. In this case, we can write the information gain as $p(\mathbf{Q}) = -\sum_{D \in \mathcal{I}} \log |[D]_{\mathbf{Q}}|$, and construct an estimator of the price that samples independently m databases from $\mathcal{I}$ and outputs their average: $\tilde{p}(\mathbf{Q}) = \frac{1}{m} \sum_{i=1}^{m} \log |[D_i]_{\mathbf{Q}}|$. In [14, 3], the authors show that such an estimator can achieve an additive δ-approximation of the price with a number of samples that is polynomial in $1/\delta, \log(|\mathcal{I}|)$. We say that a pricing function is *ε-approximately arbitrage-free* if the arbitrage conditions are violated within an additive ε. It is straightforward to see that $\tilde{p}$ results in a (3δ)-approximately arbitrage-free pricing scheme. This implies that we can compute in polynomial time an approximation of the entropy function that is as close to arbitrage-free as we would like to.

6 Related Work

The problem of data pricing has been studied from a wide range of perspectives, including online markets and privacy [10, 22]. [12] examined a variety of issues involved in pricing of information products and presented an economic approach to design of optimal pricing mechanism for online services. [2] introduced the challenge of developing pricing functions in the context of cloud-based environments, where users can pay for queries without buying the entire dataset. This work also outlines various research challenges, such as enabling fine-grained pricing and developing efficient and fair pricing models for cloud-based markets.

The first formal framework for query-based data pricing was introduced by Koutris et al. [15]. The authors define the notion of arbitrage, and provide a framework that takes a set of fixed prices for views over the data identified by seller, and extends these price points to a pricing function over any query. The authors also show that evaluation of the prices can be done efficiently in polynomial time for specific classes of conjunctive queries and a restricted set of views that include only selections. Subsequently, the authors demonstrated how the framework can be implemented into a prototype pricing system called QueryMarket [16, 17].

Further work [18] discusses the pricing and complexity of pricing for the class of aggregate queries. The work by Lin and Kifer [20] proposes several possible forms of arbitrage violations and integrates them into a single framework. The authors allow the queries to be randomized, and propose two potential pricing functions that are arbitrage-free across all forms.

Data pricing is tightly connected to differential privacy [9]. Ghosh and Roth [11] study the buying and selling of data by considering privacy as an entity. Their framework compensates the seller for the loss of privacy due to selling of private data. A similar approach to pricing in the context of privacy is discussed in [19].

We should finally mention the close connection of query pricing to the measurement of information leakage in programs. In [13], the authors apply information-theoretic measures, including various entropy measures, to compute the leakage of information from a side-channel attack that attempts to gain access to secret information. [14] uses similar ideas to quantify the flow of information in programs, and proposes various approximation techniques to efficiently compute them.

7 Conclusion

In this paper, we explore in depth the design space of arbitrage-free pricing functions. We present a characterization of the structure for both answer-dependent and instance-independent pricing functions, and propose several constructions. Our work opens several exciting research questions, including testing which pricing functions behave well in practical settings, and exploring the various tradeoffs when deploying a pricing scheme.

Acknowledgements. We would like to thank Aws Albarghouthi for pointing out the close connection of our work to quantitative information flow and information leakage in side-channel attacks.

References

1 Serge Abiteboul and Oliver M. Duschka. Complexity of answering queries using materialized views. In *PODS*, pages 254–263. ACM Press, 1998.

2 Magdalena Balazinska, Bill Howe, and Dan Suciu. Data markets in the cloud: An opportunity for the database community. *PVLDB*, 4(12), 2011.

3 Tugkan Batu, Sanjoy Dasgupta, Ravi Kumar, and Ronitt Rubinfeld. The complexity of approximating the entropy. *SIAM J. Comput.*, 35(1):132–150, 2005. URL: `http://dx.doi.org/10.1137/S0097539702403645`, `doi:10.1137/S0097539702403645`.

4 S. Boztas. Entropies, guessing and cryptography. Technical Report 6, Department of Mathematics, Royal Melbourne Institute of Technology, 1999.

5 Federal Trade Commission et al. Data brokers: A call for transparency and accountability. *Policy Reports*, Commission and Staff Reports, May 2014.

6 Kenneth Cukier and Viktor Mayer-Schoenberger. Rise of big data: How it's changing the way we think about the world, the. *Foreign Aff.*, 92:28, 2013.

7 Nilesh N. Dalvi, Christopher Ré, and Dan Suciu. Probabilistic databases: diamonds in the dirt. *Commun. ACM*, 52(7):86–94, 2009. URL: `http://doi.acm.org/10.1145/1538788.1538810`, `doi:10.1145/1538788.1538810`.

8 Shaleen Deep and Paraschos Koutris. The design of arbitrage-free data pricing schemes. *arXiv preprint arXiv:1606.09376*, 2016.

9 Cynthia Dwork. A firm foundation for private data analysis. *Commun. ACM*, 54(1):86–95, 2011.

10 Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In Shai Halevi and Tal Rabin, editors, *TCC 2006*, volume 3876, pages 265–284. Springer, 2006. URL: `http://dx.doi.org/10.1007/11681878_14`, `doi:10.1007/11681878_14`.

11 Arpita Ghosh and Aaron Roth. Selling privacy at auction. *Games and Economic Behavior*, 2013.

12 Sanjay Jain and P. K. Kannan. Pricing of information products on online servers: Issues, models, and analysis. *Management Science*, 48(9):1123–1142, 2002.

13 Boris Köpf and David Basin. An information-theoretic model for adaptive side-channel attacks. In *CCS*, pages 286–296. ACM, 2007.

14 Boris Köpf and Andrey Rybalchenko. Approximation and randomization for quantitative information-flow analysis. In *CSF, 2010*, pages 3–14. IEEE, July 2010.

15 Paraschos Koutris, Prasang Upadhyaya, Magdalena Balazinska, Bill Howe, and Dan Suciu. Query-based data pricing. In Michael Benedikt, Markus Krötzsch, and Maurizio Lenzerini, editors, *PODS*, pages 167–178. ACM, 2012.

16 Paraschos Koutris, Prasang Upadhyaya, Magdalena Balazinska, Bill Howe, and Dan Suciu. Querymarket demonstration: Pricing for online data markets. *PVLDB*, 5(12):1962–1965, 2012.

17 Paraschos Koutris, Prasang Upadhyaya, Magdalena Balazinska, Bill Howe, and Dan Suciu. Toward practical query pricing with querymarket. In Kenneth A. Ross, Divesh Srivastava, and Dimitris Papadias, editors, *ACMSIGMOD 2013*, pages 613–624. ACM, 2013. URL: `http://doi.acm.org/10.1145/2463676.2465335`, `doi:10.1145/2463676.2465335`.

18 C. Li and G. Miklau. Pricing aggregate queries in a data marketplace. In *WebDB*, 2012.

19 Chao Li, Daniel Yang Li, Gerome Miklau, and Dan Suciu. A theory of pricing private data. *ACM Trans. Database Syst.*, 39(4):34:1–34:28, 2014. URL: `http://doi.acm.org/10.1145/2691190.2691191`, `doi:10.1145/2691190.2691191`.

20 Bing-Rong Lin and Daniel Kifer. On arbitrage-free pricing for general data queries. *PVLDB*, 7(9):757–768, 2014. URL: `http://www.vldb.org/pvldb/vol7/p757-lin.pdf`.

21 J.L. Massey. Guessing and entropy. In *Information Theory, 1994*, page 204, Jun 1994. doi:10.1109/ISIT.1994.394764.

22 Frank McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In Ugur Çetintemel, Stanley B. Zdonik, Donald Kossmann, and Nesime Tatbul, editors, *ACM SIGMOD 2009*, pages 19–30. ACM, 2009. URL: http://doi.acm.org/10.1145/1559845.1559850, doi:10.1145/1559845.1559850.

23 Alan Nash, Luc Segoufin, and Victor Vianu. Determinacy and rewriting of conjunctive queries using views: A progress report. In *ICDT*, pages 59–73, 2007.

24 Alan Nash, Luc Segoufin, and Victor Vianu. Views and queries: Determinacy and rewriting. *ACM Trans. Database Syst.*, 35(3), 2010.

25 A. Renyi. On measures of information and entropy. In *Berkeley Symposium on Mathematics, Statistics and Probability*, pages 547–561, 1960. URL: http://digitalassets.lib.berkeley.edu/math/ucb/text/math_s4_v1_article-27.pdf.

26 Geoffrey Smith. On the foundations of quantitative information flow. In Luca de Alfaro, editor, *FOSSACS 2009*, LNCS, pages 288–302. Springer, 2009. URL: http://dx.doi.org/10.1007/978-3-642-00596-1_21, doi:10.1007/978-3-642-00596-1_21.

27 Constantino Tsallis. Possible generalization of boltzmann-gibbs statistics. *Journal of Statistical Physics*, 52(1-2):479–487, 1988. URL: http://dx.doi.org/10.1007/BF01016429, doi:10.1007/BF01016429.