Technical Report UW-CSE-12-03-02

Query-Based Data Pricing

Paraschos Koutris, Prasang Upadhyaya, Magdalena Balazinska, Bill Howe, Dan Suciu
University of Washington, Seattle, WA

Abstract

Increasingly, data is being bought and sold online, and Web-based marketplace services have emerged to facilitate selling and buying data. Current pricing mechanisms, however, are very simple, providing a fixed set of views, each with a specific price. In this paper, we propose a framework for pricing data on the Internet that allows the seller to set explicit prices for only a few views, yet allows the buyer to buy *any query*; the price of the query is derived *automatically* from the explicit prices of the views. We call it "query-based pricing". We first identify two important properties that the pricing function must satisfy, called arbitrage-free and discount-free, then prove that there exists a unique such pricing function that extends the seller's explicit prices to all queries. When both the views and the query are Unions of Conjunctive Queries, the complexity of computing the price is high. To address that, we restrict the explicit prices to be defined only on selection views (which is the common practice today). We give an algorithm with a PTIME data complexity for computing the price of any chain query, by reducing the problem to a network flow problem. Furthermore, we completely characterize the class of conjunctive queries without self-joins that have PTIME data complexity (it is slightly larger than chain queries), and prove that all other queries are NP-complete, thus establishing a dichotomy of the complexity of computing the price, when all views are selection queries.

1. Introduction

Whether for market research, targeted product advertisement, or other business decisions, companies commonly purchase data. Increasingly, such data is being bought and sold online. For example, Xignite [32] sells financial data, Gnip [1] provides data from social media, PatientsLikeMe [2] sells anonymized, self-reported, patient statistics to pharmaceutical companies, and AggData [6] aggregates various types of data available on the Web. To support and facilitate this online *data market*, Web-based marketplace

Email addresses: `pkoutris@cs.washington.edu` (Paraschos Koutris), `prasang@cs.washington.edu` (Prasang Upadhyaya), `magda@cs.washington.edu` (Magdalena Balazinska), `billhowe@cs.washington.edu` (Bill Howe), `suciu@cs.washington.edu` (Dan Suciu)

services have recently emerged: The Windows Azure Marketplace [9] offers over 100 data sources from 42 publishers in 16 categories, and Infochimps [19] offers over 15,000 data sets also from multiple vendors.

Current marketplace services do not support complex ad hoc queries, in part because it is not clear how to assign a price to the result. Instead, sellers are asked to define a fixed set of (possibly parameterized) views and assign each a specific price. This simplistic approach not only forces the seller to try and anticipate every view in which a buyer might be interested, but also forces the buyer to browse a large catalog of views with possibly unknown redundancies and relationships and then purchase some superset of the data they actually need. Worse, this pricing model can expose non-obvious arbitrage situations that can allow a cunning buyer to obtain data for less than the advertised price. A better approach, which we explore in this paper, is to allow the seller to assign prices to a manageable number of views and automatically derive the correct price for *any query*.

Consider an example. CustomLists [14] sells the American Business Database for \$399; a customer can also buy the subset of companies that have an e-mail address for \$299 or only information about businesses in Washington State for \$199. A customer interested only in a set of specific counties in various states may not be willing to pay \$399 for data she does not need, and so refuses to buy. In response, the seller might provide a view for each county in every state. However, the relationship between state-based pricing and county-based pricing is difficult for either the seller or the buyer to reason about, and inconsistencies or arbitrage situations may result. For example, if the database does not contain any business information for some fraction of counties in a state, then purchasing the data for the remaining counties could be cheaper, yet could yield the same information content as purchasing the data for the entire state.

Query-based Pricing. To address the above challenge, in this paper we propose a framework for pricing data on the Internet that allows the seller to set explicit prices on only a few views (or sets of views), yet allows the buyer to issue and purchase any query. The price of the query is derived *automatically* from the explicit prices of the views. Thus, buyers have full freedom to choose which query to buy, without requiring the seller to explicitly set prices on an exhaustive catalog of all possible queries. We call this pricing mechanism *query-based pricing*. Our mechanism is based on a recent economic theory of pricing information products based on *versions* [28] (reviewed in Section 5), in the sense that each query corresponds to a version. Since every query (in a given query language) is a version, our framework allows a large number of verions, and, as a consequence, appeals to large variety of buyers with a large variety of needs.

Formally, query-based pricing consists of a *pricing function*, which takes as input a database instance and a query (or set of queries) and returns a non-negative real number representing the price. We argue that a reasonable pricing function should satisfy two axioms.

First, the pricing function should be *arbitrage-free*. Consider the USA business dataset: if p is the price for the entire dataset and $p_1, \dots p_{50}$ the prices for the data in each of the 50 states, then a rational seller would ensure that $p_1 + \dots + p_{50} \geq p$. Otherwise, no buyer would pay for the entire dataset, but would instead buy all 50 states separately. In general, we say that a pricing function is arbitrage-free if whenever a query q is "determined" by the queries $q_1, \dots, q_n$, then their prices satisfy $p \leq p_1 + \dots + p_n$.

Second, the pricing function should be *discount-free*. This axiom concerns the way the pricing function is derived from the explicit views and prices set by the seller. When she specifies and explicit price p_i for some view V_i, the sellers' intent is to sell the view at a discount over the entire data set: the latter is normally sold at a premium price, $p \gg p_i$. The discount-free axiom requires that the pricing function should not introduce any new discounts over those explicitly defined by the seller.

In addition to these two axioms, we argue that the pricing function should also be *monotone* with respect to database updates: when new data items are inserted into the database, the price of a query should not decrease. We show that, in general, the pricing function is not necessarily monotone, but give sufficient conditions under which it is.

Contributions. We present several results on query-based pricing.

Our first result is a simple but fundamental formula for computing an arbitrage-free, discount-free pricing function that agrees with the seller's explicit price points, and for testing whether one exists; if it exists, we call the set of price points *consistent*. To check consistency, it suffices to check that no arbitrage is possible between the explicit price points defined by the seller: there are only finitely many arbitrage combinations, as opposed to the infinitely many arbitrage combinations on all possible queries; hence, consistency is decidable. When the set is consistent, the pricing function is unique, and is given by the *arbitrage-price* formula (Equation 2). This implies an explicit, yet inefficient method for computing the price, which is presented in Section 2.

Second, we turn to the tractability question. We show that even when the seller's explicit price points are restricted to selection queries (which is the common case for data sold online today), computing the price of certain conjunctive queries is NP-hard in the size of the input database. For this reason, we propose a restriction of conjunctive queries, which we call *Generalized Chain Queries*, or GChQ. These are full conjunctive queries whose atoms can be ordered in a sequence such that for any partition into a prefix and a suffix, the two sets of atoms share at most one variable. GChQ includes all path joins, like $R(x, y), S(y, z), T(z, u), P(u, v)$, star joins, like $R(x, y), S(x, z, u), T(x, v), P(x, w)$, and combinations. We prove that, when all explicit price points are selection queries, one can compute the price of every GChQ query in PTIME data complexity. This is the main result of our paper, and provides a practical framework for query-based pricing. The algorithm is based on a non-trivial reduction to the Min-Cut problem in weighted graphs, which is the dual of the Max-Flow problem [13], Subsection 3.1.

Third, we study the complexity of all conjunctive queries without self-joins. We prove that cycle queries (which are *not* generalized chain queries) can also be computed in polynomial time: this is the most difficult result in our paper, and the algorithm is quite different from the algorithm for GChQ. With this result, we can prove a dichotomy of the data complexity of all conjunctive queries without self-joins, in PTIME or NP-complete, Subsection 3.2.

Our pricing framework is based on a notion of query determinacy. Informally, we say that a set of views V *determines* some query Q if we can compute the answer of Q only from the answers of the views without having access to the underlying database. *Information-theoretic* determinacy, denoted $V \twoheadrightarrow Q$, is discussed by Segoufin and Vianu [27] and by Nash, Segoufin, and Vianu [24, 25], and is a notion that is independent of the database instance; their motivation comes from *local-as-view* data integration and *semantic caching*, where an instance independent rewriting is needed. For query-based pricing, however, the database instance cannot be ignored when checking determinacy, since the price normally depends on the state of the database. For example, consider a query Q_1 that asks for the businesses that are located in both Oregon and Washington State and a query Q_2 that asks for the restaurants located in Oregon, Washington and Idaho. In general, we cannot answer Q_2 if we know the answer of Q_1. But suppose we examine the answer for Q_1 and note that it includes no restaurants: then we can safely determine that Q_2 is empty. We define *instance-based determinacy*, $D \vdash V \twoheadrightarrow Q$, to mean that, for all D' if $V(D') = V(D)$, then $Q(D) = Q(D')$. Information-theoretic determinacy is equivalent to instance-based determinacy *for every instance* D. We prove several results on the complexity of checking instance-based determinacy: for unions of conjunctive queries, it is Π_2^p, and the data complexity (when V, Q are fixed and the input is only D) is co-NP complete (Theorem 2.3). When the views are restricted to selection queries (which is a case of special interest in query-based pricing), then for any monotone query Q, instance-based determinacy has polynomial time data complexity, assuming Q itself has PTIME data complexity (Theorem 3.3).

The paper is organized as follows. We introduce the query-based pricing framework and give the fundamental formula for checking consistency and computing the pricing function in Section 2. We turn to the tractability questions in Section 3, where we describe our main result consisting of the polynomial time algorithm for Generalized Chain Queries in Subsection 3.1, and give the dichotomy theorem in Subsection 3.2. We discuss some loose ends in Section 2 and related work in Section 5, then conclude in Section 6.

2. The Query Pricing Framework

2.1. Notations

Fix a relational schema $\mathbf{R} = (R_1, \dots, R_k)$; we denote a database instance with $D = (R_1^D, \dots, R_k^D)$, and the set of all database instances with $\mathit{Inst}_{\mathbf{R}}$ [21]. In this paper we only consider monotone queries, and we denote $\mathcal{L}$ a fixed query language; in particular, CQ, UCQ are the Conjunctive Queries, and Unions of Conjunctive Queries respectively. $Q(D)$ denotes the answer of a query Q on a database D. A *query bundle* is a finite set of queries; we use the term “bundle” rather than “set” to avoid confusion between a set of queries and a set of answers. We denote by $B(\mathcal{L})$ the set of query bundles over $\mathcal{L}$, and write a bundle as $\mathbf{Q} = (Q_1, \dots, Q_m)$. The *output schema* of a query bundle is $\mathbf{R_Q} = (R_{Q_1}, \dots, R_{Q_m})$, and consists of one relation name for each query. Thus, a bundle defines a function $\mathbf{Q} : \mathit{Inst}_{\mathbf{R}} \to \mathit{Inst}_{\mathbf{R_Q}}$.

The *identity bundle*, **ID**, is the bundle that returns the entire dataset, $\mathbf{ID}(D) = (R_1^D, \dots, R_k^D)$. The *empty bundle* is denoted (): it is the empty set of queries, not to be confused with the emptyset query. Given two bundles, $\mathbf{Q}_1$ and $\mathbf{Q}_2$, we denote their union as $\mathbf{Q} = \mathbf{Q}_1, \mathbf{Q}_2$: this is the query bundle consisting of all queries in $\mathbf{Q}_1$ and $\mathbf{Q}_2$, not to be confused with the union $Q_1 \cup Q_2$ of two queries of the same arity.

2.2. The Pricing Function

Definition 2.1 (Pricing Function)**.** *Fix a database instance $D \in \mathit{Inst}_{\mathbf{R}}$. A* static pricing function *is a function $p_D : B(\mathcal{L}) \to \mathbb{R}$.*

A dynamic pricing function *is a partial function $p : \mathit{Inst}_{\mathbf{R}} \to (B(\mathcal{L}) \to \mathbb{R})$, s.t. for each D where p is defined, $p(D)$ is a static pricing function. We write p_D for $p(D)$.*

The intuition is as follows. If the user asks for the bundle $\mathbf{Q}$, then she has to pay the price $p_D(\mathbf{Q})$, where D is the current database instance. The static pricing function is defined only for the current state of the database D. A dynamic pricing function p allows the database to be updated, and associates a different pricing function p_D to each database; notice that it need not be defined for all instances $D \in \mathit{Inst}_{\mathbf{R}}$. We start with static pricing in this section, and call a static pricing function simply a pricing function; we discuss dynamic pricing in Subsection 2.7.

The price is for an entire query bundle, not just for one query. For example, if a user needs to compute queries Q_1, Q_2, and Q_3, then she could issue them separately, and pay $p_D(Q_1) + p_D(Q_2) + p_D(Q_3)$, but she also has the option of issuing them together, as a bundle, and pay $p_D(Q_1, Q_2, Q_3)$. We will show that, in general, the pricing function is subadditive. In particular, the price of the bundle is lower than the sum of the individual prices.

In the query pricing framework, the seller does not specify the pricing function directly, but gives only a finite set of explicit price points, and the system computes the pricing function on all queries; this function must, furthermore, satisfy two axioms, arbitrage-free and discount-free. In the rest of this section we discuss the details of this framework.

2.3. Axiom 1: Arbitrage-Free

The first axiom that a pricing function must satisfy is defined in terms of a notion of determinacy. Intuitively, a bundle $\mathbf{V}$ determines a bundle $\mathbf{Q}$ given a database D, denoted $D \vdash \mathbf{V} \twoheadrightarrow \mathbf{Q}$, if one can answer $\mathbf{Q}$ from the answer of $\mathbf{V}$ by applying a function f such that $\mathbf{Q}(D) = f(\mathbf{V}(D))$. The impact on pricing is that if the user needs to answer the query $\mathbf{Q}$, she also has the option of querying $\mathbf{V}$, and then applying f. The arbitrage-free axiom requires that $p_D(\mathbf{Q}) \leq p_D(\mathbf{V})$, meaning that the user will never have the incentive to compute $\mathbf{Q}$ indirectly by purchasing $\mathbf{V}$. Thus, the notion of arbitrage depends on the notion of determinacy, which we define here:

Definition 2.2 (Instance-based Determinacy)**.** $\mathbf{V}$ *determines* $\mathbf{Q}$ *given the database* D*, denoted* $D \vdash \mathbf{V} \twoheadrightarrow \mathbf{Q}$*, if for any* D'*,* $\mathbf{V}(D) = \mathbf{V}(D')$ *implies* $\mathbf{Q}(D) = \mathbf{Q}(D')$*.*

The connection to answerability is the following. Let $f : \mathit{Inst}_{\mathbf{R}_{\mathbf{V}}} \to \mathit{Inst}_{\mathbf{R}_{\mathbf{Q}}}$ be Q composed with any left inverse of $\mathbf{V}$: that is, for every $E \in \mathit{Inst}_{\mathbf{R}_{\mathbf{V}}}$, if there exists D s.t. $\mathbf{V}(D) = E$, then choose any such D and define $f(E) = \mathbf{Q}(D)$, otherwise, $f(E)$ is undefined. One can check that $D \vdash \mathbf{V} \twoheadrightarrow \mathbf{Q}$ iff $\forall D'.\mathbf{V}(D) = \mathbf{V}(D') \Rightarrow f(\mathbf{V}(D')) = \mathbf{Q}(D')$. Thus, if the user knows $\mathbf{V}(D)$ and $D \vdash \mathbf{V} \twoheadrightarrow \mathbf{Q}$ holds, then she can compute $\mathbf{Q}(D)$ as $f(\mathbf{V}(D))$. We establish the following (in the Appendix, Subsection Appendix B.2):

Theorem 2.3. *The combined complexity of instance based determinacy* $D \vdash \mathbf{V} \twoheadrightarrow \mathbf{Q}$ *when* $\mathbf{V}, \mathbf{Q}$ *are in* $B(UCQ)$ *is in* Π_2^p*; the data complexity (where* $\mathbf{V}, \mathbf{Q}$ *are fixed) is co-NP-complete, and remains co-NP complete even for* $B(CQ)$*.*

Instance based determinacy is different from *information theoretic determinacy*, defined in [24] as follows: $\mathbf{V} \twoheadrightarrow \mathbf{Q}$ if $\forall D : D \vdash \mathbf{V} \twoheadrightarrow \mathbf{Q}$. Information-theoretic determinacy, $\mathbf{V} \twoheadrightarrow \mathbf{Q}$, is undecidable for $B(UCQ)$ and its status is unknown for $B(CQ)$ [24].

Example 2.4. *Let $Q_1(x,y,z) = R(x,y), S(y,z)$, $Q_2(y,z,u) = S(y,z), T(z,u)$ and $Q(x,y,z,u) = R(x,y), S(y,z), T(z,u)$. Then $(Q_1, Q_2) \twoheadrightarrow Q$, because it suffices to define f as the function that joins $Q_1(D)$ and $Q_2(D)$; then $Q(D) = f(Q_1(D), Q_2(D))$ for all D. On the other hand, $Q_1 \not\twoheadrightarrow Q$. However, let D be a database instance s.t. $Q_1(D) = \emptyset$. Then $D \vdash Q_1 \twoheadrightarrow Q$, because we know that $Q(D) = \emptyset$. For example, let f always return the emptyset: then, for any D' s.t. $Q_1(D) = Q_1(D')(= \emptyset)$ we have $Q(D') = f(Q_1(D'))$.*

In this paper we use instance-based determinacy to study pricing. However, other options are possible: for example one may use information-theoretic determinacy, or one may use the closure that we discuss in Subsection 2.7. To keep the framework general, we base our discussion on an abstract notion of determinacy, defined below. Our results in this section apply to any determinacy relation that satisfies this definition, except for complexity results, which are specific to instance-based determinacy. Our results in the next section are specific to instance-based determinacy.

Definition 2.5. *A* determinacy relation *is a ternary relation $D \vdash \mathbf{V} \twoheadrightarrow \mathbf{Q}$ that satisfies the following properties:*

Reflexivity: $D \vdash \mathbf{V}_1, \mathbf{V}_2 \twoheadrightarrow \mathbf{V}_1$.
Transitivity: *if* $D \vdash \mathbf{V}_1 \twoheadrightarrow \mathbf{V}_2$ *and* $D \vdash \mathbf{V}_2 \twoheadrightarrow \mathbf{V}_3$, *then* $D \vdash \mathbf{V}_1 \twoheadrightarrow \mathbf{V}_3$.
Augmentation: *if* $D \vdash \mathbf{V}_1 \twoheadrightarrow \mathbf{V}_2$, *then* $D \vdash \mathbf{V}_1, \mathbf{V}' \twoheadrightarrow \mathbf{V}_2, \mathbf{V}'$.
Boundedness: $D \vdash \mathbf{ID} \twoheadrightarrow \mathbf{V}$

We prove in Appendix Appendix B, that both instance-based and information-theoretic determinacy satisfy this definition. We also have:

Lemma 2.6. *If $\twoheadrightarrow$ is a determinacy relation, then (a) $D \vdash \mathbf{V} \twoheadrightarrow ()$ for every bundle $\mathbf{V}$, and (b) if $D \vdash \mathbf{V} \twoheadrightarrow \mathbf{V}_1$ and $D \vdash \mathbf{V} \twoheadrightarrow \mathbf{V}_2$, then $D \vdash \mathbf{V} \twoheadrightarrow \mathbf{V}_1, \mathbf{V}_2$.*

Proof. The reflexivity axiom $D \vdash \mathbf{V}, () \twoheadrightarrow ()$ proves the first claim, since $\mathbf{V}, () = \mathbf{V}$. For the second, we apply augmentation to $D \vdash \mathbf{V} \twoheadrightarrow \mathbf{V}_1$ and obtain $D \vdash \mathbf{V}, \mathbf{V} \twoheadrightarrow \mathbf{V}, \mathbf{V}_1$; next apply augmentation to $D \vdash \mathbf{V} \twoheadrightarrow \mathbf{V}_2$ and obtain $D \vdash \mathbf{V}, \mathbf{V}_1 \twoheadrightarrow \mathbf{V}_1, \mathbf{V}_2$; transitivity gives us $D \vdash \mathbf{V}, \mathbf{V} \twoheadrightarrow \mathbf{V}_1, \mathbf{V}_2$, which proves the claim because $\mathbf{V}, \mathbf{V} = \mathbf{V}$. □

The Arbitrage-Free Axiom. We can now state the first axiom that a pricing function must satisfy:

Definition 2.7 (Arbitrage-free). *A pricing function p_D is* arbitrage-free *if, whenever $D \vdash \mathbf{Q}_1, \ldots, \mathbf{Q}_k \twoheadrightarrow \mathbf{Q}$, then $p_D(\mathbf{Q}) \leq \sum_i p_D(\mathbf{Q}_i)$.*

Of course, even if $\mathbf{Q}_1, \ldots, \mathbf{Q}_k$ determine $\mathbf{Q}$, it may be non-trivial for the buyer to compute the answer of $\mathbf{Q}$ from the answers of $\mathbf{Q}_1, \ldots, \mathbf{Q}_k$, for two reasons: she first needs to find the function f for which $f(\mathbf{Q}_1(D), \ldots, \mathbf{Q}_k(D)) = \mathbf{Q}(D)$, and, second, it may be computationally expensive to evaluate f. In this paper, however, we do not address the economic cost of the computation, focusing only on the information-theoretic aspect; i.e. we assume that the only cost that matters is that of the data itself. Thus, if an arbitrage condition exists, then the buyer will exploit it, by avoiding to pay $p_D(\mathbf{Q})$ and purchasing $\mathbf{Q}_1, \ldots, \mathbf{Q}_k$ instead, then computing $\mathbf{Q}$ (at no extra cost).

Arbitrage-free pricing functions exists: the trivial function $p_D(\mathbf{Q}) = 0$, for all $\mathbf{Q}$, is arbitrage-free; we will show non-trivial functions below. First, we prove some properties.

Proposition 2.8. *Any arbitrage-free pricing function p_D has the following properties:*

1. *Subadditive:* $p_D(\mathbf{Q}_1, \mathbf{Q}_2) \leq p_D(\mathbf{Q}_1) + p_D(\mathbf{Q}_2)$.
2. *Non-negative:* $p_D(\mathbf{Q}) \geq 0$.
3. *Not asking*[1] *is free:* $p_D() = 0$.
4. *Upper-bounded:* $p_D(\mathbf{Q}) \leq p_D(\mathbf{ID})$.

Proof. We apply arbitrage-freeness to two instances of the reflexivity property. First to $D \vdash \mathbf{Q}_1, \mathbf{Q}_2 \twoheadrightarrow \mathbf{Q}_1, \mathbf{Q}_2$, and derive $p_D(\mathbf{Q}_1, \mathbf{Q}_2) \leq p_D(\mathbf{Q}_1) + p_D(\mathbf{Q}_2)$, which proves item 1. Next to $D \vdash \mathbf{Q}, \mathbf{Q}' \twoheadrightarrow \mathbf{Q}'$, and derive $p_D(\mathbf{Q}') \leq p_D(\mathbf{Q}) + p_D(\mathbf{Q}')$, which implies $p_D(\mathbf{Q}) \geq 0$, proving item 2. For item 3, take $\mathbf{Q} = ()$ and $k = 0$ in Definition 2.7: then $D \vdash \mathbf{Q}_1, \ldots, \mathbf{Q}_k \twoheadrightarrow \mathbf{Q}$ holds by reflexivity ($D \vdash () \twoheadrightarrow ()$) and $p_D(\mathbf{Q}) \leq \sum_i p_D(\mathbf{Q}_i)$ implies $p_D() \leq 0$. Also, arbitrage-freeness applied to the boundedness axiom $D \vdash \mathbf{ID} \twoheadrightarrow \mathbf{Q}$ proves item 4. □

2.4. Explicit Price Points

It is difficult to specify a non-trivial arbitrage-free pricing function, and we do not expect the data owner to define such a function herself. Instead, the data owner specifies a set of explicit price-points, and the system extrapolates them to a pricing function on all query bundles. A *price point* is a pair consisting of a *view* (query bundle) and a *price* (positive real number).

Definition 2.9 (Price points). *A* price point *is a pair $(\mathbf{V}, p)$, where $\mathbf{V} \in B(\mathcal{L})$ and $p \in \mathbb{R}^+$. We denote a finite set of price points $\mathcal{S}$ as $\{(\mathbf{V}_1, p_1), \ldots, (\mathbf{V}_m, p_m)\}$.*

We will assume that $D \vdash (\mathbf{V}_1, \ldots, \mathbf{V}_m) \twoheadrightarrow \mathbf{ID}$; i.e., the seller is always willing to sell the entire dataset, perhaps indirectly through the other views. This is a reasonable assumption: if the seller does not wish to sell certain parts of the data, then we can simply not model those parts by removing relation names from the schema or removing tuples from the instance. To simplify the discussion, in this section we assume that $(\mathbf{ID}, B) \in \mathcal{S}$; i.e., $\mathbf{ID}$ is sold explicitly at some (high) premium price B. We will relax this assumption in Section 3.

Definition 2.10 (Validity). *A pricing function p_D is* valid *w.r.t. a set $\mathcal{S}$ of price-points if:*

1. *p_D is arbitrage-free.*
2. *$\forall (\mathbf{V}_i, p_i) \in \mathcal{S}$, $p_D(\mathbf{V}_i) = p_i$.*

Our goal is to compute a valid pricing function for a set $\mathcal{S}$. In general, such a function may not exist; if it exists, then we call $\mathcal{S}$ consistent.

Definition 2.11 (Consistency). *A set of price points $\mathcal{S}$ is* consistent *if it admits a valid pricing function.*

[1] $p_D()$ means $p_D(())$, the price of the empty bundle.

2.5. Axiom 2: Discount-Free

To see the intuition behind the second axiom, recall that B is the price set by the data owner for the entire dataset. Any arbitrage-free pricing function will be $\leq B$, by Proposition 2.8 (item 4). The explicit price points in $\mathcal{S}$ can be seen as *discounts* offered by the seller relative to the price that would be normally charged if that price point were not included in $\mathcal{S}$. The second axiom requires a pricing function to make no additional implicit discounts.

Definition 2.12 (Discount-free)**.** *A valid pricing function p_D for $\mathcal{S}$ is called* discount-free *if for any other valid pricing function p'_D we have: $\forall \mathbf{Q}$, $p'_D(\mathbf{Q}) \leq p_D(\mathbf{Q})$.*

A discount-free pricing function is unique, because if both p_D and p'_D are discount free, then we have both $p_D \leq p'_D$ and $p'_D \leq p_D$, hence $p_D = p'_D$. We will show that, if $\mathcal{S}$ is consistent, then it admits a discount-free pricing function.

2.6. The Fundamental Query Pricing Formula

The fundamental formula gives an explicit means for checking consistency and for computing the discount-free price. We start by showing that a valid set $\mathcal{S}$ has a discount-free price function.

Existence. If $\mathcal{S}$ is consistent, then the set of valid pricing functions is non-empty, and may be infinite; if this set has a maximum price function, then the maximum function is discount-free. We prove that this is indeed the case.

Lemma 2.13. *Suppose $\mathcal{S}$ contains the identity bundle* **ID***, and let P be a non-empty set of pricing functions valid for $\mathcal{S}$ (P may be infinite). Then*[2] $p_D(\mathbf{Q}) = \sup_{r_D \in P}\{r_D(\mathbf{Q})\}$ *is also a valid pricing function for $\mathcal{S}$.*

Proof. Clearly, p_D satisfies the second condition of validity, $p_D(\mathbf{V}_i) = \sup_{r_D \in P}\{r_D(\mathbf{V}_i)\} = \sup_{r_D \in P}\{p_i\} = p_i$, for every price point $(\mathbf{V}_i, p_i) \in \mathcal{S}$.

Next, we prove that the function p_D is arbitrage-free. Let $D \vdash \mathbf{Q}_1, \ldots, \mathbf{Q}_k \twoheadrightarrow \mathbf{Q}$; we will show that $p_D(\mathbf{Q}) \leq \sum_{i=1}^{k} p_D(\mathbf{Q}_i)$. We have $p_D(\mathbf{Q}) = \sup_{r_D \in P}\{r_D(\mathbf{Q})\} \leq \sup_{r_D \in P}\{\sum_{i=1}^{k} r_D(\mathbf{Q}_i)\}$. Since for any bounded real functions f_i we have that $\sup_{x \in P}\{\sum_{i=1}^{k} f_i(x)\} \leq \sum_{i=1}^{k} \sup_{x \in P}\{f_i(x)\}$, it follows that $\sup_{r_D \in P}\{\sum_{i=1}^{k} r_D(\mathbf{Q}_i)\} \leq \sum_{i=1}^{k} \sup_{r_D \in P}\{r_D(\mathbf{Q}_i)\}$, and this implies our claim. $\square$

Corollary 2.14. *Suppose $\mathcal{S}$ is consistent and contains the identity bundle* **ID***. Then there exists a discount-free pricing function for $\mathcal{S}$.*

We will show below that the requirement that $\mathcal{S}$ contains **ID** is necessary: otherwise $\mathcal{S}$ may be consistent yet not admit any discount-free pricing function.

The Arbitrage-Price. The fundamental formula associates to any $\mathcal{S}$ (not necessarily consistent) a pricing function, called *arbitrage-price*. To introduce the formula, we need some notations. If $\mathbf{Q}_i$ for $i = 1, 2 \ldots, k$ are bundles, then denote their union as $\bigodot_i \mathbf{Q}_i = \mathbf{Q}_1, \ldots, \mathbf{Q}_k$. If $\mathcal{C} \subseteq \mathcal{S}$ is a set of price points, then we denote its total price as $p(\mathcal{C}) = \sum_{(\mathbf{V}_i, p_i) \in \mathcal{C}} p_i$.

[2]Note that $\sup_{r_D \in P}\{r_D(\mathbf{Q})\}$ is well defined (i.e. $\neq \infty$) because for each $r_D \in P$, $p_D(\mathbf{Q}) \leq B$, by Proposition 2.8.

Fix a price points set $\mathcal{S}$ and an instance D. The *support* of a query bundle $\mathbf{Q}$ is:

$$supp_D^{\mathcal{S}}(\mathbf{Q}) = \{\mathcal{C} \subseteq \mathcal{S} \mid D \vdash \bigodot_{(\mathbf{V},p)\in\mathcal{C}} \mathbf{V} \twoheadrightarrow \mathbf{Q}\} \tag{1}$$

Definition 2.15 (Arbitrage-price)**.** *The* arbitrage-price *of a query bundle* $\mathbf{Q}$ *is:*

$$p_D^{\mathcal{S}}(\mathbf{Q}) = \min_{\mathcal{C}\in supp_D^{\mathcal{S}}(\mathbf{Q})} p(\mathcal{C}) \tag{2}$$

The arbitrage price is our fundamental formula. It represents the price that a savvy buyer would pay for the query Q: find the cheapest support $\mathcal{C}$, meaning the cheapest set of views that determine the query $\mathbf{Q}$. We prove:

Lemma 2.16. *(a) For all $(\mathbf{V}_i, p_i) \in \mathcal{S}$, $p_D^{\mathcal{S}}(\mathbf{V}_i) \leq p_i$. In other words, the arbitrage-price is never larger than the explicit price. (b) The arbitrage-price $p_D^{\mathcal{S}}$ is arbitrage-free.*

Proof. The first claim follows from the fact that $\{(\mathbf{V}_i, p_i)\} \in supp_D^{\mathcal{S}}(\mathbf{V}_i)$, because of the reflexivity axiom $D \vdash \mathbf{V}_i \twoheadrightarrow \mathbf{V}_i$. For the second claim, consider $D \vdash \mathbf{Q}_1, \ldots, \mathbf{Q}_k \twoheadrightarrow \mathbf{Q}$; we will prove that $p_D^{\mathcal{S}}(\mathbf{Q}) \leq \sum_i p_D^{\mathcal{S}}(\mathbf{Q}_i)$. For $i = 1, \ldots, k$, let $\mathcal{C}_i^m = \arg\min_{\mathcal{C}\in supp_D^{\mathcal{S}}(\mathbf{Q}_i)} p(\mathcal{C})$. By definition, $D \vdash \bigodot_{(\mathbf{V}_j,p_j)\in\mathcal{C}_i^m} \mathbf{V}_j \twoheadrightarrow \mathbf{Q}_i$ and $p_D^{\mathcal{S}}(\mathbf{Q}_i) = p(\mathcal{C}_i^m)$. Let $\mathcal{C} = \bigcup_i \mathcal{C}_i^m \subseteq \mathcal{S}$. Let $\mathbf{V}^m = \bigodot_{(\mathbf{V}_j,p_j)\in\mathcal{C}} \mathbf{V}_j$. Since $\mathcal{C}_i^m \in supp_D^{\mathcal{S}}(\mathbf{Q}_i)$, it follows that $\mathcal{C} \in supp_D^{\mathcal{S}}(\mathbf{Q}_i)$ because the set $supp_D^{\mathcal{S}}(\mathbf{Q}_i)$ is upwards closed[3]. It follows that $D \vdash \mathbf{V}^m \twoheadrightarrow \mathbf{Q}_i$, for every $i = 1, k$. By inductively applying Lemma 2.6 (b), we derive $D \vdash \mathbf{V}^m \twoheadrightarrow \mathbf{Q}_1, \ldots, \mathbf{Q}_k$ and, by transitivity, we further derive $D \vdash \mathbf{V}^m \twoheadrightarrow \mathbf{Q}$. This implies $\mathcal{C} \in supp_D^{\mathcal{S}}(\mathbf{Q})$, and therefore:

$$p_D^{\mathcal{S}}(\mathbf{Q}) \leq p(\mathcal{C}) = \sum_{(V_j,p_j)\in\mathcal{C}} p_j \leq \sum_i \sum_{(\mathbf{V}_j,p_j)\in\mathcal{C}_i^m} p_j = \sum_i p_D^{\mathcal{S}}(\mathbf{Q}_i)$$

The second inequality holds because the p_i's are non-negative (Proposition 2.8). This proves that $p_D^{\mathcal{S}}$ is arbitrage-free. □

The arbitrage-price is a fundamental formula because it allows us to check consistency, and, in that case, it gives the discount-free price.

Theorem 2.17. *Consider a set of price points $\mathcal{S}$. Let $p_D^{\mathcal{S}}$ denote the arbitrage-price function, Equation 2. Then:*

1. *$\mathcal{S}$ is consistent iff $\forall(\mathbf{V}_i, p_i) \in \mathcal{S}$, $p_i \leq p_D^{\mathcal{S}}(\mathbf{V}_i)$.*
2. *If $\mathcal{S}$ is consistent, then $p_D^{\mathcal{S}}$ is the discount-free pricing function for $\mathcal{S}$.*

Proof. We claim that, for any pricing function p_D valid for $\mathcal{S}$ and every query bundle $\mathbf{Q}$, we have that $p_D(\mathbf{Q}) \leq p_D^{\mathcal{S}}(\mathbf{Q})$. The claim proves the theorem. Indeed, the "if" direction of item 1 follows from two facts. First, $p_D^{\mathcal{S}}$ is arbitrage-free by Lemma 2.16(b). Second, if $p_i \leq p_D^{\mathcal{S}}(\mathbf{V}_i)$ holds for all price points $(\mathbf{V}_i, p_i) \in \mathcal{S}$, then by Lemma 2.16(a) $p_D^{\mathcal{S}}(\mathbf{V}_i) = p_i$. Hence, $p_D^{\mathcal{S}}$ is valid, proving that $\mathcal{S}$ is consistent. The "only if" direction follows from the

[3]For any query bundle $\mathbf{Q}$, if $\mathcal{C}_1 \in supp_D^{\mathcal{S}}(\mathbf{Q})$ and $\mathcal{C}_1 \subseteq \mathcal{C}_2$ then $\mathcal{C}_2 \in supp_D^{\mathcal{S}}(\mathbf{Q})$, by the reflexivity axiom.

claim: if p_D is any valid pricing function for $\mathcal{S}$ then $p_i = p_D(\mathbf{V}_i) \le p^{\mathcal{S}}_D(\mathbf{V}_i)$. The claim also implies item 2 immediately.

To prove the claim, let p_D be a valid pricing function (thus $p_D(\mathbf{V}_i) = p_i$ for all $(\mathbf{V}_i, p_i) \in \mathcal{S}$), and let $\mathbf{Q}$ be a bundle. Let $\mathcal{C} \in \mathit{supp}^{\mathcal{S}}_D(\mathbf{Q})$, and $\mathbf{V} = \bigodot_{(\mathbf{V}_i,p_i)\in\mathcal{C}} \mathbf{V}_i$. By definition we have $D \vdash \mathbf{V} \twoheadrightarrow \mathbf{Q}$. Since p_D is arbitrage-free, we have:

$$p_D(\mathbf{Q}) \le \sum_{(\mathbf{V}_i,p_i)\in\mathcal{C}} p_D(\mathbf{V}_i) = \sum_{(\mathbf{V}_i,p_i)\in\mathcal{C}} p_i = p(\mathcal{C})$$

It follows:

$$p_D(\mathbf{Q}) \le \min_{\mathcal{C}\in \mathit{supp}^{\mathcal{S}}_D(\mathbf{Q})} p(\mathcal{C}) = p^{\mathcal{S}}_D(\mathbf{Q})$$

□

The theorem says that, in order to check consistency it suffices to rule out arbitrage situations among the views in $\mathcal{S}$. There are infinitely many possible arbitrage situations in Definition 2.7: the theorem reduces this to a finite set.

Next, we examine the complexity of checking consistency and computing the price. For this discussion, we will assume that $\twoheadrightarrow$ is the instance-based determinacy given by Definition 2.2. Denote by $\textsc{Price}(\mathcal{S}, \mathbf{Q})$ the decision version of the price computation problem: "given a database D and k, is the price $p^{\mathcal{S}}_D(\mathbf{Q})$ less than or equal to k"? Let us also denote by $\textsc{Price}(\mathbf{Q})$ the decision version of the same problem, but where the set of price points $\mathcal{S}$ is now part of the input.

Corollary 2.18. *Suppose $\mathcal{S}, \mathbf{Q}$ consist of UCQs. Then, (a) the complexity of* $\textsc{Price}(\mathbf{Q})$ *is in Σ^p_2 and (b) the complexity of* $\textsc{Price}(\mathcal{S}, \mathbf{Q})$ *is coNP-complete.*

Proof. For (a), to check whether $p^{\mathcal{S}}_D(\mathbf{Q}) \le k$, guess a subset of price points $(\mathbf{V}_1, p_1), \ldots, (\mathbf{V}_m, p_m)$ in $\mathcal{S}$, then check that both $D \vdash \mathbf{V}_1, \ldots, \mathbf{V}_m \twoheadrightarrow \mathbf{Q}$ (this is in coNP by Theorem 2.3) and that $\sum_i p_i \le k$. For (b), instead of guessing, we can iterate over all subset of price points, since there is only a fixed number of them. □

Thus, computing the price is expensive. This expense is unacceptable in practice, since prices are computed as frequently as queries, perhaps even more frequently (for example users may just inquire about the price, then decide not to buy). We have an extensive discussion of tractability in Section 3, and will describe an important restriction under which pricing is tractable. For now, we restrict our discussion of the complexity to showing that pricing is at least as complex as computing the determinacy relation.

Let $\textsc{Price-Consistency}(\mathcal{S})$ be the problem of deciding whether a set $\mathcal{S}$ is consistent for a database D, and $\textsc{Determinacy}(\mathbf{V}, Q)$ the problem of checking determinacy $D \vdash \mathbf{V} \twoheadrightarrow Q$. The proof of Corollary 2.18 shows that the former problem is no more than exponentially worse than the latter. We prove here a weak converse:

Proposition 2.19. *There is a polynomial time reduction from* $\textsc{Determinacy}(\mathbf{V}, Q)$ *to*[4] $\textsc{Price-Consistency}(\mathcal{S})$.

[4] $\mathcal{S}$ has one price point for each $V \in \mathbf{V}$ and one for Q; the database instance D is part of the input in both cases.

Proof. Assume that we want to decide whether $D \vdash \mathbf{V} \twoheadrightarrow Q$, where $\mathbf{V} = \{V_1, \dots, V_k\}$. We can assume w.l.o.g. that none of the V_i are constant, since in this case we could just remove them from $\mathbf{V}$. We will reduce this to an instance of the Price-Consistency problem. Indeed, consider the following set of price points:

$$\mathcal{S} = \{(V_1, p), \dots, (V_k, p), (Q, kp + \epsilon), (\mathbf{ID}, kp + 2\epsilon)\} \quad \epsilon > 0$$

We will prove that $D \vdash \mathbf{V} \twoheadrightarrow Q$ if and only if $\mathcal{S}$ is not consistent for D. For the one direction, suppose that $D \vdash \mathbf{V} \twoheadrightarrow Q$. Then, for any valid pricing function p_D we must have that $k \cdot p + \epsilon = p_D(Q) \leq \sum_i p_D(V_i) = k \cdot p$, a contradiction. Hence $\mathcal{S}$ admits no valid pricing function and is not consistent.

For the other direction, assume that $\mathcal{S}$ is not consistent. By applying Theorem 2.17, we have that for the arbitrage price $p_D^{\mathcal{S}}$ either there exists some i such that $p_D^{\mathcal{S}}(V_i) < p$ or $p_D^{\mathcal{S}}(Q) < kp + \epsilon$. The first case is not possible, since every set which may determine V_i, including V_i, is priced at least p (note that V_i is not constant). Hence, it must be that $p_D^{\mathcal{S}}(Q) < kp + \epsilon$. It follows that there exists a choice of price points that determine Q and are priced less than $kp + \epsilon$; however, this can only be a subset $\mathbf{V}' \subseteq \mathbf{V}$. Thus, $D \vdash \mathbf{V}' \twoheadrightarrow Q$ and by reflexivity $D \vdash \mathbf{V} \twoheadrightarrow Q$. □

We end this section with a brief discussion of the case when $\mathbf{ID}$ is not determined by $\mathcal{S} = \{(\mathbf{V}_1, p_1), \dots, (\mathbf{V}_k, p_k)\}$, that is, $D \vdash (\mathbf{V}_1, \dots, \mathbf{V}_k) \not\twoheadrightarrow \mathbf{ID}$; the seller does not sell the entire dataset. In that case $\mathcal{S}$ has no discount-free pricing function. Indeed, consider any B such that $B \geq \sum_i p_i$, and denote $\mathcal{S} + B = \mathcal{S} \cup \{(\mathbf{ID}, B)\}$. One can check that, if $\mathcal{S}$ is consistent, then so is $\mathcal{S} + B$, and that $p_D^{\mathcal{S}+B}$ is a valid pricing function for $\mathcal{S}$. If p_D were any discount-free pricing function for $\mathcal{S}$, we fix some database instance D and choose $B > p_D(\mathbf{ID})$. Then $p_D^{\mathcal{S}+B}(\mathbf{ID}) = B > p_D(\mathbf{ID})$, contradicting the fact that p is discount-free. In the rest of the paper, we will always assume that $\mathbf{ID}$ is included in the set of price points.

2.7. Dynamic Pricing

So far we assumed that the database instance was static. We now consider the pricing function in a dynamic setting, i.e. when the database D is updated; we consider only insertions. Note that the set of price points $\mathcal{S}$ remains unchanged, even when the database gets updated. For example, the seller has decided to sell the entire dataset for the price B, $(\mathbf{ID}, B) \in \mathcal{S}$, and this price remains unchanged even when new items are inserted in the database. This is the most common case encountered today: explicit prices remain fixed over long periods of time, even when the underlying data set is updated.

When the database is updated, we can simply recompute the pricing function on the new data instance. However, we face two issues. The first is that the price points $\mathcal{S}$ may become inconsistent: $\mathcal{S}$ was consistent at D_1, but after inserting some items, $\mathcal{S}$ becomes inconsistent at D_2. This must be avoided in practice. Second, as more data items are added, the seller does not want any price to drop.

Example 2.20. *Let $V(x, y) = R(x), S(x, y)$ and $Q() = R(x)$ (a Boolean query checking whether R is non-empty). Let $D_1 = \emptyset$, $D_2 = \{R(a), S(a, b)\}$. Then $D_1 \vdash V \not\twoheadrightarrow Q$ and $D_2 \vdash V \twoheadrightarrow Q$. The second claim is obvious: since $V(D_2) = \{(a, b)\}$ we know for certain that $R^{D_2} \neq \emptyset$. To see the first claim consider $D_1' = \{R(a)\}$. Then $V(D_1) = V(D_1') = \emptyset$ but $Q(D_1) =$ false $\neq Q(D_1') =$ true, proving that $D_1 \vdash V \not\twoheadrightarrow Q$.*

This example implies two undesired consequences. First, let $\mathcal{S}_1 = \{(V, \$1), (Q, \$10), (\mathbf{ID}, \$100)\}$: the owner sells the entire dataset for \$100, the query Q for \$10, and the view V for \$1. $\mathcal{S}_1$ is consistent when the database instance is D_1, but when tuples are inserted and the database instance becomes D_2, then $\mathcal{S}_1$ is no longer consistent (because a buyer can avoid paying \$10 for Q by asking V instead, for just \$1). Alternatively, consider the set of price points $\mathcal{S}_2 = \{(V, \$1), (\mathbf{ID}, \$100)\}$. The reader may check that $\mathcal{S}_2$ is consistent for any database instance D. However, the price of Q decreases when the database is updated: $p^{\mathcal{S}_2}_{D_1}(Q) = \$100$, while $p^{\mathcal{S}_2}_{D_2}(Q) = \$1$.

We describe next two ways to fix both issues.

Definition 2.21. *Fix the bundles $\mathbf{V}, \mathbf{Q}$. We say that a determinacy relation $\twoheadrightarrow$ is* monotone *for $\mathbf{V}, \mathbf{Q}$ if, whenever $D_1 \subseteq D_2$ and $D_2 \vdash \mathbf{V} \twoheadrightarrow \mathbf{Q}$, then $D_1 \vdash \mathbf{V} \twoheadrightarrow \mathbf{Q}$.*

Information-theoretic determinacy is vacuously monotone, since it does not depend on the instance. But, as we saw in Example 2.20, instance-based determinacy is not monotone in general. We prove as follows:

Proposition 2.22. *If $\mathbf{V}$ is a bundle consisting only of selection queries and $\mathbf{Q}$ is a bundle of full conjunctive queries, then instance-based determinacy is monotone for $\mathbf{V}, \mathbf{Q}$.*

Proof. It suffices to prove the proposition for a single query Q. Let $D_2 \vdash \mathbf{V} \twoheadrightarrow Q$. We want to show that, if $D_1 \subseteq D_2$, then $D_1 \vdash \mathbf{V} \twoheadrightarrow Q$. In order to show this, it suffices to prove that for any tuple $t \in D$, $D - t \vdash \mathbf{V} \twoheadrightarrow Q$; then, the proposition follows by induction.

First, notice that $Q(D - t) \subseteq Q(D)$, since Q is monotone. Let $t' \in Q(D - t)$. Then, $t' \in Q(D)$; consider the projections of t' on the different atoms in Q: $t_{R_1}, \ldots, t_{R_k}$. Notice that for any $i = 1, k$, $t_{R_i} \neq t$, since otherwise t' would not belong in $Q(D - t)$. Now, all these tuples must be covered by views $\mathbf{V}$ by Lemma Appendix E.1. Thus, (a) for $t' \in Q(D - t)$, all its projections are covered by $\mathbf{V}$.

Next, let $t' \notin Q(D-t)$. If also $t' \notin Q(D)$, then by Lemma Appendix E.1, there exists some projection $t_{R_i} \notin D$ (hence $t_{R_i} \neq t$) of t' such that t_{R_i} is covered by some view in $\mathbf{V}$. Otherwise, $t' \in Q(D)$ and thus t contributes to t' in D. By Lemma Appendix E.1, it must be covered by some selection in $\mathbf{V}$. In either case, (b) for $t' \notin Q(D)$, there exists a projection t_R on some atom R s.t. $t_R \notin D - t$ and t_R is covered by $\mathbf{V}$.

Combining (a) and (b) and invoking Lemma Appendix E.1, we obtain the desired proposition. □

A conjunctive query is full if it has no projections; in particular, a selection query is full. As the next example shows, the above proposition fails for conjunctive queries with projections.

Example 2.23. *Consider $Q(y) = R(x, y)$, where $Col_x = Col_y = \{0, 1\}$. Let a database D such that $R^D = \{(0, 0), (0, 1)\}$. It is easy to see that $D \vdash \sigma_{R.x=0} \twoheadrightarrow Q$. Now, let us consider $D_1 = \{(0, 0)\}$ and $D_2 = \{(0, 0), (1, 1)\}$. Then, $\sigma_{R.x=0}(D_1) = \sigma_{R.x=0}(D_2) = \{(0, 0)\}$ and also $D_1 \subseteq D$. However, $Q(D_1) = \{0\}$ and $Q(D_2) = \{0, 1\}$; hence, $D_1 \vdash \sigma_{R.x=0} \not\twoheadrightarrow Q$.*

The desired property for a dynamic pricing function is that it be monotone: when data is added to the database, the price should never decrease:

Definition 2.24 (Monotonicity). *Let p be a totally defined, dynamic pricing function. We say that p is* monotone *on $\mathbf{Q}$ if, for any $D_1 \subseteq D_2$, $p_{D_1}(\mathbf{Q}) \leq p_{D_2}(\mathbf{Q})$.*

Fix a set of price points $\mathcal{S}$. The arbitrage-price given by Equation 2 is a totally defined function $p^{\mathcal{S}}$, since $p^{\mathcal{S}}_D$ is well defined for every database instance D. We prove:

Proposition 2.25. *Fix $\mathcal{S}$ and $\mathbf{Q}$, and suppose $\twoheadrightarrow$ is monotone for every subset $\mathbf{V}_1, \ldots, \mathbf{V}_m$ of $\mathcal{S}$, and $\mathbf{Q}$. Then, the dynamic arbitrage-price $p^{\mathcal{S}}$ is monotone on $\mathbf{Q}$.*

Proof. By Equation 1: $supp^{\mathcal{S}}_{D_1}(\mathbf{Q}) \supseteq supp^{\mathcal{S}}_{D_2}(\mathbf{Q})$. By Equation 2: $p^{\mathcal{S}}_{D_1}(\mathbf{Q}) \leq p^{\mathcal{S}}_{D_2}(\mathbf{Q})$. $\square$

Proposition 2.26. *If $p^{\mathcal{S}}$ is monotone on every $\mathbf{V}_i$, $\mathcal{S}$ is consistent on D_1, and $D_1 \subseteq D_2$, then $\mathcal{S}$ is consistent on D_2.*

Proof. To check consistency on D_2 it suffices to check that $p_i \leq p^{\mathcal{S}}_{D_2}(\mathbf{V}_i)$, for all $i = 1, \ldots, m$. We have $p_i \leq p^{\mathcal{S}}_{D_1}(\mathbf{V}_i)$ since $\mathcal{S}$ is consistent on D_1, and $p^{\mathcal{S}}_{D_1}(\mathbf{V}_i) \leq p^{\mathcal{S}}_{D_2}(\mathbf{V}_i)$ because $p^{\mathcal{S}}$ is monotone. $\square$

The goal in the dynamic setting is to ensure that $p^{\mathcal{S}}$ is monotone on every query (Definition 2.24). There are two ways to achieve this. One is to restrict all views to selection queries and all queries to full conjunctive queries: we will pursue this in Section 3. However, if one needs more general views and queries, then we propose a second alternative: to consider a different determinacy relation along with monotone views. Let $\twoheadrightarrow$ be any determinacy relation (Definition 2.5). Its *restriction* $D \vdash \mathbf{V} \twoheadrightarrow^* \mathbf{Q}$ is: $\forall D_0, \mathbf{V}(D_0) \subseteq \mathbf{V}(D)$, $D_0 \vdash \mathbf{V} \twoheadrightarrow \mathbf{Q}$. We prove in Subsection Appendix B.3:

Proposition 2.27. *(a) $\twoheadrightarrow^*$ is a determinacy relation (Definition 2.5), (b) $\twoheadrightarrow^*$ is monotone (Definition 2.21) for any monotone $\mathbf{V}$ and any $\mathbf{Q}$, (c) if $p^{\mathcal{S}}_D$ and $q^{\mathcal{S}}_D$ are the arbitrage-prices for $\twoheadrightarrow$ and $\twoheadrightarrow^*$, respectively, then $p^{\mathcal{S}}_D(\mathbf{Q}) \leq q^{\mathcal{S}}_D(\mathbf{Q})$ for all $\mathbf{Q}$, and (d) if $\twoheadrightarrow$ is the instance-based determinacy, then the data complexity of $\twoheadrightarrow^*$ is in coNP.*

Thus, by replacing instance-based determinacy $\twoheadrightarrow$ with its restriction $\twoheadrightarrow^*$, we obtain a monotone pricing function $q^{\mathcal{S}}$. In particular, if $\mathcal{S}$ is consistent in a database D, then it will remain consistent after insertions. To illustrate, recall that in Example 2.20 $\mathcal{S}_1$ became inconsistent when D_1 was updated to D_2: this is because $D_1 \vdash V \not\twoheadrightarrow Q$ and $D_2 \vdash V \twoheadrightarrow Q$. Now we have both $D_1 \vdash V \not\twoheadrightarrow^* Q$ and $D_2 \vdash V \not\twoheadrightarrow^* Q$, hence $\mathcal{S}_1$ is consistent in both states of the database.

3. Tractable Query-Based Pricing

The combined complexity for computing the price when the views and queries are UCQs is high: it is coNP-hard and in Σ^P_2. This is unacceptable in practice. In this section, we restrict both the views on which the seller can set explicit prices and the queries that the buyer can ask, and present a polynomial time algorithm for computing the price. This is the main result in the paper, since it represents a quite practical framework for query-based pricing. For the case of conjunctive queries without self-joins, we prove a dichotomy of their complexity into polynomial time and NP-complete, which is our most technically difficult result.

The Views. We restrict the views to *selection queries.* We denote a selection query by $\sigma_{R.X=a}$, where R is a relation name, X an attribute, and a a constant. For example, given a ternary relation $R(X, Y, Z)$, the selection query $\sigma_{R.X=a}$ is $Q(x, y, z) = R(x, y, z) \wedge x = a$. Throughout this section, the seller can set explicit prices only on selection views. We argue that this restriction is quite reasonable in practice. Many concrete instances of online data pricing that we have encountered set prices only on selection queries[5]. For example, CustomLists [14] sells the set of all businesses in any given state for \$199, thus it sells 50 selection views. Infochimps [19] sells the following selection queries, in the form of API calls. The Domains API: *given IP address, retrieve the domain, company name and NAICS Code.* The MLB Baseball API: *given an MLB team name, retrieve the wins, losses, current team colors, seasons played, final regular season standings, home stadium, and team_ids.* The Team API: *given the team_ids, get the team statistics, records, and game_ids.* And, the Game API: *given game_id, get the attendance, box scores, and statistics.* Thus, we argue, restricting the explicit price points to selection queries is quite reasonable for practical purposes.

An important assumption made by sellers today is that the set of values on which to select is known. For example, the set of valid MLB team names is known to the buyers, or can be obtained for free from somewhere else. In general, for each attribute $R.X$ we assume a finite set $Col_{R.X} = \{a_1, \ldots, a_n\}$, called the *column.* This set is known both to the seller and the buyer. Furthermore, the database D satisfies the inclusion constraint $R^D.X \subseteq Col_{R.X}$. The input to the pricing algorithm consists of both the database instance D, and all the columns $Col_{R.X}$: thus, the latter are part of the input in data complexity. A column should not be confused with a domain: while a domain may be infinite, a column has finitely many values. It should not be confused with the active domain either, since the database need not have all values in a column. We also assume that columns always remain fixed when the database is updated.

We call the set of all selections on column $R.X$, $\Sigma_{R.X} = \{\sigma_{R.X=a} \mid a \in Col_{R.X}\}$, the *full cover* of $R.X$. Note that $D \vdash \Sigma_{R.X} \twoheadrightarrow R$. We denote Σ the set of all selections on all columns. Given $\mathbf{V} \subseteq \Sigma$, we say that it *fully covers* $R.X$ if $\Sigma_{R.X} \subseteq \mathbf{V}$. Thus, the explicit price points $\mathcal{S} = \{(V_1, p_1), (V_2, p_2), \ldots\}$ are such that each $V_i \in \Sigma$. We denote $p : \Sigma \to \mathbb{R}^+$ the partial function defined as: $p(V_i) = p_i$ if $(V_i, p_i) \in \mathcal{S}$.

Recall that $\textsc{Price}(\mathcal{S}, \mathbf{Q})$ denotes the data complexity of the pricing problem in Section 2. Since now $\mathcal{S}$ can be as large as Σ, we treat it as part of the input. Thus, we denote the pricing problem as $\textsc{Price}(\mathbf{Q})$, where the input consists of the database instance D, all columns $Col_{R.X}$, and the function p.

We start with a basic lemma.

Lemma 3.1. *Let* $\mathbf{V} \subseteq \Sigma$*. Then* $D \vdash \mathbf{V} \twoheadrightarrow \sigma_{R.X=a}$ *iff (a) it is trivial (i.e.* $\sigma_{R.X=a} \in \mathbf{V}$*), or (b)* $\mathbf{V}$ *fully covers some attribute* Y *of* R*.*

Proof. Assume that $D \vdash \mathbf{V} \twoheadrightarrow \sigma_{R.X=a}$ and neither (a) or (b) holds. Let $R(X_1, \ldots, X_k)$. Then, for every attribute X_i of R, there exists a selection $\sigma_{R.X_i=a_i} \notin \mathbf{V}$. Let $X = X_1$ and consider the database $D' = D \cup \{R(a, a_2, \ldots, a_k)\}$. It is easy to see that $\mathbf{V}(D) = \mathbf{V}(D')$, since the tuple $t_R = R(a, a_2, \ldots, a_k)$ does not appear in any of the views. However, $t_R \in \sigma_{R.X=a}(D')$ and $t_R \notin \sigma_{R.X=a}(D)$, a contradiction. $\square$

[5]The only exception are sites that sell data by the number of tuples; for example, Azure allows the seller to set a price on a "transaction", which means any 100 tuples.

The lemma has two consequences. First, recall that in Subsection 2.4 we required that the views in $\mathcal{S}$ determine **ID**. By the lemma, this requirement becomes equivalent to requiring that, for any relation R, $\mathcal{S}$ fully covers some attribute X. Second, the lemma gives us a simple criterion for checking whether $\mathcal{S}$ is consistent. By Theorem 2.17, this holds iff there is no arbitrage between the views in $\mathcal{S}$. The lemma implies that the only risk of arbitrage is between a full cover $\Sigma_{R.Y}$ and a selection view $\sigma_{R.X=a}$, hence:

Proposition 3.2. *$\mathcal{S}$ is consistent iff for every relation R, any two attributes X, Y of R and any constant $a \in Col_{R.X}$:*

$$p(\sigma_{R.X=a}) \leq \sum_{b \in Col_{R.Y}} p(\sigma_{R.Y=b})$$

Note that now consistency is independent of the database instance; this is unlike Subsection 2.7, where we showed that consistency may change with the database.

The Queries. We would like to support a rich query language that buyers can use, while ensuring tractability for the price computation. We start with an upper bound on the data complexity of pricing. We say that a query Q has PTIME data complexity if $Q(D)$ can be computed in polynomial time in the size of D. UCQ queries, datalog queries, and extensions of datalog with negation and inequalities have PTIME data complexity[4].

Theorem 3.3. *Assume $\mathbf{V} \subseteq \Sigma$. Let Q be any monotone query that has PTIME data complexity. Then, $D \vdash \mathbf{V} \twoheadrightarrow Q$ for $\mathbf{V} \subseteq \Sigma$ can be decided in PTIME data complexity.*

We give the proof of this in the Appendix, Appendix Appendix C.

Corollary 3.4. *Let $\mathbf{Q}$ be a bundle of monotone queries that have polynomial time data complexity. Then* Price($\mathbf{Q}$) *is in NP (data complexity).*

Proof. To check if $p_D^{\mathcal{S}}(\mathbf{Q}) \leq k$, guess a subset of selection views $\mathbf{V} \subseteq \Sigma$, then check that both $D \vdash \mathbf{V} \twoheadrightarrow \mathbf{Q}$ (which is equivalent to $D \vdash \mathbf{V} \twoheadrightarrow Q_i$, for all $Q_i \in \mathbf{Q}$, by Lemma 2.6) and that $\sum_{V \in \mathbf{V}} p(V) \leq k$. □

Thus, the restriction to selection queries has lowered the complexity of price computation from Σ_2^p (Corollary 2.18) to NP. However, for some conjunctive queries the price is NP-complete (the proof is in Appendix Appendix D):

Theorem 3.5 (NP-Complete Queries)**.** Price(Q) *is NP-complete (data complexity) when Q is any of the following queries:*

$$H_1(x, y, z) = R(x, y, z), S(x), T(y), U(z) \tag{3}$$
$$H_2(x, y) = R(x), S(x, y), T(x, y) \tag{4}$$
$$H_3(x, y) = R(x), S(x, y), R(y) \tag{5}$$
$$H_4(x) = R(x, y) \tag{6}$$

If Q is one of H_1, H_2, H_3 then the pricing complexity remains NP-complete even is the database instance D is restricted s.t. $Q(D) = \emptyset$.

Thus, we cannot afford to price every conjunctive query. In Subsection 3.1, we introduce a class of conjunctive queries whose prices can be computed in PTIME. In Subsection 3.2, we study the complexity of *all* conjunctive queries without self-joins, and establish a dichotomy for pricing into PTIME and NP-complete.

3.1. A PTIME Algorithm

We define a class of conjunctive queries, called *Generalized Chain Queries*, denoted GChQ, and we provide a non-trivial algorithm that computes their prices in polynomial time.

We consider conjunctive queries with interpreted unary predicates $C(x)$ that can be computed in PTIME: that is, we allow predicates like $x > 10$ or User-Defined-Predicate(x), but not $x < y$. A conjunctive query is *without self-joins* if each relation R_i occurs at most once in Q; e.g. query H_3 in Theorem 3.5 has a self-join (since R occurs twice), the other three queries are without self-joins. A conjunctive query is *full* if all variables in the body appear in the head; e.g. queries H_1, H_2, H_3 are full, while H_4 is not. We restrict our discussion to full, conjunctive queries without self-joins. We abbreviate such a query with $Q = R_0, R_1, \dots, R_k, C_1, \dots, C_p$, where each R_i is an atomic relational predicate, and each C_j is an interpreted unary predicate; we assume the order $R_1, \dots, R_k$ to be fixed. For $0 \leq i \leq j \leq k$, we denote $Q_{[i,j]}$ the full conjunctive $Q_{[i,j]} = R_i, R_{i+1}, \dots, R_j$ (ignoring the unary predicates). For example, if $Q(x, y, z) = R(x), S(x, y), T(y), U(y, z), V(z)$, then $Q_{[1:2]}(x, y) = S(x, y), T(y)$. If k is the index of the last relational predicate, then we abbreviate $Q_{[j,k]}$ with $Q_{[j:*]}$. Denote $Var(Q)$ the set of variables in Q.

Definition 3.6. *A generalized chain query,* GChQ*, is a full conjunctive query without self-joins, Q, such that, for all i, $|Var(Q_{[0:i-1]}) \cap Var(Q_{[i:*]})| = 1$. We denote x_i the unique variable shared by $Q_{[0:i-1]}$ and $Q_{[i:*]}$. The (not necessarily distinct) variables $x_1, \dots, x_k$ are called* join variables. *All other variables are called* hanging variables.

In other words, a GChQ query is one in which every join consists of only one shared variable. Note that the definition ignores the interpreted unary predicates occurring in Q. The following are some examples of GChQ queries:

$$\begin{aligned}
&Q_1(x, y) = R(x), S(x, y), T(y)\\
&Q_2(x, y, z, w) = R(x, y), S(y, z), T(z), U(z), V(z, w)\\
&Q_3(x, y, z, u, v, w) = R(x, y), S(y, u, v, z), T(z, w), U(w)
\end{aligned}$$

On the other hand, none of the queries in Theorem 3.5 are GChQ: the atoms in queries H_1 and H_2 cannot be ordered to satisfy Definition 3.6, H_3 has a self-join, and H_4 is not a full query.

We can now state our main result in this paper:

Theorem 3.7 (Main Theorem)**.** *Assume that all explicit price points in $\mathcal{S}$ are selection queries. Then, for any* GChQ *query, one can compute its price in PTIME (data complexity).*

Before we give the algorithm, we illustrate pricing with an example.

R

X
a_1
a_2

S

X	Y
a_1	b_1
a_1	b_2
a_2	b_2
a_3	b_2
a_4	b_1

T

Y
b_1
b_3

$$Q(x, y) = R(x), S(x, y), T(y)$$
$$Q_{[0:1]}(x, y) = R(x), S(x, y)$$
$$Q_{[1:2]}(x, y) = S(x, y), T(y)$$

$$Col_x = \{a_1, a_2, a_3, a_4\}$$
$$Col_y = \{b_1, b_2, b_3\}$$

(a)

$Q_{[0:1]}(D) =$

X	Y
a_1	b_1
a_1	b_2
a_2	b_2

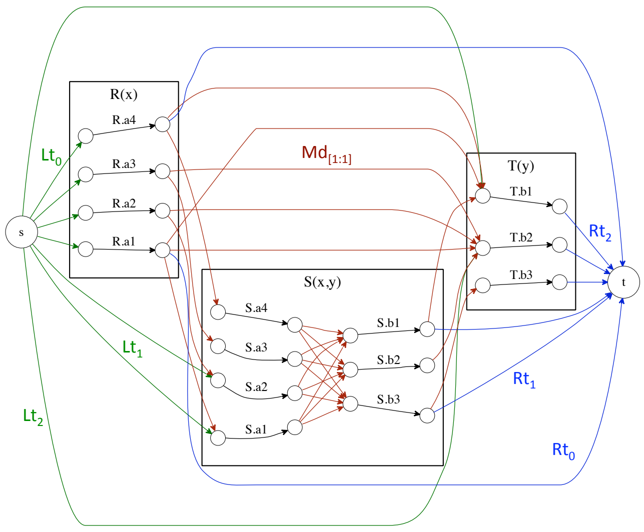
$Q_{[1:2]}(D) =$

X	Y
a_1	b_1
a_4	b_1

$Q_{[0:2]}(D) = Q(D)$

X	Y
a_1	b_1

(b)



(c)

Figure 1: (a) The database D and query Q for Example 3.8. (b) The answers to the partial queries $Q_{[0:1]}, Q_{[1:2]}, Q_{[0:2]}$. (c) The flow graph describing the example (see Theorem 3.13).

Example 3.8. *Consider $Q = R(x), S(x, y), T(y)$ over the database D in Figure 1(a). We have $Q(D) = \{(a_1, b_1)\}$. There are 14 possible selection queries: $\Sigma_{R.X} = \{\sigma_{R.X=a_1}, \sigma_{R.X=a_2}, \sigma_{R.X=a_3}, \sigma_{R.X=a_4}\}$, $\Sigma_{S.X} = \{\sigma_{S.X=a_1}, \sigma_{S.X=a_2}, \sigma_{S.X=a_3}, \sigma_{S.X=a_4}\}$, and similarly for $S.Y$ and $T.Y$. Suppose $\mathcal{S}$ assigns the price \$1 to each selection query.*

To compute the price of Q, we need to find the smallest set $\mathbf{V} \subseteq \Sigma$ that "determines" Q: that is, forall D' s.t. $\mathbf{V}(D) = \mathbf{V}(D')$, the query must return the same answer

$\{(a_1, b_1)\}$ on D', as on D. First, $\mathbf{V}$ must guarantee that (a_1, b_1) is an answer, and for that it must ensure that all three tuples $R(a_1), S(a_1, b_1), T(b_1)$ are in D'; for example, it suffices to include in $\mathbf{V}$ the views $\mathbf{V}_0 = \{\sigma_{R.X=a_1}, \sigma_{S.X=a_1}, \sigma_{T.Y=b_1}\}$ (we could have chosen $\sigma_{S.Y=b_1}$ instead of $\sigma_{S.X=a_1}$). Second, $\mathbf{V}$ must also ensure that none of the other 11 tuples (a_i, b_j) are in the answer to Q. $\mathbf{V}_0$ is not sufficient yet. For example, consider the tuple (a_3, b_2), which is not in the answer. Let $D' = D \cup \{R(a_3), T(b_2)\}$; then $\mathbf{V}_0(D) = \mathbf{V}_0(D')$, since $\mathbf{V}_0$ does not inquire about either $R(a_3)$ or $T(b_2)$, yet $Q(D')$ contains (a_3, b_2). Thus, $\mathbf{V}$ must ensure that either $R(a_3)$ is not in D', or that $T(b_2)$ is not in D'. Continuing this reasoning, leads us to the following set of views $\mathbf{V} = \{\sigma_{R.X=a_1}, \sigma_{R.X=a_3}, \sigma_{R.X=a_4}, \sigma_{S.X=a_1}, \sigma_{S.X=a_2}, \sigma_{T.Y=b_1}, \sigma_{T.Y=b_2}\}$. The reader may check that this is a minimal set that determines Q, hence the price of Q is $p_D^{\mathcal{S}}(Q) = 6$.

We can also generalize the algorithm to GChQ query bundles, which are defined as follows.

Definition 3.9. *A* GChQ *query bundle is a set $\mathbf{Q}$ of* GChQ *queries without interpreted predicates, such that any two queries $Q, Q' \in \mathbf{Q}$ only share in common a prefix and/or a suffix: $\exists i, j, m : Q_{[0:i-1]} = Q'_{[0:i-1]}, Q_{[j:*]} = Q'_{[m:*]}$, and $Q_{[i:j-1]}$, $Q'_{[i:m-1]}$ have no common relation names.*

For example, the bundle $\{Q_1 = S(x, y), R(y, z), U(z),\ Q_2 = S(x, y), T(y, z),\ Q_3 = S(x, y), T(y, z), U(z)\}$ is a GChQ bundle. To simplify the presentation, we discuss here only single queries. We now describe the algorithm, which consists of the four steps below.

STEP 1: Remove Atomic Predicates. Suppose Q has a variable x with an atomic predicate $C(x)$: here we simply shrink the column[6] of x to $Col'_x = \{a \in Col_x \mid C(a) = true\}$, thus removing all constants that do not satisfy C. Let $\mathcal{S}' \subseteq \mathcal{S}$ be obtained by removing all selection views that refer to these constants, and similarly $D' \subseteq D$ be the database obtained by filtering on the predicate C. Finally, let Q' be the query obtained from Q by removing the predicate $C(x)$. We prove in Proposition Appendix G.2 that $p_{D'}^{\mathcal{S}'}(Q') = p_D^{\mathcal{S}}(Q)$.

To illustrate this construction, let $Q(y, w, z) = R(y), S(y, w, z), T(z), w = a_1$ and $Col_w = \{a_1, a_2, a_3\}$. Then, we restrict the column of w to $\{a_1\}$, remove the views $\sigma_{S.W=a_2}, \sigma_{S.W=a_3}$ from $\mathcal{S}$ to obtain $\mathcal{S}'$, filter D on $w = a_1$ to obtain D', and then compute the price of $Q'(x, y, z) = R(y), S(y, w, z), T(z)$.

STEP 2: Remove Multiple Variable Occurrences from Each Atom. We only sketch this step, and defer the details to Proposition Appendix G.4. Suppose a variable x occurs twice in the atom $R(x, x, z)$, where R has schema $R(X, Y, Z)$. Let $R'(X, Z)$ be a new relation name s.t. $Col_{R'.X} = Col_x$, and set the prices on $R'.X$ as follows: $p(\sigma_{R'.X=a}) = \min\{p(\sigma_{R.X=a}), p(\sigma_{R.Y=a})\}$. We prove that the price of the new query (obtained by replacing the atom $R(x, x, z)$ with $R'(x, z)$) is the same as the price of Q.

STEP 3: Removing Hanging Variables. Recall that a *hanging* variable is one that occurs in only one atom of Q; by the previous step, it only occurs in one position $R.X$. We prove in Appendix Appendix E the following:

[6] If x occurs on several attribute positions $R.X$, $S.Y$, etc, then we may assume w.l.o.g. that $Col_{R.X} = Col_{S.Y} = \ldots$ and denote it with Col_x.

Lemma 3.10. *Let x be a hanging variable in Q, occurring in the attribute position $R.X$. Let $\mathbf{V} \subseteq \Sigma$. If $D \vdash \mathbf{V} \twoheadrightarrow Q$ then either (a) $\mathbf{V}$ fully covers $R.X$ or (b) $D \vdash (\mathbf{V} \backslash \Sigma_{R.X}) \twoheadrightarrow \mathbf{Q}$ (in other words, every view in $\mathbf{V}$ referring to $R.X$ is redundant).*

Thus, when computing the price of Q, for each hanging variable we need to consider two cases: either fully cover it, or not cover it at all. We claim that each of these cases becomes another price computation problem, namely for the query Q', obtained from Q by replacing R with R' (obtained from R by removing the attribute $R.X$), on the database D' obtained from D by projecting out $R.X$:

Lemma 3.11. *Let $R.X$ be an attribute containing a hanging variable in Q, $\mathbf{V} \subseteq \Sigma$, and $\mathbf{V}' = \mathbf{V} \setminus \Sigma_{R.X}$.*

- *If $\mathbf{V}$ fully covers $R.X$, let Y be any attribute $Y \neq X$ of R. Then $D \vdash \mathbf{V} \twoheadrightarrow Q$ iff $D' \vdash \mathbf{V}', \Sigma_{R'.Y} \twoheadrightarrow Q'$.*
- *If $\mathbf{V}$ does not fully cover $R.X$, then $D \vdash \mathbf{V} \twoheadrightarrow Q$ iff $D' \vdash \mathbf{V}' \twoheadrightarrow Q'$.*

We prove this as part of Proposition Appendix G.6. The lemma gives us an algorithm for removing hanging variables: compute two prices for Q', and take the minimum. The first price corresponds to the case when $R.X$ is fully covered: in that case, we give out R' for free (by setting all prices $\sigma_{R'.Y=a}$ to 0, for some other attribute Y) and compute the price of Q': then, add to that the true cost of the full cover $\Sigma_{R.X}$, i.e. $\sum_a p(\sigma_{R.Y=a})$. The second price corresponds to the case when $R.X$ is not covered at all, and is equal to the price of Q'. For a simple example, if $Q(x, y, z) = R(x, y), S(y, z), T(z)$, then $Q'(y, z) = R'(y), S(y, z), T(z)$. Let p_1 be the price of Q' where we set all prices of $\sigma_{R'.Y=b}$ to 0; let p_2 be the regular price of Q' (where all prices are unchanged, but the views $\sigma_{R.X=a}$ are removed); return $\min(p_1 + p(\Sigma_{R.X}), p_2)$. In general, we need to repeat this process once for every hanging variable; thus, we end up computing 2^k prices, if there are k attributes with hanging variables.

STEP 4: Reduction to Maximum Flow. Finally, we have reached the core of the algorithm. At this point, the query is a *Chain Query*:

Definition 3.12. *A* Chain Query *is a full conjunctive query without self-joins, $Q = R_0, R_1, \ldots, R_k$ s.t.: (a) every atom R_i is either binary or unary, (b) any two consecutive atoms R_i, R_{i+1} share exactly one variable, denoted x_i, (c) the first and the last atoms are unary, $R_0(x_0), R_k(x_k)$. Denote* ChQ *the set of chain queries.*

We show that the price of chain query can be we reduced to the Min-Cut problem, which is the dual of the Max-Flow graph problem and can be solved in polynomial time [13].

Given a chain query Q, denote x_i, x_{i+1} the variables occurring in R_i: if R_i is unary, then $x_i = x_{i+1}$. In particular, $x_0 = x_1$ and $x_k = x_{k+1}$ since the first and last atoms are unary. Thus, each query $Q_{[i:j]} = R_i, \ldots, R_j$ has variables $x_i, \ldots, x_{j+1}$. Let us also define $Q_{[i:i-1]} = Col_{x_i} = Col_{R_{i-1}.Y} \cap Col_{R_i.X}$. Define the *left-, middle-, and right-partial-answers*:

$$\begin{aligned} Lt_i &= \Pi_{x_i}(Q_{[0:i-1]}(D)), & 0 \leq i \leq k \\ Md_{[i:j]} &= \Pi_{x_i, x_{j+1}}(Q_{[i:j]}(D)), & 1 \leq i \leq k, i-1 \leq j \leq k-1 \\ Rt_j &= \Pi_{x_{j+1}}(Q_{[j+1:k]}(D)), & 0 \leq j \leq k \end{aligned}$$

We construct the following graph G. Its vertices are:

- A source node s and a target (sink) node t.
- For each attribute $R.X$ and constant $a \in Col_{R.X}$, we introduce two nodes: $v_{R.X=a}$ and $w_{R.X=a}$.

Its edges are:

View edges: For each attribute $R.X$ and constant $a \in Col_{R.X}$ we create the edge:

$$v_{R.X=a} \xrightarrow{view} w_{R.X=a}$$

Capacity = the price[7] $p(\sigma_{R.X=a})$ in $\mathcal{S}$.

Tuple edges: For each binary atom $R(X,Y)$, and constants $a \in Col_{R.X}$, $b \in Col_{R.Y}$, we create:

$$w_{R.X=a} \xrightarrow{tuple} v_{R.Y=b}$$

Capacity = ∞.

Skip edges: For all partial answers we create the edges:

$$\begin{aligned} s &\xrightarrow{skip} v_{R_i.X=a} && \text{if } a \in Lt_i \\ w_{R_{j-1}.Y=b} &\xrightarrow{skip} v_{R_{i+1}.X=a} && \text{if } (b,a) \in Md_{[j:i]} \\ w_{R_j.Y=b} &\xrightarrow{skip} t && \text{if } a \in Rt_j \end{aligned}$$

In all cases, capacity = ∞.

In particular, since $Lt_0 = Col_{x_0}$, $Md_{[i:i-1]} = Col_{x_i}$, $Rt_k = Col_{x_k}$ we also have the following skip edges:

$$\begin{aligned} s &\xrightarrow{skip} v_{R_0.X=a} \\ w_{R_{i-1}.Y=a} &\xrightarrow{skip} v_{R_i.X=a} \\ w_{R_k.Y=a} &\xrightarrow{skip} t \end{aligned}$$

We explain now the intuition behind the graph construction, and will also refer to Figure 1 (b) and (c), which illustrates the graph for Example 3.8. Notice that the edges of finite capacity in G are in one-to-one correspondence with the views in $\mathcal{S}$. The main invariant (which we prove in Subsection Appendix F.1) is: *for every set of edges C of finite capacity, C is a "cut" (it separates s and t) if and only if the corresponding set of views $\mathbf{V}$ determines the query.* Before justifying this invariant, note that the core of the graph consists of sequences of three edges:

$$v_{R_i.X=a} \xrightarrow{view} w_{R_i.X=a} \xrightarrow{tuple} w_{R_i.Y=b} \xrightarrow{view} w_{R_i.Y=b}$$

for all binary relations $R_i(X,Y)$ and constants $a \in Col_{R_i.X}, b \in Col_{R_i.Y}$. (Unary relations have just one *view* edge.) Consider a possible answer to Q, $t = (u_1, u_2, \ldots, u_k)$,

[7]If the query has no explicit price in $\mathcal{S}$ then capacity = ∞.

where $u_1 \in Col_{x_1}, \ldots, u_k \in Col_{x_2}$. If $D \vdash \mathbf{V} \twoheadrightarrow Q$, then, for all D' s.t. $\mathbf{V}(D) = \mathbf{V}(D')$, $\mathbf{V}$ must ensure two things: if $t \in Q(D)$ then it must ensure that $t \in Q(D')$, and if $t \notin Q(D)$ then it must ensure that $t \notin Q(D')$. Take the first case, $t \in Q(D)$. For each $i = 0, \ldots, k$, denoting $a = u_i$ and[8] $b = u_{i+1}$, we have: $a \in Lt_i$ (is a left partial answer), $R_i(a, b) \in D$, and $b \in Rt_i$ (is a right partial answer). Hence there are two skip edges:

$$s \xrightarrow{skip} v_{R_i.X=a} \qquad\qquad w_{R_i.Y=b} \xrightarrow{skip} t$$

Combined with the three edges above, they form an $s - t$ path: thus, any cut of finite capacity must include one of the two *view* edges, hence, the corresponding set of views $\mathbf{V}$ includes either $\sigma_{R_i.X=a}$ or $\sigma_{R_i.Y=b}$, ensuring $R_i(a, b) \in D'$. Since this holds for any i, it follows that D' has all the tuples needed to ensure $t \in Q(D')$. For example, in Figure 1 the answer $(a_1, b_1) \in Q(D)$ leads to three $s - t$ paths:

$$\begin{array}{ll}
s \xrightarrow{skip} v_{R.X=a_1} \xrightarrow{view} & w_{R.X=a_1} \xrightarrow{skip} t \\
s \xrightarrow{skip} v_{S.X=a_1} \xrightarrow{view} & w_{S.X=a_1} \xrightarrow{tuple} v_{S.Y=b_1} \xrightarrow{view} w_{S.Y=a_1} \xrightarrow{skip} t \\
 & \quad s \xrightarrow{skip} v_{T.Y=b_1} \xrightarrow{view} w_{T.Y=b_1} \xrightarrow{skip} t
\end{array}$$

Any cut ensures $R(a_1), S(a_1, b_1), T(b_1)$ are present.

Take the second case, $t \notin Q(D)$. Then some of the tuples $R_i(u_i, u_{i+1})$ are missing from D, and $\mathbf{V}$ must ensure that *at least one* is missing. The sequence $u_1, \ldots, u_k$ consists of partial answers, alternating with missing tuples. We are interested only in the latter and the skip edges help by skipping over the partial answers. Thus the missing tuples are on a path from s to t. For an illustration, assume that exactly two tuples are missing, $R_i(u_i, u_{i+1})$ and $R_j(u_j, u_{j+1})$; denoting $a = u_i, b = u_{i+1}, c = u_j, d = u_{j+1}$ we have:

$$a \in Lt_i, (a, b) \notin Md_{[i:i]}, (b, c) \in Md_{[i+1:j-1]}, (c, d) \notin Md_{[j:j]}, d \in Rt_j$$

leading to the following $s - t$ path:

$$\begin{array}{l}
s \xrightarrow{skip} v_{R_i.X=a} \xrightarrow{view} w_{R_i.X=a} \xrightarrow{tuple} w_{R_i.Y=b} \xrightarrow{view} w_{R_i.Y=b} \\
\xrightarrow{skip} v_{R_j.X=c} \xrightarrow{view} w_{R_j.X=c} \xrightarrow{tuple} w_{R_j.Y=d} \xrightarrow{view} w_{R_j.Y=d} \xrightarrow{skip} t
\end{array}$$

To summarize, we prove the following in Subsection Appendix F.1:

Theorem 3.13. *The cost of the minimum cut in G is equal to the price of Q. Therefore, the price of Q can be computed in polynomial time, by reduction to* Min-cut.

3.2. A Dichotomy Theorem

Are there any other queries besides GChQ whose data complexity is in PTIME? The answer is *yes*. We study them here, and give a full characterization of the complexity of all conjunctive queries without self-joins, showing that for each query its complexity is either PTIME or NP-complete. Note that our characterization applies to *all* queries

[8]When $i = k$ then $u_i = u_{i+1}$, hence $a = b$.

without self-joins, not just full queries. However, it only applies to single queries, not to query bundles: we leave open whether query bundles admit a similar dichotomy as single queries.

We start by characterizing the PTIME class. Clearly, all GChQ queries are in PTIME. By definition, every GChQ query is connected: it is easy to check that PTIME queries are closed under cartesian products:

Proposition 3.14. *Assume that Q is disconnected, and partitioned into $Q(\bar{x}_1, \bar{x}_2) : -Q_1(\bar{x}_1), Q_2(\bar{x}_2)$, where x_1, x_2 are disjoint sets of variables. Then,*

$$p_D^{\mathcal{S}}(Q) = \begin{cases} \min\{p_D^{\mathcal{S}}(Q_1), p_D^{\mathcal{S}}(Q_2)\} & \text{if } Q_1(D) = Q_2(D) = \emptyset, \\ p_D^{\mathcal{S}}(Q_1) & \text{if } Q_1(D) = \emptyset,\ Q_2(D) \neq \emptyset, \\ p_D^{\mathcal{S}}(Q_2) & \text{if } Q_2(D) = \emptyset,\ Q_1(D) \neq \emptyset, \\ p_D^{\mathcal{S}}(Q_1) + p_D^{\mathcal{S}}(Q_2) & \text{else} \end{cases}$$

We prove this proposition, along with the converse reduction, in Proposition Appendix G.1. As a consequence, the complexity of any disconnected query is no larger than that of any of its connected components.

A more surprising class of queries that admits a PTIME algorithm is the the class of *cyclic queries*:

Theorem 3.15. *For any integer k,* Price(C_k) *is in PTIME, where $C_k(x_1, \dots, x_k) = R_1(x_1, x_2), \dots, R_k(x_k, x_1)$.*

The algorithm for computing C_k is in Subsection Appendix F.2. It is technically the most difficult result in this paper, and is quite different from the reduction to Min-cut that we used for GChQ, suggesting that these two classes cannot be unified in a natural way. The class of queries C_k is also much more brittle than GChQ: adding a single unary predicate makes the query NP-hard. For example, see the query H_2 in Theorem 3.5: it is obtained by adding one unary predicate to C_2, and is NP-hard. By contrast, we can add freely unary predicates to GChQ.

We conclude our analysis with the following theorem, whose proof is in Appendix Appendix G:

Theorem 3.16 (Dichotomy Theorem)**.** *Let $\mathcal{S}$ contain only selection views (in Σ) and Q be a CQ w/o self-joins. The data complexity for* Price(Q) *is the following:*

- *If Q has connected components $Q_1, \dots, Q_k$, then: if all components Q_i are in PTIME, it is in PTIME, and if one component Q_i is NP-complete, Q is NP-complete.*
- *Else if Q is neither full nor boolean, it is NP-complete.*
- *Else if Q is a boolean query, then let Q^f be the corresponding full query (add all variables to the head); then the complexity of Q is the same as that of Q^f.*
- *Else if Q is a full CQ, let Q' be obtained from Q by removing all hanging variables, constants and multiple occurrences of a variable in the same atom: (a) if Q' is a* GChQ *then it is PTIME, (b) if $Q' = C_k$ for some k, then it is also PTIME, (c) otherwise, Q is NP-complete.*

4. Discussion

We end this this paper with a brief discussion on loose ends and design choices.

Step v.s. smooth pricing function. The pricing function $p^{\mathcal{S}}$ is a step function: its range is always finite, because the arbitrage-price $p_D^{\mathcal{S}}(\mathbf{Q})$ is always the sum of a subset of prices from $\mathcal{S}$. In some applications, this may be too limiting. One example of interest is in selling private data, where the price should be proportional to the degree of privacy revealed by the query: since privacy mechanisms add a noise that can be tuned continuously (e.g. the ε parameter in differential privacy [16]), one expects the pricing function to also vary continuously. Studying "smooth" pricing functions is part of future work.

Pricing and query containment. The price should *not* be required to be monotone w.r.t. query containment. Recall that two queries (of the same arity) are said to be contained if $Q_1(D) \subseteq Q_2(D)$ for any database D. If Q_2 always returns at least as much data as Q_1, one might insist that $p_D(Q_1) \leq p_D(Q_2)$. We argue against this.

Example 4.1. *Consider $Q_1(x, y) = R(x), S(x, y)$ and $Q_2(x, y) = S(x, y)$. Then, $Q_1 \subseteq Q_2$, but the information in Q_1 may be more valuable than that in Q_2. For example, $S(x, y)$ may be the list of the top 500 companies and their stock price, while $R(x)$ may be an analyst's confidential list of 5 companies with very high potential for growth. Clearly, the seller wants to set $p_D(Q_1) \gg p_D(Q_2)$.*

There is also a theoretical argument: if p_D is arbitrage-free *and* monotone w.r.t. query containment, then all Boolean queries have the same price! Indeed, let T be the Boolean query that is always true, i.e. $T(D) = \mathit{true}$ for any database D, and let Q be any Boolean query. We have $Q \subseteq T$, hence $p_D(Q) \leq p_D(T)$; on the other hand, $D \vdash Q \twoheadrightarrow T$, which implies $p_D(T) \leq p_D(Q)$.

Price updates. What happens if the seller adds price points to $\mathcal{S}$? We prove next that, as long the price points remain consistent, the prices never increase; in other words, the seller can only add more discounts, but cannot raise the prices (of course, one can modify $\mathcal{S}$ to raise prices, but prices do not increase through additions to $\mathcal{S}$.)

Proposition 4.2. *If a set of price points $\mathcal{S}$ is consistent for D and $\mathcal{S}' \supseteq \mathcal{S}$ is also consistent for D, then $\forall \mathbf{Q} : p_D^{\mathcal{S}'}(\mathbf{Q}) \leq p_D^{\mathcal{S}}(\mathbf{Q})$.*

Proof. The key observation is that $p_D^{\mathcal{S}'}$ will also be a valid pricing function for $\mathcal{S}$ under D, since it is arbitrage-free and also, if $(\mathbf{V}, p) \in \mathcal{S}$, then $p_D^{\mathcal{S}'}(\mathbf{V}) = p$. For the sake of contradiction, assume that there exists $D, \mathbf{Q}$ such that $p_D^{\mathcal{S}}(\mathbf{Q}) < p_D^{\mathcal{S}'}(\mathbf{Q})$. Then, by Lemma 2.13, the pricing function $\max\{p_D^{\mathcal{S}}, p_D^{\mathcal{S}'}\}$ also is valid, which contradicts the uniqueness of $p_D^{\mathcal{S}}$. $\square$

Selections on Multiple Attributes. The PTIME algorithm in Subsection 3.1 allows explicit prices only on selection queries on single attributes, e.g. $\sigma_{R.X=a}$. A natural question is whether one can extend it to prices on two or more attributes, e.g. $\sigma_{R.X=a,R.Y=b}$. The answer to this question varies. For Chain Queries (Definition 3.12) this is possible: simply modify the flow graph by setting the capacity of the TUPLE-EDGE $(w_{R.X=a}, v_{R.Y=b})$ to $p(\sigma_{R.X=a,R.Y=b})$ instead of ∞. For Generalized Chain Queries, however, this is not possible in general. In fact, even for a very simple query, $Q(x, y, z) =$

$R(x, y, z)$, if $\mathcal{S}$ has prices on all these types of selection queries: $\sigma_{R.X=a}$, $\sigma_{R.Y=b}$, $\sigma_{R.Z=c}$, $\sigma_{R.X=a,R.Y=b,R.Z=c}$, then we prove in the full version of this paper that computing the price of Q is NP-hard.

5. Related Work

There exist many independent vendors selling data online [1, 2, 6, 14, 32] and Amazon cloud users can sell their S3 data for a profit [7]. In addition, digital market services for data have recently emerged in the cloud [9, 19, 30], which enable content providers to upload their data and make it available either freely or for a fee, and support some limited forms of views. In the case of Infochimps [19], the seller can set prices on APIs (modeled as selection queries) or entire datasets. The Azure DataMarket [9] uses data subscriptions with query limits: *i.e.*, a group of records returned by a query and that can fit on a page (currently 100) is called a *transaction.* WebScaled is a pre-launch startup providing a marketplace for datasets from ongoing Web crawls: social graphs, lists of sites using a particular advertising platforms, frequency of specific doctypes and other HTML elements, etc. [30, 31]. Apollo Mapping sells access to satellite imagery [8]. The approach that we develop in this paper extends these pricing methods with the ability to interpolate prices for *arbitrary queries* over a seller's database.

While the interaction between data management and economics has been studied in the database research community before [15, 29], to the best of our knowledge, this paper is the first to study the problem of data pricing, with the exception of a short vision paper that we recently published [10].

There is a rich literature on pricing information products (*e.g.*, [20, 28]). We were most influenced by Shapiro and Varian [28], who argued that the price of *information products* is quite different from that of *physical goods*, and proposed a new theory for pricing information products, based on the notion of versions. The difference is that information products have very high fixed costs, while the marginal costs are tiny. For example, the cost of conducting a detailed consumer survey in several countries is very high, while the cost of distributing the resulting data tiny (copying a file). As a consequence, the price of information products cannot be determined by traditional means (production costs and competition), but must be linked to the value that the buyers place on the data. Different buyers may use the data in different ways, and should be charged different prices. For example, a retailer may be willing to pay a high price for the entire consumer survey, while a journalist may only be willing to pay a small amount for a few interesting statistics from the consumer survey. In order to leverage these differences in willingness to pay, Shapiro and Varian conclude that information products should be offered in different *versions*, at different prices. Our approach extends version-based pricing to relational data, by associating a version of the product to each query that a user may ask.

The classic notion of determinacy was extensively studied by Nash, Segoufin and Vianu [27, 24, 25], who have investigated both the decidability question, and the subtle relationship between determinacy and rewritability. We have reviewed information theoretic determinacy earlier ($\mathbf{V} \twoheadrightarrow Q$ if forall D, D', $\mathbf{V}(D) = \mathbf{V}(D')$ implies $Q(D) = Q(D')$). Rewritability is specific to a query language $\mathcal{R}$: Q can be *rewritten using* $\mathbf{V}$ *in the language* $\mathcal{R}$ if there exists a query $R \in \mathcal{R}$ s.t. $Q(D) = R(\mathbf{V}(D))$ for all D. One goal of this line of research was to establish tight bounds on the language $\mathcal{R}$; a surprising result is an example where both $\mathbf{V}$ and Q are conjunctive queries, yet R is non-monotone, proving that no monotone language is sufficient for CQ to CQ rewriting. In our query pricing framework we do not impose any restriction on the language used for rewriting; in other words, we assume that the user has unrestricted computational power, and as a consequence the two notions become equal. A second goal of the research [27, 24, 25] is to study the decision problem for determinacy: it is shown to be undecidable even for Unions of Conjunctive Queries, and its status is open for Conjunctive Queries. However, several classes of CQ queries where determinacy is well-behaved have been found: path queries [5], syntactic restrictions of FO and UCQ which are called packed FO and UCQ [22] and monadic views [25]. Determinacy has also been examined in the restricted setting of aggregate queries [18].

A key difference in our paper is that we consider instance-based determinacy, where determinacy is defined with respect to a given view extension. While applications like data integration or semantic caching require instance-independent determinacy, in query pricing the current state of the database cannot be ignored;. Instance-based determinacy is identical to the notion of *lossless views* [12] under the exact view assumption. The definition is based on the notion of *certain answers* [3]. We note that instance-specific reasoning also arises in data security and authorization views: in that context, Zhang and Mendelzon study *conditional query containment*, where the containment is conditioned on a particular view output [33].

Finally, we should mention that, on the surface, our complexity results for pricing seem related to complexity results for computing *responsibility* [23]. The PTIME algorithm for responsibility is also based on network flow, and some queries have the same complexity for both the pricing and the responsibility problems. However, the connection is superficial: the price of H_2 is NP-complete, while its responsibility is in PTIME; and the price of C_3 is in PTIME while its responsibility is NP-complete.

6. Conclusion

We have presented a framework for pricing relational data based on queries. The seller sets explicit prices on some views, while the buyer may ask arbitrary queries; their prices are determined automatically. We gave several results: an explicit formula for the price, a polynomial time algorithm for pricing generalized chain queries, and a dichotomy theorem for conjunctive queries without self-joins. We also gave several results on instance-based determinacy.

Interesting future work includes considering competition: when a seller sets prices for her data, she needs to consider other data instances on the market that offer "related" data, to avoid arbitrage. This requires reasoning about mappings between the different data sources, and these mappings are often approximate in practice. Another is the interaction between pricing and privacy. Most of the literature on data privacy [16] focuses on restricting access to private information. Privacy, however, has a broader definition, and usually means the ability of the data owner to control how her private information is used [26]. Setting a price for private data is one form of such control that we plan to investigate.

Acknowledgments. This work is supported in part by the NSF and Microsoft through NSF grant CCF-1047815.

References

[1] http://gnip.com.

[2] http://www.patientslikeme.com.

[3] Abiteboul, S., and Duschka, O. M. Complexity of answering queries using materialized views. In *PODS* (1998), ACM Press, pp. 254–263.

[4] Abiteboul, S., Hull, R., and Vianu, V. *Foundations of Databases.* Addison-Wesley, 1995.

[5] Afrati, F. N. Rewriting conjunctive queries determined by views. In *MFCS* (2007), pp. 78–89.

[6] http://www.aggdata.com/.

[7] Using Amazon S3 Requester Pays with DevPay. http://docs.amazonwebservices.com/AmazonDevPay/latest/DevPayDeveloperGuide/index.html?S3RequesterPays.html.

[8] http://www.apollomapping.com/.

[9] https://datamarket.azure.com/.

[10] Balazinska, M., Howe, B., and Suciu, D. Data markets in the cloud: An opportunity for the database community. *Proc. of the VLDB Endowment 4*, 12 (2011).

[11] Calvanese, D., Giacomo, G. D., Lenzerini, M., and Vardi, M. Y. Answering regular path queries using views. In *ICDE* (2000), pp. 389–398.

[12] Calvanese, D., Giacomo, G. D., Lenzerini, M., and Vardi, M. Y. Lossless regular views. In *PODS* (2002), L. Popa, Ed., ACM, pp. 247–258.

[13] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. *Introduction to Algorithms, Second Edition.* The MIT Press and McGraw-Hill Book Company, 2001.

[14] http://www.customlists.net/.

[15] Dash, D., Kantere, V., and Ailamaki, A. An economic model for self-tuned cloud caching. In *Proc. of the 25th ICDE Conf.* (2009), pp. 1687–1693.

[16] Dwork, C. A firm foundation for private data analysis. *Commun. ACM 54*, 1 (2011), 86–95.

[17] Gottlob, G., and Senellart, P. Schema mapping discovery from data instances. *J. ACM 57*, 2 (2010).

[18] Grumbach, S., and Tininini, L. On the content of materialized aggregate views. *J. Comput. Syst. Sci. 66*, 1 (2003), 133–168.

[19] http://www.infochimps.com/.

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

[21] Libkin, L. *Elements of Finite Model Theory.* Springer, 2004.

[22] Marx, M. Queries determined by views: pack your views. In *PODS* (2007), L. Libkin, Ed., ACM, pp. 23–30.

[23] Meliou, A., Gatterbauer, W., Moore, K. F., and Suciu, D. The complexity of causality and responsibility for query answers and non-answers. *PVLDB 4*, 1 (2010), 34–45.

[24] Nash, A., Segoufin, L., and Vianu, V. Determinacy and rewriting of conjunctive queries using views: A progress report. In *ICDT* (2007), pp. 59–73.

[25] Nash, A., Segoufin, L., and Vianu, V. Views and queries: Determinacy and rewriting. *ACM Trans. Database Syst. 35*, 3 (2010).

[26] Schneier, B. *Secrets & Lies, Digital Security in a Networked World.* John Wiley & Sons, 2000.

[27] Segoufin, L., and Vianu, V. Views and queries: determinacy and rewriting. In *PODS* (2005), C. Li, Ed., ACM, pp. 49–60.

[28] Shapiro, C., and Varian, H. R. Versioning: The smart way to sell information. *Harvard Business Review 76* (November-December 1998), 106–114.

[29] Stonebraker et al. Mariposa: a wide-area distributed database system. *VLDB Journal 5*, 1 (1996), 048–063.

[30] http://webscaled.com/.

[31] Web marketing. Google group forum post, http://groups.google.com/group/webmarketing/msg/c6643da409802f85.

[32] http://www.xignite.com/.

[33] Zhang, Z., and Mendelzon, A. O. Authorization views and conditional query containment. In *ICDT* (2005), pp. 259–273.

Appendix A. Case Study

CustomLists.net [14] is an online vendor of marketing data including business and consumer contact information[9]. The CustomList database includes different tables. Table `Business(name, email, address, city, state, zip, phone, fax, category_code, category_descr, url)` contains 28.6 million entries describing American businesses. The table `Consumer(email, fname, lname, address, city, state, zip))` contains 59.8 million records with consumer information. The price for the entire `Business` table is $399. The price is only $199 for a single state and it is only $299 for the subset of American businesses that also have an email address (*i.e.*, for the query `select * from Business where email NOT NULL`). The consumer table costs $699 and the subset of consumers in each state costs only $349. Buyers thus have the flexbility to purchase the views that most closely match their needs.

Whether the prices satisfy the arbitrage-free property depends on the database content. If all businesses have an email address then the corresponding view determines the Business table, creating an arbitrage opportunity.

With the above prices, however, a buyer who wishes to purchase only Alabama businesses in the automotive category must purchase and pay for *all* of Alabama's business information. Similarly, a buyer interested only in consumers who interact with businesses in the automotive category, must buy the *entire* consumers table. In this paper, we posit that such restrictions limit business opportunities because some buyers may not be willing to pay extra for the unnecessary data. Ideally, the seller and buyer should be able to negotiate and price personalized data products. AggData [6] is an example data seller that provides such custom solutions. However, such one-on-one negotiations are not scalable to large numbers of buyers and large numbers of personalized data products.

Similarly, one can not expect that the seller will be able to manually set the prices of every possible query ahead of time. There are two reasons for this. First, if the goal is to allow any query to be asked on the database, the seller must manually determine the prices for an extremely large (infinite actually) sets of queries. Moreover, one cannot expect that the seller will be a database expert and thus have the knowledge about query determinacy to set valid prices to the queries.

On the other hand, the data marketplace can not be solely responsible for setting the prices of the views. The seller must be able to regulate the prices according to economic models, demand on the market, and other constraints. Thus, the seller must be able to provide a rough sketch of how to price the queries. In order to reconcile the pricing decisions of the seller and the constraints of pricing relational queries, we propose a framework where the seller specifies some pricing points, *i.e.*, the prices of some queries, and the underlying data marketplace decides whether these points are consistent and, if yes, infers the prices of the other queries.

Continuing the CustomLists example, our framework should automatically infer the price for the queries that ask only for automotive industries in Alabama or consumers interacting with such industries.

Here, we should emphasize that the goal is to price the queries based on the information content of the query result. For example, a query that returns an empty result may not be priced necessarily to zero, since it may reveal information about the database.

[9]All data is opt-in and permission-based according to the company website.

Consider the following query: who are the Monterey, CA consumers who interact with businesses in the Beads Wholesale category. In the case that the query result is empty, the query will not necessarily be priced to zero, as it reveals information: we now know that none of the consumers located in Monterey purchase Beads.

Appendix B. Determinacy

In this section, we discuss in detail the two notions of determinacy that are mentioned in this paper: *information-theoretic* and *instance-based* determinacy.

Appendix B.1. Information-Theoretic Determinacy

Here, we review the *information-theoretic* determinacy introduced by Nash, Segoufin and Vianu [25].

Definition Appendix B.1 (Information-Theoretic Determinacy)**.** *Let* $\mathbf{V}$*,* $\mathbf{Q}$ *be two query bundles. We say that* $\mathbf{V}$ determines $\mathbf{Q}$*, in notation* $\mathbf{V} \twoheadrightarrow \mathbf{Q}$*, if for all* $D_1, D_2 \in \mathit{Inst}_{\mathbf{R}}$*,* $\mathbf{V}(D_1) = \mathbf{V}(D_2)$ *implies* $\mathbf{Q}(D_1) = \mathbf{Q}(D_2)$*.*

Let $\mathbf{R_V}$ and $\mathbf{R_Q}$ denote the output schemas of the query bundles $\mathbf{V}$, $\mathbf{Q}$ respectively. In other words, $\mathbf{V}$ is a function $\mathbf{V} : \mathit{Inst}_{\mathbf{R}} \rightarrow \mathit{Inst}_{\mathbf{R_V}}$, and similarly $\mathbf{Q} : \mathit{Inst}_{\mathbf{R}} \rightarrow \mathit{Inst}_{\mathbf{R_Q}}$. The following is easy to check.

Proposition Appendix B.2. $\mathbf{V} \twoheadrightarrow \mathbf{Q}$ *iff there exists a function* $f : \mathit{Inst}_{\mathbf{R_V}} \rightarrow \mathit{Inst}_{\mathbf{R_Q}}$ *such that* $\mathbf{Q} = f \circ \mathbf{V}$*. In other words, for all instances* D*,* $\mathbf{Q}(D) = f(\mathbf{V}(D))$*.*

This justifies the interest in information-theoretic determinacy: $\mathbf{V} \twoheadrightarrow \mathbf{Q}$ holds iff, for any instance D, the answer $\mathbf{Q}(D)$ can be obtained by examining only the view output $\mathbf{V}(D)$, and not the instance itself. It is also easy to check that:

Proposition Appendix B.3. *The information-theoretic determinacy* $D \vdash \mathbf{V} \twoheadrightarrow \mathbf{Q}$ *satisfies all properties in Section 2.*

Notice that the determinacy relationship satisfies also vacuously monotonicity, since it does not depend on the database instance. The following facts are known about information-theoretic determinacy.

- The following problem is undecidable: given UCQ's $V_1, \ldots, V_k, Q$, decide whether $V_1, \ldots, V_k \twoheadrightarrow Q$.
- It is an open problem whether the following is decidable: given CQs $V_1, \ldots, V_k, Q$, decide whether $V_1, \ldots, V_k \twoheadrightarrow Q$.
- The following is also an open problem: given CQs $V_1, \ldots, V_k, Q$ such that $V_1, \ldots, V_k \twoheadrightarrow Q$, what is the minimum language $\mathcal{L}$ that we cane express the function f?

The fact that information-theoretic determinacy is very hard to decide is an argument against using this determinacy for pricing. The other characteristic of information-theoretic determinacy is that the prices are *independent* of the database instance. Hence, using this determinacy may be appropriate for applications where the seller wants the prices to remain unchanged in the case of updates.

Appendix B.2. Instance-Based Determinacy

In this subsection, we discuss in detail *instance-based determinacy.*

Definition Appendix B.4 (Instance-Base Determinacy)**.** *Let D be an instance and $\mathbf{V}$, $\mathbf{Q}$ be two query bundles. We say that* $\mathbf{V}$ determines $\mathbf{Q}$ given D, *in notation $D \vdash \mathbf{V} \twoheadrightarrow \mathbf{Q}$, if for all $D' \in Inst_{\mathbf{R}}$, $\mathbf{V}(D') = \mathbf{V}(D)$ implies $\mathbf{Q}(D') = \mathbf{Q}(D)$.*

It will be also very useful to keep in mind the notion of *certain answers.*

Definition Appendix B.5 (Certain Answers Under CWA)**.** *Let $\mathbf{V}$ be a set of views, E corresponding view extensions, and Q a query. A tuple t is a* certain answer *(under the closed world assumption, CWA) if $t \in Q(D)$ for every D such that $\mathbf{V}(D) = E$.*

Moreover, let $cert_{Q,\mathbf{V}}(E)$ be the set of certain answers. Alternatively, $cert_{Q,\mathbf{V}}(E) = \bigcap_{D':\mathbf{V}(D')=E} Q(D')$.

The definition of instance-based determinacy is indentical to the definition of lossless views under the exact view assumption proposed in [12]. We next show the equivalence for alternative definitions of instance-based determinacy.

Proposition Appendix B.6. *Let $\mathbf{V}$ be a set of views, a database D, and a query Q. Then, the following are equivalent:*

1. *$\forall D', D''$ s.t. $\mathbf{V}(D') = \mathbf{V}(D'') = \mathbf{V}(D)$: $Q(D') = Q(D'')$.*
2. *$\forall D'$ s.t. $\mathbf{V}(D') = \mathbf{V}(D)$: $Q(D') = cert_{Q,\mathbf{V}}(\mathbf{V}(D))$.*
3. *$\forall D'$ s.t. $\mathbf{V}(D') = \mathbf{V}(D)$: $Q(D') = Q(D)$.*

Proof. For ease of exposition, let $E = \mathbf{V}(D)$. Then,

(1) $\Rightarrow$ (2). Consider a database D' such that $\mathbf{V}(D') = E$. Then, by definition (1), for every D'' such that $V(D'') = E$, we have that $Q(D'') = Q(D')$. Hence, $Q(D') = cert_{Q,\mathbf{V}}(E)$.

(2) $\Rightarrow$ (1). Let us consider databases D', D'' such that $\mathbf{V}(D') = \mathbf{V}(D'') = E$. By definition (2), $Q(D') = cert_{Q,\mathbf{V}}(E)$; hence, $Q(D'') \subseteq Q(D')$. Due to symmetry, we also have that $Q(D'') \subseteq Q(D')$. Thus, $Q(D') = Q(D'')$.

(1) $\Rightarrow$ (3). Consider a database D' such that $\mathbf{V}(D') = E$. Then, for the pair of databases D, D' we have that $\mathbf{V}(D') = \mathbf{V}(D) = E$. Hence, by definition (1), we have that $Q(D') = Q(D)$.

(3) $\Rightarrow$ (1). Let D', D'' be a pair of databases such that $\mathbf{V}(D') = \mathbf{V}(D'') = E$. Since $\mathbf{V}(D') = E$, by definition (1) we have that $Q(D') = Q(D)$. Similarly, since $\mathbf{V}(D'') = E$, $Q(D'') = Q(D)$. Thus, $Q(D') = Q(D) = Q(D'')$. □

Our interest in instance-based determinacy is justified by the following proposition, similar to Proposition Appendix B.2:

Proposition Appendix B.7. *For any $\mathbf{V}, \mathbf{Q}$ there exists a function $f : Inst_{\mathbf{R}_\mathbf{V}} \to Inst_{\mathbf{R}_\mathbf{Q}}$ such that, forall D, if $D \vdash \mathbf{V} \twoheadrightarrow \mathbf{Q}$, then $\mathbf{Q}(D) = f(\mathbf{V}(D))$.*

Moreover, if there exists a function f such that for all D' such that $\mathbf{V}(D') = \mathbf{V}(D)$, $f(\mathbf{V}(D')) = \mathbf{Q}(D')$, then $D \vdash \mathbf{V} \twoheadrightarrow \mathbf{Q}$.

Proof. Define f as follows:

$$f(\mathbf{E}) = \begin{cases} \mathbf{Q}(D) & \text{if there exists } D \text{ such that } \mathbf{V}(D) = \mathbf{E}, \\ \bot & \text{else.} \end{cases}$$

Now, consider a database D such that $D \vdash \mathbf{V} \twoheadrightarrow \mathbf{Q}$. We need to show that $f(\mathbf{V}(D)) = \mathbf{Q}(D)$. By the construction of f, there exists some D' such that $\mathbf{V}(D') = \mathbf{V}(D)$. Thus, by Definition Appendix B.4, we also have that $\mathbf{Q}(D') = \mathbf{Q}(D)$. It follows that $f(\mathbf{V}(D)) = \mathbf{Q}(D') = \mathbf{Q}(D)$.

For the second part, let $D' : \mathbf{V}(D') = \mathbf{V}(D)$. Then, $\mathbf{Q}(D') = f(\mathbf{V}(D')) = f(\mathbf{V}(D)) = \mathbf{Q}(D)$. □

Thus, we can still compute $\mathbf{Q}$ by examining only the view $\mathbf{V}$, but only for those instances where the instance-based determinacy holds.

Proposition Appendix B.8. *Instance-based determinacy satisfies all properties in Section 2.*

Proof. **Reflexivity:** Let some D' such that $\mathbf{Q}_1(D') = \mathbf{Q}_1(D)$ and $\mathbf{Q}_2(D') = \mathbf{Q}_2(D)$. Since $\mathbf{Q}_1(D') = \mathbf{Q}_1(D)$, $\mathbf{Q}_1, \mathbf{Q}_2$ determine $\mathbf{Q}_1$ under D.

Transitivity: Let $D \vdash \mathbf{Q}_1 \twoheadrightarrow \mathbf{Q}_2$ and $D \vdash \mathbf{Q}_2 \twoheadrightarrow \mathbf{Q}_3$. Let D' such that $\mathbf{Q}_1(D') = \mathbf{Q}_1(D)$. By the first determinacy relation, $\mathbf{Q}_2(D') = \mathbf{Q}_2(D)$. By the second determinacy relation now, $\mathbf{Q}_3(D') = \mathbf{Q}_3(D)$. Hence, $D \vdash \mathbf{Q}_1 \twoheadrightarrow \mathbf{Q}_3$.

Augmentation: Let $D \vdash \mathbf{V} \twoheadrightarrow \mathbf{Q}$. Consider some D' such that $\mathbf{V}(D') = \mathbf{V}(D)$ and $\mathbf{V}'(D') = \mathbf{V}'(D)$. The first equality, together with the determinacy relation imply that $\mathbf{Q}(D') = \mathbf{Q}(D)$. Hence, $D \vdash \mathbf{V}, \mathbf{V}' \twoheadrightarrow \mathbf{Q}, \mathbf{V}'$.

Boundedness: Given $\mathbf{ID}$, we can construct D exactly, and then answer $\mathbf{Q}(D)$. Hence, $D \vdash \mathbf{ID} \twoheadrightarrow \mathbf{Q}$ □

In contrast to information-theoretic determinacy, instance-based determinacy is in general not monotone, as we showed in Example 2.20. More importantly though, instance-based determinacy is decidable for a large class of queries. We prove next Theorem 2.3 in two steps.

Theorem Appendix B.9. *For views and query in UCQ, the combined complexity of* Instance-based determinacy *is in Π_2^p, while the data complexity is in co-NP.*

Proof. We reduce the problem of instance-based determinacy to the problem of finding certain answers. Given a set of views $\mathbf{V}$, a query Q, and the views extension E, to show that $\mathbf{V}$ does *not* determine Q relative to E, it suffices to find a *witness tuple* t such that

1. t is a possible answer: $\exists D_1 : \mathbf{V}(D_1) = E \wedge t \in Q(D_1)$
2. t is not a certain answer: $\exists D_2 : \mathbf{V}(D_2) = E \wedge t \notin Q(D_2)$

Thus, we need to find independently two databases D_1, D_2 such that D_1 satisfies condition (1) and D_2 satisfies condition (2).

Our proof strategy is to find the witness tuple by enumeration over all possibilities. Let us assume that Q has arity k. Let $C = \{c_1, \ldots, c_k\}$ be new distinct constants that do not appear in E. Moreover, let A be the set of all constants appearing in E. Then, it follows from the genericity of the database that we need only check for the witness from

$(A \cup C)^k$. These are at most $(|E| + k)^k$, which is polynomially many to the size of the view extension.

Moreover, in [11], it is shown that checking whether a tuple is a possible answer can be reduced to checking that a tuple is not a certain answer. Indeed, notice that we can rewrite condition (1) as $\exists D_1 : \mathbf{V}(D_1) = E \wedge \{t\} \subseteq Q(D_1) \wedge c \notin (Q'(c) = false)$ (the last condition is trivially correct). However, this is exactly equivalent to checking whether c is not a certain answer for Q' under the views $\mathbf{V}, Q$, where we have to interpret Q under the *open world* assumption. Futher, the size of the new set of views to check for certain answers is linear in the size of $(\mathbf{V}, Q)$.

Hence, conditions (1) and (2) are both equivalent to asking whether t is not a certain answer for certain views and extensions. Additionally, it is known from [3, Th. 3.1] that for UCQs it suffices to look for databases of polynomial size to extensions and views to find the counterexample database. Hence, our problem is now transformed as follows: (a) guess a tuple t (there are polynomially many), (b) guess two databases D_1, D_2 of polynomial size to $E, \mathbf{V}, Q$ and finally (c) check that $\mathbf{V}(D_1) = \mathbf{V}(D_2) = E$, $t \in Q(D_2)$ and $t \notin Q(D_1)$.

Since query evaluation for UCQs has PTIME data complexity and NP-complete combined complexity, we have:

- For data complexity, finding a witness tuple is in NP; hence, instance-based determinacy is in co-NP.
- For combined complexity, finding a witness tuple is in NP^{NP}; hence, instance-based determinacy is in Π_2^P.

This concludes the proof. □

We next show the co-NP completeness of Instance-Based Determinacy in terms of data complexity.

Theorem Appendix B.10. Instance-Based Determinacy *is co-NP hard for CQs.*

Proof. We will reduce Non-3-Colorability to Instance-Based Determinacy. The proof is similar to [3]. Let $G = (V, E)$ be a graph with at least one edge (otherwise 3-colorability is trivial). Fix a relational schema $\mathbf{R}$ with the relations $color(X, Y)$ (node X has color Y) and $edge(X, Y)$ (node X is connected with node Y). Next, consider the bundle $\mathbf{V} = (V_1, V_2, V_3)$, where $V_1(X) = color(X, Y)$, $V_2(Y) = color(X, Y)$ and $V_3(X, Y) = edge(X, Y)$.

The database D is such that $edge(X, Y) = E$ and $color(X, Y)$ assigns exactly one of three colors $\{a, b, c\}$ to a node of G. Let $Q() = edge(X, Y), color(X, Z), color(Y, Z)$, i.e. Q asks whether there exist two neighboring nodes with the same color. We will show that $D \vdash \mathbf{V} \twoheadrightarrow Q$ if and only if G is not 3-colorable. Notice first that any database D' such that $\mathbf{V}(D') = \mathbf{V}(D)$ is equivalent to a color assignment to each node of the graph.

Indeed, assume that G is not 3-colorable. Then, for every coloring Q returns true. Hence, for every database D' such that $\mathbf{V}(D') = \mathbf{V}(D)$, Q is true. This implies that $\mathbf{V}$ determines Q.

For the other direction, assume that $\mathbf{V}$ determines Q under D. Then, for every database D' such that $\mathbf{V}(D') = \mathbf{V}(D)$, Q returns the same answer, true or false. Consider the database D' which assigns to every node the same color. In this case, since G has at least one edge, Q on D' will return true. Hence, Q will always return true. This implies that for any coloring, Q can find a pair of neighbors with the same color; hence, G is not 3-colorable. $\square$

Appendix B.3. Restriction of Instance-Based Determinacy

Let $\twoheadrightarrow$ be any determinacy relation (Definition 2.5). Its *restriction* $D \vdash \mathbf{V} \twoheadrightarrow^* \mathbf{Q}$ is: $\forall D_0, \mathbf{V}(D_0) \subseteq \mathbf{V}(D)$, $D_0 \vdash \mathbf{V} \twoheadrightarrow \mathbf{Q}$. We prove:

Proposition Appendix B.11. *(a) $\twoheadrightarrow^*$ is a determinacy relation (Definition 2.5), (b) $\twoheadrightarrow^*$ is monotone (Definition 2.21) for any monotone$\mathbf{V}$ and any $\mathbf{Q}$, (c) if $p_D^{\mathcal{S}}$ and $q_D^{\mathcal{S}}$ are the arbitrage-prices for $\twoheadrightarrow$ and $\twoheadrightarrow^*$, respectively, then $p_D^{\mathcal{S}}(\mathbf{Q}) \leq q_D^{\mathcal{S}}(\mathbf{Q})$ for all $\mathbf{Q}$, and (d) if $\twoheadrightarrow$ is the instance-based determinacy, then the data complexity of $\twoheadrightarrow^*$ is in coNP.*

Proof. (a) **Reflexivity:** Since $\twoheadrightarrow$ is a determinacy relation, it is reflexive. Hence, $\forall D_0, (\mathbf{V}_1, \mathbf{V}_2)(D_0) \subseteq (\mathbf{V}_1, \mathbf{V}_2)(D), D_0 \vdash \mathbf{V}_1, \mathbf{V}_2 \twoheadrightarrow \mathbf{V}_1$. Hence $D \vdash \mathbf{V}_1, \mathbf{V}_2 \twoheadrightarrow^* \mathbf{V}_1$ and $\twoheadrightarrow^*$ is reflexive.

Transitivity: Given $D \vdash \mathbf{V}_1 \twoheadrightarrow^* \mathbf{V}_2$ and $D \vdash \mathbf{V}_2 \twoheadrightarrow^* \mathbf{V}_3$, and by the definition of $\twoheadrightarrow^*$, we know that $\forall D_0, \mathbf{V}_1(D_0) \subseteq \mathbf{V}_1(D), D_0 \vdash \mathbf{V}_1 \twoheadrightarrow \mathbf{V}_2$ and $\forall D_0, \mathbf{V}_2(D_0) \subseteq \mathbf{V}_2(D), D_0 \vdash \mathbf{V}_2 \twoheadrightarrow \mathbf{V}_3$. Thus, $\forall D_0, \mathbf{V}_1(D_0) \subseteq \mathbf{V}_1(D), D_0 \vdash \mathbf{V}_1 \twoheadrightarrow \mathbf{V}_2, \mathbf{V}_2(D_0) \subseteq \mathbf{V}_2(D), D_0 \vdash \mathbf{V}_2 \twoheadrightarrow \mathbf{V}_3, D_0 \vdash \mathbf{V}_1 \twoheadrightarrow \mathbf{V}_3$. The last subformula comes from the transitivy of $\twoheadrightarrow$. Hence, $\twoheadrightarrow^*$ is transitive.

Augmentation: Given $D \vdash \mathbf{V}_1 \twoheadrightarrow^* \mathbf{V}_2$ and by the definition of $\twoheadrightarrow^*$, we know that $\forall D_0, \mathbf{V}_1(D_0) \subseteq \mathbf{V}_1(D), D_0 \vdash \mathbf{V}_1 \twoheadrightarrow \mathbf{V}_2$. Since $\twoheadrightarrow$ permits augmentation we know that $D_0 \vdash \mathbf{V}_1, \mathbf{V}' \twoheadrightarrow \mathbf{V}_2, \mathbf{V}'$. Thus, $\forall D_0, (\mathbf{V}_1, \mathbf{V}')(D_0) \subseteq (\mathbf{V}_1, \mathbf{V}')(D), D_0 \vdash \mathbf{V}_1, \mathbf{V}' \twoheadrightarrow \mathbf{V}_2, \mathbf{V}'$.

Boundedness Since $\twoheadrightarrow$ is bounded, $\forall D, D \vdash \mathbf{ID} \twoheadrightarrow \mathbf{V}$. Hence $\twoheadrightarrow^*$ is also bounded.

(b) Since $D_1 \subseteq D_2$ and $\mathbf{V}$ is monotone, $\mathbf{V}(D_1) \subseteq \mathbf{V}(D_2)$. Given $D_2 \vdash \mathbf{V} \twoheadrightarrow^* \mathbf{Q}$, we know that $\forall D', \mathbf{V}(D') \subseteq \mathbf{V}(D_2), D' \vdash \mathbf{V} \twoheadrightarrow \mathbf{Q}$. Thus, $\forall D', \mathbf{V}(D') \subseteq \mathbf{V}(D_1) \subseteq \mathbf{V}(D_2), D' \vdash \mathbf{V} \twoheadrightarrow \mathbf{Q}$. Hence, $D_1 \vdash \mathbf{V} \twoheadrightarrow^* \mathbf{Q}$.

(c) Let the minimum priced support for $\mathbf{Q}$ with the determinacy relation $\twoheadrightarrow^*$ be $\mathbf{V} \subseteq \mathcal{S}$. Thus $p(\mathbf{V}) = q_D^{\mathcal{S}}$. But $\mathbf{V}$ is also a support for $\mathbf{Q}$ with the determinacy relation $\twoheadrightarrow$ (because by definition of $\twoheadrightarrow^*$, $D \vdash \mathbf{V} \twoheadrightarrow^* \mathbf{Q}$ implies $D \vdash \mathbf{V} \twoheadrightarrow \mathbf{Q}$). Hence, $p_D^{\mathcal{S}} \leq p(\mathbf{V}) = q_D^{\mathcal{S}}$.

(d) We prove that $\twoheadrightarrow^*$ is in co-NP. We refer the reader to the proof of Theorem Appendix B.9. To show that a set of views $\mathbf{V}$ with views extension E does not determine Q under $\twoheadrightarrow^*$, we need to find a witness tuple t for *any* view extension $E' \subseteq E$. Note that the size of the counterexample database is polynomial in size to the extensions E', and hence polynomial in size of E which is a super-set of E', and the views $\mathbf{V}$. Thus, the data complexity of finding a witness tuple is in NP; hence, the restriction of instance-based determinacy is in co-NP. $\square$

Appendix C. Complexity of Determinacy

In this subsection, we prove Theorem 3.3, namely that $D \vdash \mathbf{V} \twoheadrightarrow Q$ can be decided in PTIME data complexity when $\mathbf{V} \subseteq \Sigma$ and Q is any monotone query that can be evaluated in PTIME.

First, we define two databases D^{min} and D^{max}. For a relation $R(X_1, \dots, X_k)$ in Q, let us call a tuple $t \in Col_{X_1} \times \cdots \times Col_{X_k}$ *invisible in* R if, for any selection $\sigma_{R.X_i=a} \in \mathbf{V}$, $t.X_i \neq a$. Then, for R:

$$R^{D^{min}} = \bigcup_{\sigma_{R.A=c} \in \mathbf{V}} \sigma_{R.A=c}(D) \tag{C.1}$$

$$R^{D^{max}} = R^{D^{min}} \cup \{t \mid t \text{ invisible in } R\} \tag{C.2}$$

Notice that both databases are of polynomial size to the size of the columns and the input database. Moreover, they can be defined only using $\mathbf{V}(D)$, without having access to D. Additionally, $D^{min} \subseteq D^{max}$. The following lemma establishes that both instances we have constructed agree with D on the views D.

Lemma Appendix C.1. $\mathbf{V}(D^{min}) = \mathbf{V}(D^{max}) = \mathbf{V}(D)$.

Proof. We first examine D^{min}. For a view $\sigma_{R.A=c} \in \mathbf{V}$, consider a tuple $t_R \in \sigma_{R.A=c}(D^{min})$. Then, by construction, $t_R \in D$. For the converse, if $t_R \in \sigma_{R.A=c}(D)$, then $t_R \in R^{D^{min}}$ and thus in D^{min}.

As for D^{max}, by definition the new tuples we have added to a relation R are invisible in this relation under the views $\mathbf{V}$, hence they can not appear in any $\mathbf{V}$. Thus, $\mathbf{V}(D^{max}) = \mathbf{V}(D^{min})$. □

The next lemma justifies the way we have constructed D^{min} and D^{max}: they are the minimum and maximum databases respectively that can agree with $\mathbf{V}(D)$.

Lemma Appendix C.2 (Sandwich Lemma)**.** *For any database D' s.t. $\mathbf{V}(D') = \mathbf{V}(D)$: $D^{min} \subseteq D' \subseteq D^{max}$.*

Proof. We will first show that $D^{min} \subseteq D'$. For the sake of contradiction, suppose that there exists some tuple $t_R \in R^{D^{min}}$ such that $t_R \notin D'$. By the definition of D^{min}, $t_R \in \sigma_{R.A=c}(D)$ for a view $\sigma_{R.A=c} \in \mathbf{V}$. However, we also have that $t_R \notin \sigma_{R.A=c}(D')$. This implies that $\mathbf{V}(D') \neq \mathbf{V}(D)$, a contradiction.

We next show that $D^{max} \supseteq D'$. Again, for the sake of contradiction let us assume that there exists some tuple $t_R \in R^{D'}$ such that $t_R \notin D^{max}$. We now distinguish two cases. In the first case, let $t_R \in \sigma_{R.A=c}(D')$ for some $\sigma_{R.A=c} \in \mathbf{V}$. We can apply Lemma Appendix C.1 to obtain that $\sigma_{R.A=c}(D') = \sigma_{R.A=c}(D^{max})$. This would imply that $t_R \in \sigma_{R.A=c}(D^{max})$, a contradiction. Otherwise, $t_R \notin \sigma_{R.A=c}(D')$ for any $\sigma_{R.A=c} \in \mathbf{V}$. This implies in turn that t_R is invisible in R under $\mathbf{V}$. By the construction of D^{max}, we would have that $t_R \in D^{max}$, a contradiction. □

We can now show how to characterize determinacy using the databases D^{min} and D^{max}.

Proposition Appendix C.3. *Let Q be any monotone query. Then, $D \vdash \mathbf{V} \twoheadrightarrow Q$ if and only if $Q(D^{min}) = Q(D^{max})$.*

Proof. For the one direction, let us assume that $Q(D^{min}) = Q(D^{max})$. Consider a database D' such that $\mathbf{V}(D') = \mathbf{V}(D)$. By Lemma Appendix C.2, $D^{min} \subseteq D' \subseteq D^{max}$. Since Q is monotone, it follows that $Q(D^{min}) \subseteq Q(D') \subseteq Q(D^{max})$. By the equality of

$Q(D^{min}), Q(D^{max})$, $Q(D') = Q(D^{min})$. Notice also that we can apply Lemma Appendix C.2 to sandwich D between $Q(D^{min}), Q(D^{max})$. Thus, $Q(D) = Q(D^{min})$ and $Q(D) = Q(D')$.

For the other direction, assume that $Q(D^{min}) \neq Q(D^{max})$. Then, since by Lemma Appendix C.1 we have that $\mathbf{V}(D^{min}) = \mathbf{V}(D^{max}) = \mathbf{V}(D)$, the databases D^{min}, D^{max} form a counterexample for determinacy. □

The algorithm for determinacy computes D^{min}, D^{max} from $\mathbf{V}(D)$ and then checks whether $Q(D^{min}) = Q(D^{max})$, in which case it outputs yes, otherwise no. The validity of the algorithm follows from Proposition Appendix C.3. Moreover, the algorithm runs in PTIME, since the databases D^{min}, D^{max} are of polynomial size and also the evaluation of Q can be done in polynomial data complexity. Theorem 3.3 follows directly from this discussion.

We next show that, under stronger conditions than monotonicity and PTIME evaluation, we can have an algorithm such that its complexity depends only on $|\mathbf{V}(D)|$ and not on the size of the columns (hence, the columns can even be infinite). More specifically, we prove the following.

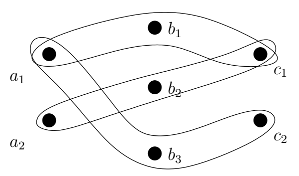
Lemma Appendix C.4. *Let Q be a positive Datalog query without inequalities. Then, there exists a database D^m of polynomial size to $\mathbf{V}(D)$ such that (a) $\mathbf{V}(D^m) = \mathbf{V}(D)$, and (b) $Q(D^{max}) = Q(D^{min})$ if and only if $Q(D^{min}) = Q(D^m)$.*

Proof. The database D^m is constructed from D^{max} as follows. For each column Col_x, consider a fixed value c_x that does not appear in $\mathbf{V}(D)$. Then, construct a function $f : Col \rightarrow Col$, such that each value of Col_x that does not appear in $\mathbf{V}(D)$ is mapped to c_x; otherwise it is mapped to itself. Let $D^m = f(D^{max})$. Clearly, D^m is of size polynomial to $|\mathbf{V}(D)|$ (considering $Q, \mathbf{V}$ fixed). Also, by construction, $D^m \subseteq D^{max}$.

It is also easy to see that $\mathbf{V}(D^m) = \mathbf{V}(D^{max}) = \mathbf{V}(D)$. It remains to show that $Q(D^m) = Q(D^{min})$ iff $Q(D^{max}) = Q(D^{min})$. Indeed, suppose that $Q(D^{max}) = Q(D^{min})$. Then, since D^m agrees with D on $\mathbf{V}$, by Lemma Appendix C.2, $Q(D^m)$ is sandwiched between $Q(D^{min}), Q(D^{max})$ and hence $Q(D^m) = Q(D^{min})$. For the other direction, suppose that $Q(D^{min}) \neq Q(D^{max})$. Since $Q(D^{min}) \subseteq Q(D^{max})$, there exists $t \in Q(D^{max})$ such that $t \notin Q(D^{min})$. Now, Q is in *Datalog* without any inequalities, so $f(t) \in Q(D^m)$. To conclude the proof, we will show that $f(t) \notin Q(D^{min})$. Suppose that $f(t) \in Q(D^{min})$ and consider the tuples $t_1, \ldots, t_k$ that contribute to $f(t)$. By the definition of D^{min}, every tuple t_i is selected by some view in $\mathbf{V}$. Thus, every constant appearing in the t_i's appears also in $\mathbf{V}(D)$. This implies that $f(t) = t$, hence $t \in Q(D^{min})$, a contradiction. □

Appendix D. Hardness Results

In this subsection, we prove the NP-hardness of the queries H_1, H_2, H_3, H_4 and H_5.



(a)

S

x

T

y

U

z

R

x	y	z
a_1	b_1	c_1
a_2	b_2	c_1
a_1	b_3	c_2

(b)

Figure D.2: An example 3-partite 3-uniform Hypergraph and the corresponding database for the reduction to PRICE(H_1).

Proposition Appendix D.1. PRICE(H_1) *is NP-complete, where*

$$H_1(x, y, z) = R(x, y, z), S(x), T(y), U(z)$$

Proof. We will show the NP-hardness also for the queries $H_1'(x, y, z) = R(x, y, z), S(x), T'(y, z)$ and $H_1''(x, y, z) = R(x, y, z), S'(x, y, z)$. In order to present a uniform approach, we will assume a query Q such that the variables x, y, z appear, apart from R, in the relations R_x, R_y, R_z respectively (which are not necessarily different, for example for H_1', $R_y = R_z$).

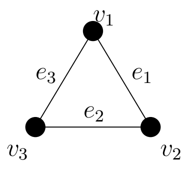
The reduction for Q is from 3-PARTITE 3-UNIFORM HYPERGRAPH VERTEX COVER (3P3UHVC), which is proven NP-complete in [17]. In this, we are given a hypergraph $G(A, B, C, E)$ which is 3-partite (no vertices of the same partition can belong in the same hyperedge) and let A, B, C be the partitions. The graph is also 3-uniform, i.e. each hyperedge contains exactly 3 vertices. The decision version asks whether there exists a vertex cover of G of size $\leq K$.

Let the columns be $Col_x = A$, $Col_y = B, Col_z = C$. Let the database D have R_x^D, R_y^D, R_z^D empty and $R^D = E$ (see Figure D.2). Notice that $Q(D) = \emptyset$. Let us price every selection query on R to 0 and every selection query of the form $\sigma_{R_i.i=a}$ to 1, where $i = x, y, z$. Notice that there exists a straightforward one-to-one mapping from nodes in G to selection queries on the relations R_x, R_y, R_z. Let $\sigma(v)$ be the selection query corresponding to node v. Moreover, let Σ_R denote the set of all selection queries in R. We will show that G has a vertex cover of size K if and only if Q can be determined by a subset of Σ with cost K.

For the one direction, assume that G has a vertex cover C of size K. We will prove that $D \vdash \bigodot_{v \in C} \sigma(v), \Sigma_R \twoheadrightarrow Q$ (notice that the cost of this set of views is exactly K). Since $Q(D) = \emptyset$, it suffices to show that for every D' such that the views agree with D, $Q(D') = \emptyset$. Let $(a, b, c) \in Q(D')$. Since Σ_R contains all views from R, it must be that (a, b, c) also appears in R^D and so (a, b, c) is a hyperedge of G. Hence, it is covered by some node, let it be a. Thus, $\sigma(a) = \sigma_{R_x.x=a}$ belongs in the views we have selected. Since $\sigma_{R_x.x=a}(D') = \sigma_{R_x.x=a}(D) = \emptyset$, a does not join anywhere, so $(a, b, c) \notin Q(D')$, a contradiction.

For the converse direction, assume that there exists a set of views $\mathbf{V}$ that determines Q and has price K. This means that $\mathbf{V}$ has exactly K views selecting from R_x, R_y, R_z: let $v_1, \ldots, v_k$ be the corresponding vertices in G. We will show that $\{v_1, \ldots, v_k\}$ is a vertex cover for G. Indeed, suppose that it is not a vertex cover. Then, there exists an edge $(a, b, c) \in E$ that is not covered; this means that the views $\sigma_{R_x.x=a}, \sigma_{R_y.y=b}, \sigma_{R_z.z=b}$

are missing from $\mathbf{V}$ and also $(a,b,c) \in R^D$. Now, take $D' = D \cup \{R_x(a), R_y(b), R_z(c)\}$ (if $R_x = R_y$, we add $R_x(a,b)$, and if $R_x = R_y = R_z$ we add $R_x(a,b,c)$). It is easy to observe that D' agrees with D for the views $\mathbf{V}$. However, $Q(D') = \{(a,b,c)\} \neq \emptyset$, a contradiction for the determinacy. $\square$



(a)

R

x
v_0

S

x	y
v_1	e_1
v_2	e_2
v_3	e_3

T

x	y
v_2	e_1
v_3	e_2
v_1	e_3

(b)

Figure D.3: An example graph G and the corresponding database for the reduction to Price(H_2).

Proposition Appendix D.2. Price(H_2) *is NP-complete, where*

$$H_2(x,y) = R(x), S(x,y), T(x,y)$$

Proof. We reduce Vertex Cover (VC) to pricing $H_2(x,y)$. Given an undirected graph $G(V,E)$, we construct an instance of $H_2(x,y)$ as follows: (a) take the columns to be $Col_x = V \cup \{v_0\}$ where $v_0 \notin V$ is a dummy value, and $Col_y = E$, (b) $\forall e = (v_1, v_2) \in E$ add (v_1, e) to S and (v_2, e) to T (here, assume an arbitrary orientation of the edges that will be fixed throughout the reduction), *i.e.*, for each edge one endpoint contributes to S and the other to T, and (c) add v_0 to R. Note that $H_2(x,y) = \emptyset$, since $R = \{v_0\}$ and v_0 has no corresponding tuples to join, since $\forall e \in Col_x$, $S(v_0, e)$ and $T(v_0, e)$ are both false.

To construct the price points $\mathcal{S}$, we price the selection queries as follows: (a) $\forall v \in V$, $p(\sigma_{R.x=v}) = 1$ and $p(\sigma_{R.x=v_0}) = 0$, (b) $\forall a \in Col_x$, $p(\sigma_{S.x=a}) = p(\sigma_{T.x=a}) = |E|$, and (c) $\forall e \in Col_y$, $p(\sigma_{S.y=e}) = p(\sigma_{T.y=e}) = 1$. Note that given a set of selections $\mathbf{V}$ that determines H_2, *i.e.*, $D \vdash \mathbf{V} \twoheadrightarrow H_2$, if $\sigma_{S.x=a} \in \mathbf{V}$, then we can replace $\sigma_{S.x=a}$ by $\bigodot_{e \in E} \sigma_{S.y=e}$ to get a new set of selections that determines H_2 and that is no costlier than $\mathbf{V}$. This is because $\forall a \in Col_x : D \vdash \Sigma_{S.y} \twoheadrightarrow \sigma_{S.x=a}$ and $p(\bigodot_{e \in E} \sigma_{S.y=e}) = \sum_{e \in E} 1 = |E| = p(\sigma_{S.x=a})$.

We next show that there exists a vertex cover $C \subseteq V$ of size $\leq k$, iff there exists a set of views $\mathbf{V}$ that determines H_2, such that $p(\mathbf{V}) \leq k + |E|$.

To prove the forward direction, assume that $|C| = k$. Define $\mathbf{V}$ to be the set of selections that includes $\forall v \in C : \sigma_{R.x=v}$ and $\sigma_{R.x=v_0}$. Moreover, $\forall e = (v_i, v_j) \in E$:

1. if $v_i, v_j \in C$, include $\sigma_{S.y=e}$
2. if $v_i \in C$ and $v_j \notin C$, include $\sigma_{S.y=e}$
3. if $v_i \notin C$ and $v_j \in C$, include $\sigma_{T.y=e}$

(notice that $v_i, v_j \notin C$ is not possible since C is a vertex cover). It is easy to see that $p(\mathbf{V}) = |C| + |E|$. Further, $D \vdash \mathbf{V} \twoheadrightarrow H_2$. Indeed, consider a possible answer (v, e). If $v \in C$, $\sigma_{R.x=v} = \emptyset$ is selected and hence any database D' that agrees with $\sigma_{R.x=v} \in \mathbf{V}$

can not contain (v,e). If $v \notin C$, then either v is an endpoint of e or it is not. In the latter case, the selection over e (either $\sigma_{S.y=e}$ or $\sigma_{T.y=e}$) does not include (v,e); hence, for any database D' agreeing with the selection, $(v,e) \notin H_2(D')$. In the former case, w.l.o.g. we can assume that $e=(v,v_j)$. Then, v_j must belong in C and this is case (3) in the construction of $\mathbf{V}$; hence, $\sigma_{T.y=e} \in \mathbf{V}$. However, by the construction of T it is equal to $(v,e) \notin \sigma_{T.y=e}(D)$ and thus, for any database D' that agrees with $\mathbf{V}$, $(v,e) \notin H_2(D')$.

For the converse direction, consider a set of selections $\mathbf{V}$ that determines H_2 with $p(\mathbf{V}) = k + |E|$, where $k < |V|$. For costlier selections, the proposition is trivially valid by choosing the cover $C = V$. Now, for each $e \in E$, $\mathbf{V}$ necessarily includes $\sigma_{S.y=e}$ or $\sigma_{T.y=e}$, else we can construct database D' with $(v_0,e) \in S^{D'}, T^{D'}$. Then, $(v_0,e) \in H_2(D') \neq \emptyset = H_2(D)$ even though $\mathbf{V}(D) = \mathbf{V}(D')$, leading to a contradiction. Further, for $e=(v_i,v_j)$, if both $\sigma_{S.y=e}, \sigma_{T.y=e} \in \mathbf{V}$, then we can replace $\sigma_{S.y=e}$ with $\sigma_{R.x=v_i}$ or replace $\sigma_{T.y=e}$ with $\sigma_{R.x=v_j}$ and have an equal cost determinacy set. Indeed, the views $\sigma_{S.y=e}, \sigma_{T.y=e} \in \mathbf{V}$ prevents (v,e) (for any $v \in Col_x$) to belong to $H_2(D)$. And, so does $\sigma_{R.x=v_i}, \sigma_{T.y=e} \in \mathbf{V}$, since $\sigma_{T.y=e} = \{(v_j,e)\}$ and $\sigma_{R.x=v_i} = \emptyset$. A symmetrical argument holds for replacing $\sigma_{T.y=e}$ with $\sigma_{R.x=v_j}$.

Thus, each edge has a selection from exactly one of S or T ($|E|$ of them), and for each edge, at least one of its endpoints is selected in R. Suppose not: then, consider the edge $e=(v_i,v_j)$ for which this not true. Then, $\sigma_{R.x=v_i}, \sigma_{R.x=v_j} \notin \mathbf{V}$. Moreover, w.l.o.g. let $\sigma_{S.x=e} \in \mathbf{V}$ and $\sigma_{T.y=e} \notin \mathbf{V}$. Since $\sigma_{S.x=e}(D) = (v_i,e)$, we can add tuples $T(v_j,e), R(v_j)$ to create a database D' that agrees with D in the views, but now $(v_j,e) \in H_2(D')$. Hence the selections from R lead to a valid vertex cover of size $p(\mathbf{V}) - |E| = k$. □

Proposition Appendix D.3. Price(H_3) *is NP-complete, where*

$$H_3(x,y) = R(x), S(x,y), R(y)$$

Proof. The reduction is from Vertex Cover. Consider a graph $G(V,E)$, where we ask whether there exists a vertex cover of size $\leq k$. Fix a schema $\{R(X), S(X,Y)\}$ and let $Col_X = Col_Y = V$. Let a database D such that $R^D = \emptyset$ and $S^D = E$ (fix an arbitrary direction for the undirected edges E). As for the prices, let $p(\sigma_{S.X=c}) = p(\sigma_{S.Y=c}) = 0$ and $p(\sigma_{R.X=c}) = 1$. Notice that $H_3(D) = \emptyset$. We now show that G has a vertex cover of size k if and only if H_3 can be determined with cost k.

For the one direction, assume that G has a vertex cover $C = \{v_1,\ldots,v_k\}$ of size k. Let Σ_S be the set of all the selections that include relation S. Then, we will show that for $\mathbf{V}^C = \bigodot_{i=1}^k \sigma_{R.X=v_i}, \Sigma_S$, we have $D \vdash \mathbf{V}^C \twoheadrightarrow H_3$. The cost of this set of views is k. In order to show the determinacy, it suffices to show that for every D' such that the view agrees with D, $H_3(D') = \emptyset$. For the sake of contradiction, assume that $(a,b) \in H_3(D')$. Then, it must be that $(a,b) \in S^{D'}$ and $(a),(b) \in R^{D'}$. Since $\sigma_{S.X=a} \in \mathbf{V}^C$, $(a,b) \in S^D$. Hence, (a,b) is an edge of the graph and one of the two vertices, let is be a, is covered by the vertex cover. Thus, $\sigma_{R.X=a} \in \mathbf{V}^C$ as well. Since $\sigma_{R.X=a}(D) = \emptyset$, and $(a) \in \sigma_{R.X=a}(D')$, we have reached a contradiction.

For the converse direction, assume that there exists a set of views $\mathbf{V}$ that determines H_3 and has price k. This means that $\mathbf{V}$ has exactly k views of the form $\sigma_{R.X=v_i}$. We will show that $\{v_1,\ldots,v_k\}$ is a vertex cover for G. Indeed, suppose that it is not a vertex cover. Then, there exists an edge $(a,b) \in E$ that is not covered; this means that $\sigma_{R.X=a}, \sigma_{R.X=b} \notin \mathbf{V}$. We can also assume w.l.o.g. that $(a,b) \in \mathbf{V}(D)$. Let $D' =$

$D \cup \{R(a), R(b)\}$. It is easy to observe that D' agrees with D for the views $\mathbf{V}$. However, $H_3(D') = \{(a,b)\} \neq \emptyset$, a contradiction for the determinacy. $\square$

Proposition Appendix D.4. PRICE(H_4) *is NP-complete, where*

$$H_4(x) = R(x,y)$$

Proof. We prove the hardness by reduction from SET COVER. Consider a universe U and a family of subsets $\mathcal{S} \subseteq 2^U$. We ask for the minimum size subset of $\mathcal{S}$ that covers every element from U.

In order to do the reduction, we define a schema $\{R(X,Y)\}$ such that $Col_X = \mathcal{S}$ and $Col_Y = U$. Then, consider the database D where $(a,b) \in R^D$ iff $b \in a$. Notice that $H_4(D) = U$. Let us price $p(\sigma_{R.Y=c}) = |\mathcal{S}|$, $p(\sigma_{R.X=S}) = 1$. We will show that U has a set cover of size k if and only if H_4 can be determined by a set of views of price k.

For the one direction, consider a set cover $\{S_1, \dots, S_k\}$. Then, construct the set of views $\mathbf{V} = \{\sigma_{R.X=S_i} \mid i = 1, k\}$. We will show that $D \vdash \mathbf{V} \twoheadrightarrow H_4$ (notice that $\mathbf{V}$ costs k). Let us consider a database D' such that $\mathbf{V}(D') = \mathbf{V}(D)$ and for the sake of contradiction assume that some $a \notin H_4(D')$. However, by the construction of $\mathbf{V}$, some set S covers a, hence $(S,a) \in \sigma_{R.X=S}(D)$ and $\sigma_{R.X=S} \in \mathbf{V}$. This implies that $(S,a) \in \sigma_{R.X=S}(D')$, a contradiction.

For the converse direction, suppose that H_4 is determined by a set of views $\mathbf{V}$ of price k. We can assume w.l.o.g. that no view from $R.Y$ belongs in $\mathbf{V}$, since in this case $k \geq |\mathcal{S}|$ and we could instead buy all the views $\sigma_{R.X=a}$ for a cost of $|\mathcal{S}|$ and determine H_4 trivially. Hence, $\mathbf{V}$ contains views of the form $\sigma_{R.X=S_i}$ for $i = 1, \dots k$. We will show that $S_1, \dots, S_k$ is a set cover. Suppose not; then, there exists an element $a \in U$ not covered by any set. Consider the database $D' = D \setminus \{R(S,a) \mid a \in S\}$. It is easy to see that D', D agree with the views, since $\mathbf{V}$ contains no view of the form $\sigma_{R.X=S}$ for $a \in S$. Moreover, $a \notin H_4(D')$, a contradiction. $\square$

Proposition Appendix D.5. PRICE(H_5) *is NP-hard, where*

$$\begin{aligned} H_5(x_1, x_2, x_3, y) = & R_1(x_1), S_1(x_1, y), \\ & R_2(x_2), S_2(x_2, y), \\ & R_3(x_3), S_3(x_3, y). \end{aligned}$$

Proof. We prove the hardness of H_5 by applying the same idea presented in the hardness proof of H_1, namely by showing a reduction from 3P3UHVC.

Let the graph be $G(A,B,C,E)$. Consider the schema $\{R_i(X_i), S_i(X_i,Y)\}$ $(i = 1,2,3)$ and let $Col_{X_1} = A$, $Col_{X_2} = B$, $Col_{X_3} = C$. Let $Col_Y = \{1, \dots, |E|\}$. Next, consider any one-to-one mapping $m : E \to Col_Y$. For the construction of D, let $R_i^D = \emptyset$. The construction of the S_i's is as follows: for every edge $(a,b,c) \in E$, we add to D the tuples $S_1(a, m(a,b,c))$, $S_2(b, m(a,b,c)), S_3(c, m(a,b,c))$. Notice that $H_5(D) = \emptyset$. Finally, we set the prices for the selections on S_i to be zero and for the R_i to be 1. We prove that G has a vertex cover of size K if and only if H_5 can be determined by a subset of selection views with cost K. Let Σ_S be the set of selections from S_1, S_2, S_3.

For the one direction, assume that G has a vertex cover C of size K. For a vertex $v \in C$, let $\sigma(v)$ be the corresponding selection. We will prove that $D \vdash \bigodot_{v \in C} \sigma(v), \Sigma_S \twoheadrightarrow Q$. It suffices to show that for every D' such that the views agree with D, $H_5(D') =$

$\emptyset$. Let $(a,b,c,m) \in H_5(D')$. Since Σ_S contains all views from S_i, it must be that $S_1(a,m), S_2(b,m), S_3(c,m) \in D$. However, this means that $m = m(a,b,c)$ and thus (a,b,c) is a hyperedge of G. Hence, it is covered by some node, let it be a. Then, $\sigma(a) = \sigma_{R_1.X=a}$ belongs in the views we have selected. Since $\sigma_{R_1.X=a}(D') = \sigma_{R_1.X=a}(D) = \emptyset$, a does not join anywhere, so $(a,b,c,m) \notin H_5(D')$, a contradiction.

For the converse direction, assume that there exists a set of views $\mathbf{V}$ that determines H_5 and has price K. This means that $\mathbf{V}$ has exactly K views selecting from R_1, R_2, R_3: let $v_1, \dots, v_k$ be the corresponding vertices in G. We will show that $\{v_1, \dots, v_k\}$ is a vertex cover for G. Indeed, suppose that it is not a vertex cover. Then, there exists an edge $(a,b,c) \in E$ that is not covered; this means that the views $\sigma_{R_1.X=a}, \sigma_{R_2.Y=b}, \sigma_{R_3.Z=c}$ are missing from $\mathbf{V}$. Let w.l.o.g. $R_1(a, m(a,b,c)), R_2(b, m(a,b,c)), R_3(c, m(a,b,c)) \in D$. Now, take $D' = D \cup \{R_1(a), R_2(b), R_3(c)\}$. It is easy to observe that D' agrees with D for the views $\mathbf{V}$. However, $Q(D') = \{(a,b,c)\} \neq \emptyset$, a contradiction. □

Appendix E. Technical Tools

We develop here a set of tools that will be useful for analyzing the determinacy relation in the case of views with selections.

Let us say that a set of selections $\mathbf{V} \subseteq \Sigma$ *covers* a tuple $t = (a_1, \dots, a_k)$ in a relation $R(X_1, \dots, X_k)$ if for some $i = 1, \dots, k$, $\sigma_{R_i.X_i=a_i} \in \mathbf{V}$. Notice that this definition is independent of any database instance.

Lemma Appendix E.1. *Let a database D, $\mathbf{V} \subseteq \Sigma$ and Q a full CQ. Then, $D \vdash \mathbf{V} \twoheadrightarrow Q$ if and only if, for any $t = (a_1, \dots, a_k) \in Col_{X_1} \times \cdots \times Col_{X_k}$:*

- *If $t \in Q(D)$, then for any projection t_R of t at some relation R in Q, t_R is covered by $\mathbf{V}$.*
- *If $t \notin Q(D)$, there exists a projection t_R of t at some relation R in Q such that $t_R \notin R^D$ and $\mathbf{V}$ covers t_R.*

Proof. Let $t \in Q(D)$. If $\mathbf{V}$ does not cover some t_R, then for the database $D^- = D - \{R(t_R)\}$ we have that $\mathbf{V}(D^-) = \mathbf{V}(D)$, but $t \notin Q(D^-)$; hence $\mathbf{V}$ can not determine Q. On the other hand, if $\mathbf{V}$ covers every tuple t_R for any R, any D' that agrees with $\mathbf{V}(D)$ discovers correctly that $t \in Q(D)$.

Let $t \notin Q(D)$ and assume that for any atom R in Q such that the projection $t_R \notin R^D$, $\mathbf{V}$ does not cover it. Then, consider the database D^+ that adds to D all such tuples. Clearly, the views $\mathbf{V}$ are not modified; however, $t \in Q(D^+)$, a contradiction. For the converse direction, if $\mathbf{V}$ covers at least one tuple $t_R \notin R^D$, any database that agrees with $\mathbf{V}(D)$ knows that t can not belong in $Q(D)$. □

Definition Appendix E.2. *Given a database D and a query Q, we call a subset T^c of the database D* critical *if $Q(D) \neq Q(D - T^c)$.*

For a set of tuples T and a view V, let us define $(V - T)(D) = V(D - T)$. We also extend this notation to $(\mathbf{V} - T)(D) = \{(V - T)(D) \mid V \in \mathbf{V}\}$.

Lemma Appendix E.3. *If $D \vdash \mathbf{V} \twoheadrightarrow Q$, where $\mathbf{V} \subseteq \Sigma$ and Q is a full CQ, then $D \vdash (\mathbf{V} - T) \twoheadrightarrow Q$, for any non-critical subset T of D.*

Proof. Since $D - T \subseteq D$, we can apply Proposition 2.22 to obtain that $D - T \vdash \mathbf{V} \twoheadrightarrow Q$. Hence, it follows from Proposition Appendix B.7 that there exists a function f that for any D' s.t. $\mathbf{V}(D') = \mathbf{V}(D - T)$, f has the property: $f(\mathbf{V}(D')) = Q(D - T)$. Now, consider any database D^0 such that $(\mathbf{V} - T)(D^0) = (\mathbf{V} - T)(D)$. We will show that $f((\mathbf{V} - T)(D^0)) = Q(D)$. Notice first that $\mathbf{V}(D^0 - T) = (\mathbf{V} - T)(D^0) = (\mathbf{V} - T)(D) = \mathbf{V}(D - T)$. Thus, $f((\mathbf{V} - T)(D^0)) = f(\mathbf{V}(D^0 - T)) = Q(D - T) = Q(D)$, where the last equality follows from the fact that T is a non-critical. $\square$

We next show the main theorem of this subsection.

Theorem Appendix E.4. *Let Q be a full CQ and $D \vdash \mathbf{V} \twoheadrightarrow Q$, where $\mathbf{V} \subseteq \Sigma$. Let $\mathbf{V}^0 \subseteq \mathbf{V}$ such that a tuple $t \in \mathbf{V}^0(D)$ is either non-critical or belongs in $(\mathbf{V} \setminus \mathbf{V}^0)(D)$. Then, $D \vdash \mathbf{V} \setminus \mathbf{V}^0 \twoheadrightarrow Q$.*

Proof. Let $T = \mathbf{V}^0(D)$ and let $T^c \subseteq T$ contain the individually non-critical tuples of T, i.e. the tuples $t \in T^c$ such that $Q(D) = Q(D - \{t\})$. We first show that:

Lemma Appendix E.5. *The set T^c is non-critical.*

Proof. Suppose that T^c is critical; then $Q(D - T^c) \neq Q(D)$. Since $D - T^c \subseteq D$ and Q is monotone, $Q(D - T^c) \subset Q(D)$ and hence there exists some tuple $t \in Q(D)$ such that $t \notin Q(D - T^c)$. However, since Q is full, it suffices to delete a single tuple $t' \in T^c$ from D to delete t from $Q(D)$. Thus, t' is critical, a contradiction. $\square$

We can now apply Lemma Appendix E.3 to obtain that $D \vdash (\mathbf{V} - T^c) \twoheadrightarrow Q$. We next show that $D \vdash ((\mathbf{V} \setminus \mathbf{V}^0) - T^c) \twoheadrightarrow (\sigma_{R.y=c} - T^c)$ for any $\sigma_{R.y=c} \in \mathbf{V}^0$. Indeed, by our assumption, $(\sigma_{R.y=c} - T^c)(D)$ contains only tuples that also belong to some view other than those in $\mathbf{V}^0$ and hence it can be precisely reconstructed.

Applying repeatedly augmentation and transitivity, we obtain that $D \vdash ((\mathbf{V} \setminus \mathbf{V}^0) - T^c) \twoheadrightarrow (\mathbf{V} - T^c)$. One more application of transitivity results in: $D \vdash ((\mathbf{V} \setminus \mathbf{V}^R) - T^c) \twoheadrightarrow Q$. For the final step, notice that

Lemma Appendix E.6. *For any $V \in \Sigma$, $D \vdash V \twoheadrightarrow (V - T^c)$.*

Proof. Notice that for selection views, $(V - T^c)(D) = V(D) - T^c$; hence, $(V - T^c)(D)$ can be constructed from $V(D)$ by applying the function $f(V(D)) = V(D) - T^c$. $\square$

Applying this lemma together with the transitivity property, we obtain that $D \vdash \mathbf{V} \setminus \mathbf{V}^0 \twoheadrightarrow Q$. $\square$ $\square$

Lemma Appendix E.7. *If $D \vdash \sigma_{R.A=d}, \sigma_{R.B=d}, \mathbf{V}^0 \twoheadrightarrow Q$, where $\mathbf{V} \subseteq \Sigma$, and Q requires that $R.A = R.B$, then $D \vdash \sigma_{R.B=d}, \mathbf{V}^0 \twoheadrightarrow Q$.*

Proof. From Theorem Appendix E.4, it suffices to show that every tuple $t_R \in \sigma_{R.A=d}$ is either non-critical or belongs in $\sigma_{R.B=d}$. Indeed, if $t_R \in T$ is critical, then it must be that the positions $R.A, R.B$ in the tuple will have the same value d. Hence, $t_R \in \sigma_{R.B=d}(D)$. $\square$

Finally, we present the proof of Lemma 3.10.

Proof. We will assume that $R.X$ is not fully covered and then show that $\Sigma_{R.X}$ (i.e. views of the form $\sigma_{R.X=a}$) is redundant to $\mathbf{V}$. By Theorem Appendix E.4, it suffices to show that a tuple $t_R \in \Sigma_{R.X}(D)$ is either non-critical or belongs to some view in $(\mathbf{V} \setminus \Sigma_{R.X})(D)$.

For the sake of contradiction, suppose that t_R is critical and also that it does not belong in any view of the set $(\mathbf{V} \setminus \Sigma_{R.X})(D)$. Since $R.X$ is not fully covered, there exists some $c \in Col_{R.X}$ such that $\sigma_{R.X=c} \notin \mathbf{V}$. Now, construct a database $D' = D \cup \{R(t'_R)\}$, where t'_R is as t_R apart from the attribute $R.X$, where its value is c. The new tuple t'_R does not belong in any of the views in $\mathbf{V}(D')$, hence $\mathbf{V}(D') = \mathbf{V}(D)$. Next, since t_R is critical, it contributes to an answer $t \in Q(D)$. When t'_R is added, we will thus get an answer $t' \in Q(D')$, where t is as t' apart from position $R.X$ which has value c. However $t' \notin Q(D)$, a contradiction. □

Appendix F. PTIME Algorithms

Appendix F.1. Reduction to Maximum Flow

In this subsection, we show that the reduction to Min-Cut presented in Subsection 3.1 is valid. First, notice that there exists a one-to-one correspondence of views in Σ to edges with non-infinity capacity in G: for a view $V \in \Sigma$, let $e(V)$ be the corresponding edge, and for an edge e, let $V(e)$ be the corresponding edge.

The first observation we need shows that G admits an $s-t$ cut of finite cost.

Lemma Appendix F.1. *The minimum $s-t$ cut in G has finite cost.*

Proof. By the construction, from the source node s we can reach only a v-node. Moreover, the only way to leave from a v-node is to go through an edge of finite capacity to the corresponding w-node. Hence, any path from s to t has finite capacity. □

Lemma Appendix F.2. *Let $\mathbf{V} \subseteq \Sigma$ s.t. $D \vdash \mathbf{V} \twoheadrightarrow Q$. Then, every $s-t$ path in G is cut by an edge in $C = \{e(V) \mid V \in \mathbf{V}\}$.*

Proof. Suppose not. Then, there exists an uncut path that goes from s to t. First, notice that we can describe a path uniquely by listing the non-infinity edges it crosses, which correspond to selections (since any path visits alternatingly u, v nodes). Moreover, if a path crosses a selection $\sigma_{R_i.X_i=a}$, where R_i is binary, the path will also have to cross some selection $\sigma_{R_i.x_{i+1}=b}$. Hence, any path can be viewed as a sequence of tuples $P = t_1, t_2, \dots, t_\ell$, which may or may not occur in D.

They crucial observation is that we can add/remove the tuples $t_1, \dots, t_\ell$ without modifying the views $\mathbf{V}$, since the selections on these values are not present in $\mathbf{V}$. Consider the two databases $D^+ = D \cup \{t_1, \dots, t_\ell\}$ and $D^- = D - \{t_1, \dots, t_\ell\}$. We have that $\mathbf{V}(D^+) = \mathbf{V}(D^-) = \mathbf{V}(D)$.

Now, consider two consecutive tuples in the path: t_i, t_{i+1} $(i = 1, \ell - 1)$ and let $t_i \in R_j, t_{i+1} \in R_{j'}$, where $j < j'$. Let $a = t_i.x_{j+1}, b = t_{i+1}.x_{j'}$. The path P then goes from the node $w_{R_j.x_{j+1}=a}$ to $v_{R_{j'}.x_{j'}=b}$ through an infinity edge, which implies that $(a, b) \in Md_{[j+1:j'-1]}$. For the first tuple t_1, notice that it is directly connected to s and hence $t_1.X \in Lt_m$, whereas also for t_ℓ, it must be that $t_\ell.Y \in Rt_{m'}$. Hence, D^+ is going to have an extra answer in $Q(D^+)$, since each partial query has a result and the partial queries are connected with the tuples $t_1, \dots, t_\ell$. This extra answer will not be in D^-, a contradiction. □

Lemma Appendix F.3. *If S is a finite cost cut of G, then for $\mathbf{V} = \{V(e) \mid e \in S\}$, $D \vdash \mathbf{V} \twoheadrightarrow Q$.*

Proof. We consider two cases. First, assume that $t \in Q(D)$. Then, consider all the projections of t on the different atoms in Q: $t_{R_0}, \dots, t_{R_{k+1}}$. Let us consider a tuple $t_{R_i} = (a, b)$ w.l.o.g. (it may be that $t_{R_i} = a$). Notice that $a \in Lt_{i-1}$ and $b \in Rt_i$. Hence, by the construction of G, there exists a skip edge from s to the view edge $\sigma_{R_i.x_i=a}$, and a skip edge that connects the view edge $\sigma_{R_i.x_{i+1}=b}$ to t. This implies that there exists an $s-t$ path that crosses only the two selections (or one) for R_i, on a and b. Hence, the tuple t_{R_i} will be covered, which implies that the tuple t will be ensured to be in any D' that agrees with $\mathbf{V}(D)$.

For the other case, let $t \notin Q(D)$. Again, consider the projections of t: $t_{R_0}, \dots, t_{R_{k+1}}$ and keep only the tuples that do not belong in D: $t_1, \dots, t_\ell$. Then, by the construction of G, there exists a path $P = t_1, \dots, t_\ell$. Assume that the path will be cut by S in some selection of the tuple t_i; in this case, t_i will be discovered not to be in the database, which ensures that any database that agrees with $\mathbf{V}(D)$ will not contain t_i in the answer. $\square$

Combining both lemmas, we can now easily derive the proof of Theorem 3.13.

Appendix F.2. An Algorithm for Cyclic Queries

We describe and analyze an algorithm with polynomial data complexity for cyclic queries, i.e. queries of the form

$$C_k(x_0, \dots, x_k) : -R_0(x_0, x_1), \dots, R_k(x_k, x_0)$$

The algorithm reduces the pricing of a cyclic query to Weighted Bipartite Vertex Cover, which is a PTIME problem. We first extend the definition of a *full cover.*

Definition Appendix F.4 (Full Cover). *Let variables x_i, x_j, where $1 \leq i \leq j \leq k$. Then, for $\mathbf{V} \subseteq \Sigma$, we say that the pair (x_i, x_j) is* fully covered *by $\mathbf{V}$ if, for every values $a \in Col_{R_{i-1}.x_i}, b \in Col_{R_j.x_j}$ such that $(a, b) \in Md_{[i:j-1]}$, $\mathbf{V}$ contains $\sigma_{R_{i-1}.x_i=a}$ or $\sigma_{R_j.x_j=b}$.*

If $i = j$, we just say instead that x_i is fully covered by $\mathbf{V}$. Finally, we say that x_0 is fully covered if for every $a \in Col_{x_0}$, $\mathbf{V}$ contains one of $\sigma_{R_0.x_0=a}, \sigma_{R_k.x_0=a}$.

We next show a useful lemma about full covers.

Lemma Appendix F.5. *If $D \vdash \mathbf{V} \twoheadrightarrow C_k$, where $\mathbf{V} \subseteq \Sigma$, at least one variable x_i is fully covered.*

Proof. Suppose not. Then, for every variable x_i, there exists a value $a_i \in Col_{x_i}$ such that $\sigma_{R_{i-1}.x_i=a_i}, \sigma_{R_i.x_i=a_i} \notin \mathbf{V}$ (or for x_0, $\sigma_{R_0.x_0=a}, \sigma_{R_k.x_0=a} \notin \mathbf{V}$). Thus, we can remove or add the tuples $T = \{R_0(a_0, a_1), \dots, R_k(a_k, a_0)\}$ in D without modifying $\mathbf{V}(D)$. Now, consider the databases $D^+ = D \cup T$ and $D^- = D \setminus T$. By our assumption, $\mathbf{V}(D^+) = \mathbf{V}(D^-)$; however, $(a_0, \dots, a_k) \in C_k(D^+)$, whereas $(a_0, \dots, a_k) \notin C_k(D^-)$, a contradiction. $\square$

We can generalize Lemma Appendix F.5 for any full CQ and not only cyclic queries, but this suffices for this algorithm. Notice that Lemma Appendix F.5 guarantees that at least a variable x_i is fully covered. Fix this variable (due to symmetry, we can assume that it is w.l.o.g. x_1) and then fix also a choice of whether any pair of variables (x_i, x_j), $1 \leq i \leq j \leq k$, is fully covered or not. Let us denote this choice by F.

The Graph. We now describe the construction of the weighted bipartite graph $G[F] = (A, B, E)$. Notice that the graph depends on the choice F. In $G[F]$, every node corresponds to a selection from Σ. The left partition A includes the selections that occur left in a relation (i.e. of the form $\sigma_{R_i.x_i=a}$), whereas B the selections that occur to the right (i.e. of the form $\sigma_{R_i.x_{i+1}=a}$). The weight of each node is equal to its price.

As for the edges, we distinguish the following cases:

Start edges: for any $a \in Col_{x_0}$, we add the edge $(\sigma_{R_0.x_0=a}, \sigma_{R_k.x_0=a})$.

Full Cover edges: if the pair (x_i, x_j) is fully covered, for every $(a, b) \in Md_{[i:j-1]}$, we introduce the edge $(\sigma_{R_{i-1}.x_i=a}, \sigma_{R_j.x_j=b})$.

Tuple edges: for each tuple $t = (a_0, \dots, a_k) \in Col_{x_0} \times \cdots \times Col_{x_k}$, consider the sequence of tuples projected at each relation $t_{R_0}, \dots, t_{R_k}$.

1. If $t \in Q(D)$, for any $i = 0, \dots, k$, add the edge $(\sigma_{R_i.x_i=a_i}, \sigma_{R_i.x_{i+1}=a_{i+1}})$.
2. If $t \notin Q(D)$, let $t_{R_{i_1}}, \dots, t_{R_{i_\ell}}$ be the tuple projections that do not belong in D and consider all the pairs of variables $P_t = \{(x_{i_1+1}, x_{i_2}), \dots, (x_{i_{\ell-1}+1}, x_{i_\ell})\}$. Add an edge $(\sigma_{R_{i_1}.x_{i_1}=a_{i_1}}, \sigma_{R_{i_\ell}.x_{i_\ell+1}=a_{i_\ell+1}})$ if none of the pairs in P_t is fully covered.

The Algorithm. We can now describe the full algorithm Cyclic Price.

1. Iterate over all choices of F.
2. For any F, construct the graph $G[F]$ and solve the vertex cover to obtain a set of selections $\mathbf{V}[F] \subseteq \Sigma$ with price $p(\mathbf{V}[F])$.
3. Output $\min_F\{p(\mathbf{V}[F])\}$.

Notice that Cyclic Price has polynomial data complexity, since the choices for F depend only on the size of C_k and bipartite vertex cover is in PTIME. We next show the validity of the algorithm. For this, it suffices to show the following proposition.

Proposition Appendix F.6. *If the optimum solution to pricing C_k satisfies F, the minimum cost of a vertex cover of $G[F]$ equals the price of C_k.*

Proof. Assume a vertex cover C for $G[F]$ and for any vertex v, let $\sigma(v)$ be the corresponding selection. Moreover, denote $\mathbf{V} = \bigodot_{v \in C} \sigma(v)$. We show that $D \vdash \mathbf{V} \twoheadrightarrow C_k$. Consider a tuple $t = (a_0, \dots, a_k) \in C_k(D)$. Then, by the construction, every edge of the form (a_i, a_{i+1}) will be covered by $\mathbf{V}$. Let $t \notin C_k(D)$. Then, consider all the tuple projections to atoms of Q s.t. the tuples do not belong in D: $t_{R_{i_1}}, \dots, t_{R_{i_\ell}}$. If some of the pair of variables $(x_{i_1+1}, x_{i_2}), \dots, (x_{i_{\ell-1}+1}, x_{i_\ell})$ is fully covered, then some tuple will be covered by $\mathbf{V}$. Else, there exists an edge $(\sigma_{R_{i_1}.x_{i_1}=a_{i_1}}, \sigma_{R_{i_\ell}.x_{i_\ell+1}=a_{i_\ell+1}})$, which means that a tuple will again be covered. We can apply Lemma Appendix E.1 to conclude the one direction.

For the inverse direction, consider any $\mathbf{V} \subseteq \Sigma$ such that $D \vdash \mathbf{V} \twoheadrightarrow C_k$ and $\mathbf{V}$ satisfies F. Suppose that the corresponding set of vertices C is not a vertex cover for $G[F]$. Then, there exists an edge e that is not covered. Notice that, by our assumption for $\mathbf{V}$ satisfying F, this edge can not be one of the full cover edges or start edges. Also, it can not be one of the tuple edges of a tuple that belongs in an answer, since by Lemma Appendix E.1 this edge will be covered.

Hence, it must be an tuple-edge introduced in step 2. This implies that there are tuples $t_{R_{i_1}}, \dots, t_{R_{i_\ell}}$ that do not appear in D. Consider two consecutive tuples in the sequence, $t_{R_{i_m}} = (a_{i_m}, a_{i_m}+1)$, $t_{R_{i_{m+1}}} = (a_{i_{m+1}}, a_{i_{m+1}}+1)$. Notice that there exists $(a_{i_m}+1, a_{i_{m+1}}) \in Md_{[i_m:i_{m+1}]}$, but by our construction $(x_{i_m}+1, x_{i_{m+1}})$ is not fully covered. This implies that there exists some $(a'_{i_m+1}, a'_{i_{m+1}}) \in Md_{[i_m:i_{m+1}]}$such that $\sigma_{R_{i_m}.x_{i_m+1}=a'_{i_m+1}}, \sigma_{R_{i_{m+1}}.x_{i_{m+1}}=a'_{i_{m+1}}} \notin \mathbf{V}$. We can apply this for any two consecutive tuples, hence we have a value a'_{i_m} for each position of the tuples such that the corresponding selections are not in $\mathbf{V}$. We can then create databases D^-, D^+ that remove/add to D the tuples that can be created by these constants. The views $\mathbf{V}$ remain the same, but $C_k(D^+), C_k(D-)$ have different answers now, a contradiction. □

Appendix G. Proof the Dichotomy Theorem

In this section, we complete the proof of the dichotomy theorem. For the first part of the analysis, we will focus on full Conjunctive queries w/o self-joins. We will show in the end how to deal with projections.

We will denote $\textsc{Price}(Q_1) \prec \textsc{Price}(Q_2)$ if we can reduce $\textsc{Price}(Q_1)$ to $\textsc{Price}(Q_2)$ in polynomial time. We write $\textsc{Price}(Q_1) \sim \textsc{Price}(Q_2)$ if there is a PTIME reduction in both directions.

The proof consists of several steps:

1. We first show that it suffices to look at one connected component (Proposition Appendix G.1).
2. We next show that $\sim$ holds if we remove constants (Proposition Appendix G.3), multiple occurrences of a variable in one atom (Proposition Appendix G.4), and hanging variables (Proposition Appendix G.6): we call this class of queries *normalized queries.*
3. Using Proposition Appendix D.1, we show that pricing a normalized query is NP-hard if it contains a ternary predicate.
4. Finally, using Proposition Appendix D.5, we show that if a query is not cyclic or GChQ, it is NP-hard.

We start by examining queries with more than one connected component. The following proposition establishes that for a query Q with connected components Q_1, Q_2, pricing Q is in PTIME if and only if pricing both Q_1, Q_2 is also in PTIME. Notice that the proposition holds not only for full CQs, but even for CQs with projections.

Proposition Appendix G.1. *Assume that Q can be partitioned such that $Q(\bar{x}_1, \bar{x}_2) = Q'_1(\bar{x}'_1), Q'_2(\bar{x}'_2)$, where $\bar{x}'_1, \bar{x}'_2$ are disjoint sets of variables, $\bar{x}_1 \subseteq \bar{x}'_1$ and $\bar{x}_2 \subseteq \bar{x}'_2$. Let $Q_1(\bar{x}_1) = Q'_1(\bar{x}'_1)$ and $Q_2(\bar{x}_2) = Q'_2(\bar{x}'_2)$. Then,* $\textsc{Price}(Q_i) \prec \textsc{Price}(Q)$ *and* $\textsc{Price}(Q) \prec \textsc{Price}(Q_1, Q_2)$.

Proof. In order to show that $\textsc{Price}(Q) \prec \textsc{Price}(Q_1, Q_2)$, we prove the equation in Proposition 3.14. Assume a database D and a set of price points $\mathcal{S}$.

First, consider the case where $Q_i(D) = \emptyset$ for some $i = 1, 2$. Let $\mathbf{V}_i \subseteq \Sigma$ such that $D \vdash \mathbf{V}_i \twoheadrightarrow Q_i$. Notice that $\mathbf{V}_i$ also determines Q, since $Q_i(D)$ being empty implies that $Q(D) = \emptyset$ as well. Hence, $D \vdash \mathbf{V}_i \twoheadrightarrow Q$. This implies that, in the case that $Q_i(D) = \emptyset$, $p_D^{\mathcal{S}}(Q) \leq p_D^{\mathcal{S}}(Q_i)$.

Next, consider the case where $Q_1(D) \neq \emptyset$. We will show that, if $D \vdash \mathbf{V} \twoheadrightarrow Q$, then $D \vdash \mathbf{V} \twoheadrightarrow Q_2$. This implies that $p_D^{\mathcal{S}}(Q) \geq p_D^{\mathcal{S}}(Q_2)$. Indeed, since $\mathbf{V}$ determines Q_2, we can decide $Q(D)$. Since $Q_1(D) \neq \emptyset$, however, $Q_2(D) = \pi_{\bar{x}_2}(Q(D))$. This holds symmetrically when $Q_2(D) \neq \emptyset$.

Combining the two inequalities, we can show case 2 (and symmetrically case 3) of the equation. In order to prove case 1 (where both Q_1, Q_2 are empty), assume that we have some $\mathbf{V}$ that does not determine neither Q_1 or Q_2. Then, consider the tuples t_1, t_2 that witness the non-determinacy for Q_1, Q_2 for databases D_1', D_2' which are of minimum size. The tuple $t_1 \circ t_2$ will then be a witness for not being able to determine Q, for the database $D_{1'} \cup D_{2'}$. This implies that $p_D^{\mathcal{S}}(Q) \geq \min\{p_D^{\mathcal{S}}(Q_1), p_D^{\mathcal{S}}(Q_2)\}$ and concludes case 1.

As for case 4, where both Q_1, Q_2 are non-empty, notice that $D \vdash \mathbf{V} \twoheadrightarrow Q$ implies that for every $i = 1, 2$, $D \vdash \mathbf{V} \twoheadrightarrow Q_i$. Let $\mathbf{V}^i$ be the selections from $\mathbf{V}$ that are only on atoms from Q_i. Then, it easy to see that $D \vdash \mathbf{V}^i \twoheadrightarrow Q_i$ as well. Since $\mathbf{V}^1 \cap \mathbf{V}^2 = \emptyset$ and $\mathbf{V}^1, \mathbf{V}^2 = \mathbf{V}$, we have that $p_D^{\mathcal{S}}(Q) \geq p_D^{\mathcal{S}}(Q_1) + p_D^{\mathcal{S}}(Q_2)$. The lower bound is trivial, so this concludes case 4.

We finally show that $\textsc{Price}(Q_i) \prec \textsc{Price}(Q)$ (w.l.o.g. let $i = 1$). Assume that we want to compute the price of Q_1 for some database D_1 under price points $\mathcal{S}$. We construct a database $D = D_1 \cup D_2$, where D_2 is a fixed database is such that $Q_2(D_2) \neq \emptyset$ (such a D_2 always exists, since Q_2 is not trivially empty). Moreover, let us price for $\mathcal{S}'$ every selection on the atoms of Q_2 (the set Σ^2) to 0 and every other selection (in Σ^1) the same. An immediate observation is that $\forall t_2 \in Q_2(D_2), t \circ t_2 \in Q(D)$ iff $t \in Q(D_1)$. This gives us a map between the query answers. Notice also that for $\mathbf{V} \subseteq \Sigma^1$, it is immediate to obtain $\mathbf{V}(D), \Sigma^2(D)$ from $\mathbf{V}(D_1)$ and vice versa, since $\mathbf{V}(D) = \mathbf{V}(D_1)$ and $\Sigma^2(D)$ is fixed. Hence, $D_1 \vdash \mathbf{V} \twoheadrightarrow Q_1$ if and only if $D \vdash \mathbf{V}, \Sigma^2 \twoheadrightarrow Q$; this implies that $p_{D_1}^{\mathcal{S}}(Q_1) = p_D^{\mathcal{S}'}(Q)$. □

We next show how to deal with constants (and constraints more general). The first proposition holds for any atomic constraint.

Proposition Appendix G.2. *Suppose Q has a variable x with an atomic predicate $C(x)$. Let Q' be the query obtained by deleting $C(x)$ from Q. Then,* $\textsc{Price}(Q) \prec \textsc{Price}(Q')$.

Proof. The idea is to shrink the column of x to $Col_x' = \{a \in Col_x \mid C(a) = true\}$ for Q'. Let $D' \subseteq D$ be the database obtained by filtering all the tuples of D with the predicate C. Finally, let $\mathcal{S}'$ be the corresponding price points for D', where the prices remain the same. We show that $p_D^{\mathcal{S}}(Q) = p_{D'}^{\mathcal{S}'}(Q')$. The key observation is that if for some $\mathbf{V} \subseteq \Sigma$, $D \vdash \mathbf{V} \twoheadrightarrow Q$, then we can assume that no selection on $x = a, \neg C(a)$ appears in $\mathbf{V}$ (this follows from the fact that any tuple in such a selection would be non-critical and Theorem Appendix E.4). It is now easy to see that, under this assumption, $D \vdash \mathbf{V} \twoheadrightarrow Q$ iff $D' \vdash \mathbf{V} \twoheadrightarrow Q'$. Indeed, $Q(D) = Q'(D')$. Moreover, if $D' \vdash \mathbf{V} \twoheadrightarrow Q'$, then we can take $\mathbf{V}(D)$, compute $\mathbf{V}(D')$ by filtering the tuples with C, then use determinacy to compute $Q'(D') = Q(D)$. For the converse, if $D \vdash \mathbf{V} \twoheadrightarrow Q$, then it follows that $D - T^C \vdash \mathbf{V} \twoheadrightarrow Q$, where T^C are the tuples of D filtered by C (from Proposition 2.22). Now, we can take $\mathbf{V}(D') = \mathbf{V}(D - T^C)$, and from that compute $Q(D - T^C) = Q(D) = Q'(D')$. □

Proposition Appendix G.3. *Suppose Q has a constant a in some atom R. Let Q' be the query where R is replaced with R' s.t. a is removed. Then,* $\textsc{Price}(Q) \sim \textsc{Price}(Q')$.

Proof. We first show that $\textsc{Price}(Q) \prec \textsc{Price}(Q')$. Indeed, we can rewrite Q by introducing a new variable x_a to replace the constant a and add the constraint $x_a = a$. By Proposition Appendix G.2, we can now remove the constraint $x_a = a$. Then, notice that x_a is a hanging variable, so we can remove it as well by Proposition Appendix G.6 to obtain Q'. To prove that $\textsc{Price}(Q') \prec \textsc{Price}(Q)$, we can apply the same proof as in the second part of Proposition Appendix G.6 to reduce Q' to Q'', where we have added to R' an extra attribute x_a with column $Col_{x_a} = \{a\}$. But know adding to Q'' the constraint $x_a = a$ gives an equivalent query, which is Q. □

We next show that it suffices to characterize the complexity of a query by looking at the one that occurs after having removed multiple occurrences of the same variable in the same atom of Q.

Proposition Appendix G.4. *Let Q contain an atom R where a variable x appears more than once and let Q' be the query obtained if we replace R by R', where we keep only one occurrence of x. Then,* $\textsc{Price}(Q) \sim \textsc{Price}(Q')$.

Proof. For ease of exposition, let us assume w.l.o.g. that R appears in Q as $R(x, x, \dots)$, where R has schema $R(A, B, \dots)$. Let R' be the corresponding relation in Q', with schema $R(A, \dots)$.

We first show that $\textsc{Price}(Q') \prec \textsc{Price}(Q)$. Suppose we want to compute the price of Q' for a database D' and prices $\mathcal{S}$. Then, we create a database D for Q such that for $S \neq R : S^D = S^{D'}$ and also replace $R'^{D'}$ with R^D such that $(a, \dots) \in R'^{D'}$ iff $(a, a, \dots) \in R^D$. It is easy to observe that $Q'(D') = Q(D)$. Let the prices in $\mathcal{S}'$ be $p(\sigma_{R.A=a}) = p(\sigma_{R.B=a}) = p(\sigma_{R'.x=a})$ for any $a \in Col_x$ and keep the same price for any other selection. It is easy to see now that $p_D^{\mathcal{S}}(Q) = p_{D'}^{\mathcal{S}'}(Q')$ (since we have a 1-1 mapping between views and query answers).

We next show that $\textsc{Price}(Q) \prec \textsc{Price}(Q')$. Given a database D, Q and $\mathcal{S}$, we construct a new database D' where we replace only R^D with $R'^{D'}$ such that $(a, \dots) \in R'^{D'}$ iff $(a, a, \dots) \in R^D$. Moreover, for $\mathcal{S}'$ we let $p(\sigma_{R'.x=a}) = \min\{p(\sigma_{R.A=a}), p(\sigma_{R.B=a})\}$ for any $a \in Col_x$ and keep the other prices the same. Again, it is easy to observe that $Q(D) = Q'(D')$. Now, consider the minimum priced $\mathbf{V} \subseteq \Sigma$ such that $D \vdash \mathbf{V} \twoheadrightarrow Q$. By Lemma Appendix E.7, $\mathbf{V}$ can not contain the maximum priced selection out of $\sigma_{R.A=a}, \sigma_{R.B=a}$, for any $a \in Col_x$. Hence, by replacing the minimum priced $\sigma_{R.A=a}$ with $\sigma_{R'.x=a}$ (w.l.o.g.), we can obtain an equal cost set $\mathbf{V}'$. We next show that $D' \vdash \mathbf{V}' \twoheadrightarrow Q'$. Indeed, if T are the tuples from R^D such that $t.A \neq t.B$, we have from Lemma Appendix E.3 that $D - T \vdash \mathbf{V} \twoheadrightarrow Q$. Now, from $\mathbf{V}'(D')$ we can compute $\mathbf{V}(D - T)$, which by the determinacy assumption gives $Q(D - T) = Q(D) = Q'(D')$. For the converse direction, let $D' \vdash \mathbf{V}' \twoheadrightarrow Q'$. We will show that $D \vdash \mathbf{V}' \twoheadrightarrow Q$. Indeed, if we know $\mathbf{V}'(D)$, we can compute $\mathbf{V}(D')$, and then by the determinacy assumption $Q(D') = Q(D)$. □

We next deal with hanging variables. Let Q be a query with a hanging variable $R.x$, where $R(X_1, \dots, X_k, X)$. For a tuple $t = (a_1, \dots, a_k, a) \in R$, define for $b \in Col_{R.x}$, $t(b) = (a_1, \dots, a_k, b)$. Also, let $M(t) = \{t(b) \mid b \in Col_{R.x}\}$.

Lemma Appendix G.5. *Let Q be a query with a hanging variable $R.x$, where $R(X_1, \dots, X_k, X)$. Given a database D, define $D_C = D \bigcup_{t \in R^D} M(t)$. Then, for any price points $\mathcal{S}$, $p_D^{\mathcal{S}}(Q) = p_{D_C}^{\mathcal{S}}(Q)$.*

Proof. It suffices to show that $D \vdash \mathbf{V} \twoheadrightarrow Q$ iff $D_C \vdash \mathbf{V} \twoheadrightarrow Q$.

The one direction comes from Proposition 2.22, since $D \subseteq D_C$. For the other direction, let $t \in D$ and $a \in Col_{R.x}$. Let $D^a = D \cup R(t(a))$. We will show that $D^a \vdash \mathbf{V} \twoheadrightarrow Q$; then, the lemma follows by induction. Indeed, notice that $(D^a - t) \vdash \mathbf{V} \twoheadrightarrow Q$ by genericity and since x is hanging. Moreover, $Q(D^a) = Q(D) \cup Q(D^a - t)$. Hence, having $\mathbf{V}(D^a)$, we can compute both $\mathbf{V}(D), \mathbf{V}(D^a - t)$, then compute the answers and union the results to obtain $Q(D^a)$. □

Proposition Appendix G.6. *Let Q^H be the query obtained from Q if we remove all the hanging variables. Then,* $\textsc{Price}(Q^H) \sim \textsc{Price}(Q)$.

Proof. We will show this for the case of one hanging variable; then, we can obtain the proposition using induction. Let us assume that the hanging variable is of the form $R.X$ and let Q^{-x} be the query obtained by removing variable x from the atom $R(X_1, \dots, X_k, X)$, obtaining the new atom is $R'(X_1, \dots, X_k)$. Notice that R also contains at least one other variable (since we can assume w.l.o.g. one connected component).

We first show that $\textsc{Price}(Q) \prec \textsc{Price}(Q^{-x})$. Let D be a database and $\mathcal{S}$ price points for Q. From Lemma Appendix G.5, we can instead consider the database D_C with the same price points.

Let D' the database for Q^{-x}, where we have replaced R^{D_C} by $R'^{D'}$, such that $R'^{D'}$ contains the tuples from R^{D_C} where x has been projected out. A key property is that if the head variables are $(x, x_1, \dots)$, then $(a_1, \dots) \in Q^{-R.x}(D')$ iff $\forall a \in Col_{R.x}$: $(a, a_1, \dots) \in Q(D_C)$. We now distinguish two cases for the prices: $\mathcal{S}'$ prices every selection in R' to zero, and every other selection to the same price. $\mathcal{S}''$ keeps all the prices the same. We show that

$$p_{D_C}^{\mathcal{S}}(Q) = \min\{p_{D'}^{\mathcal{S}'}(Q^{-x}) + p(\Sigma_{R.X}), p_{D'}^{\mathcal{S}''}(Q^{-x})\}$$

Note that by Lemma 3.10, if $D \vdash \mathbf{V} \twoheadrightarrow Q$, we can consider only two cases: either $\mathbf{V}$ contains every view in the set $\Sigma_{R.X} = \{\sigma_{R.X=a} \mid a \in Col_x\}$ or none of them.

We first show that, if $\Sigma_{R.X} \subseteq \mathbf{V}$, then $D_C \vdash \mathbf{V} \twoheadrightarrow Q$ iff $D' \vdash R, \mathbf{V} \setminus \Sigma_{R.X} \twoheadrightarrow Q^{-x}$ (notice that R is free). Indeed, we have a function to go from $Q(D_C)$ to $Q^{-R.x}(D)$ and vice versa. Moreover, we also have a function to map $\mathbf{V}(D_C)$ to $R(D'), (\mathbf{V} \setminus \Sigma_{R.X})(D')$ and vice versa.

If $\Sigma_{R.X} \not\subseteq \mathbf{V}$, we can assume that $\mathbf{V}$ contains no views from $\Sigma_{R.X}$ at all and show that $D_C \vdash \mathbf{V} \twoheadrightarrow Q$ iff $D' \vdash \mathbf{V} \twoheadrightarrow Q^{-x}$. Again, we have a direct mapping from $\mathbf{V}(D')$ to $\mathbf{V}(D_C)$ and vice versa and also the mapping for the query answers.

For the other direction, we want to show that $\textsc{Price}(Q^{-x}) \prec \textsc{Price}(Q)$. Suppose some $D', \mathcal{S}'$ for Q^{-x}. Let $Dom(R.X) = \{a\}$. We create a new database D where R^D is populated as follows: $t = (a_1, \dots, a_k, a) \in R^D$ iff $(a_1, \dots, a_k) \in R'^{D'}$. The rest of the relations remain exactly as in D'.

We now observe that, by construction, $(a_1, \dots, a_l) \in Q^{-x}(D)$ iff $(a_1, \dots, a_l, a) \in Q(D)$. We define the price points $\mathcal{S}$ by pricing the single selection $\sigma_{R.x=a}$ at the cost of the table R and keeping all the other prices the same. This implies that we can assume w.l.o.g. that the optimal pricing $p_D^{\mathcal{S}}$ will never choose $\sigma_{R.x=a}$, since we could just replace it by asking for the individual selections of the cheapest attribute of R. Hence, any selection set for Q corresponds to a selection set for Q^{-x}. Moreover, we have a one-to-one mapping of the answers of Q, Q^{-x} and a one-to-one mapping for $\mathbf{V}(D), \mathbf{V}(D')$. Thus, $p_D^{\mathcal{S}}(Q) = p_{D'}^{\mathcal{S}'}(Q^{-x})$. $\square$

We next present a very useful lemma for hardness reductions. The lemma states that, given a hard query, we can add any variable in any atom and the query still remains hard. The lemma holds also for CQs with projections. Given a query Q that contains an atom R with a variable x, we denote by $Q^{-R.x}$ the query obtained by removing x from R (and if R has a single attribute, remove R).

Lemma Appendix G.7. Price($Q^{-R.x}$) $\prec$ Price(Q).

Proof. Without loss of generality, we can assume that x is not hanging (by Proposition Appendix G.6).

Consider pricing the query $Q^{-R.x}$ under some database D and price points $\mathcal{S}$. We will show how to compute this query by reducing it to a computation of Q for $\mathcal{S}', D'$. Assume that Q is obtained from $Q^{-R.x}$ by replacing $R(X_1, \dots, X_k)$ with $R'(X_1, \dots, X_k, Y)$ (we add Y at the end w.l.o.g.). The column of Y is $Col_Y = Col_x$. In order to create D' from D, we replace only R^D with a new relation $R'^{D'}$ such that $\forall b \in Col_Y : (a_1, \dots, a_k, b) \in R'^{D'}$ if and only if $(a_1, \dots, a_k) \in R^D$. If R does not exist in $Q^{-R.x}$, we just add $R'^{D'} = Col_x$.

We now claim that $Q^{-R.x}(D) = Q(D')$. Indeed, consider a tuple $t \in Q(D')$ and any tuple from $R'^{D'}$ that contributes to t, let it be $t_{R'} = (a_1, \dots, a_k, b)$. Then, $(a_1, \dots, a_k) \in R^D$. Since the other tables are unchanged, $t \in Q^{-R.x}(D)$. For the opposite direction, let $t \in Q^{-R.x}(D)$ and a tuple $t_R = (a_1, \dots, a_k)$ from R^D that contributes to t. Moreover, for the assignment of values to variables that results in t_R contributing to t, let b be the value of the attribute x at tuple t (since x appears somewhere in $Q^{-R.x}$). Then, by our construction, $(a_1, \dots, a_k, b) \in R'^{D'}$. Again, since the other tables remain unchanged, $t \in Q(D')$.

In order to compute $\mathcal{S}'$, we let the price of any selection on $R.x$ to be equal to the price of R and all the other prices remain the same as in $\mathcal{S}$. Hence, we can assume w.l.o.g. that no selection from $R.x$ will be chosen for any $\mathbf{V}$ that determines Q. In order to prove that $p_{D'}^{\mathcal{S}'}(Q) = p_D^{\mathcal{S}}(Q^{-R.x})$, it suffices to show that $D \vdash \mathbf{V} \twoheadrightarrow Q^{-R.x}$ iff $D' \vdash \mathbf{V} \twoheadrightarrow Q$. However, this follows from the fact that we can compute $\mathbf{V}(D)$ from $\mathbf{V}(D')$ and vice versa, as well as the fact that $Q^{-R.x}(D) = Q(D')$. $\square$

We also need another useful reduction.

Lemma Appendix G.8. *Let $Q(\bar{z}) = Q_1(x, \bar{x}'), R(x, y), Q_2(y, \bar{y}')$ be a CQ. Moreover, let $Q'(\bar{z}') = Q_1(x, \bar{x}'), Q_2(x, \bar{y}')$, where $\bar{z}'$ is obtained from $\bar{z}$ by replacing y with x. Then,* Price(Q') $\prec$ Price(Q).

Proof. Assume a database D' for Q', along with a set of price points $\mathcal{S}'$. In order to prove the reduction, we construct a new database D where we have added to D' the relation $R^D = \{(a, a) \mid a \in Col_x\}$. It is easy to see that this construction allows a one-to-one mapping from tuples in $Q(D)$ to $Q'(D')$ and vice versa. More precisely, y does not appear in $\bar{z}$, then $Q(D) = Q'(D')$. Otherwise, we distinguish two cases. First, Q

has head variables $(x, y, \dots)$ and Q' has $(x, \dots)$. Then, $(a, \dots) \in Q'(D')$ if and only if $(a, a, \dots) \in Q(D)$. Second, Q has head variables $(y, \dots)$ and thus Q' has $(x, \dots)$. In this case, we again have that $Q(D) = Q'(D')$.

As for the set of price points $\mathcal{S}$, we keep the prices for the other relations the same as in $\mathcal{S}'$, and price every selection from R to zero. In order to show that $p_D^{\mathcal{S}}(Q) = p_{D'}^{\mathcal{S}'}(Q')$, it suffices to prove that $D \vdash \mathbf{V}, \Sigma_{R.x} \twoheadrightarrow Q$ iff $D' \vdash \mathbf{V} \twoheadrightarrow Q'$. However, there is a direct mapping from $\mathbf{V}(D), \Sigma_{R.x}(D)$ to $\mathbf{V}(D')$ (since R is constant and the other views are exactly the same) and vice versa. This, together with the mapping of the answer concludes the proof. □

At this point, it suffices to characterize the complexity for pricing a *normalized* query to conclude the proof of the dichotomy.

Proposition Appendix G.9. *If a normalized query Q contains an atom with ≥ 3 attributes,* Price(Q) *is NP-complete.*

Proof. Assume that Q contains an atom $R(x, y, z, \dots)$. Notice that x, y, z will be different. Applying repeatedly Lemma Appendix G.7, we can obtain a query Q' that contains only the variables x, y, z. Since Q is normalized, Q' will be normalized as well and thus x, y, z appear in other atoms in Q'. Applying again Lemma Appendix G.7, let Q'' be the query obtained from Q by keeping exactly two occurrences of each x, y, z: one in R and the other in R_x, R_y, R_z respectively.

We now distinguish several cases: (a) R_x, R_y, R_z are different. Then, $Q''(x, y, z) = R(x, y, z), R_x(x), R_y(y), R_z$. (b) $R_x = R_y \neq R_z$ (and the symmetric cases). Then, $Q''(x, y, z) = R(x, y, z), R_x(x, y), R_z(z)$ and (c) $R_x = R_y = R_z$. Then, $Q''(x, y, z) = R(x, y, z), R_x(x, y, z)$. For all of these cases, Price(Q'') is NP- complete by applying Proposition Appendix D.1 and its more general proof. □

Hence, it now suffices to characterize normalized queries where every atom has at most 2 attributes: we call these *2-normalized* queries.

Proposition Appendix G.10. *Any query of the form $C_k^+(x_1, x_k) = S_{i_1}(x_{i_1}), \dots, S_{i_\ell}(x_{i_\ell}), C_k(x_1, \dots, x_k)$ is NP-complete.*

Proof. First, we can apply Lemma Appendix G.7 to keep only one unary predicate, let it be $S(x_1)$. The query is then $Q'(\bar{x}) = S(x_1), R_1(x_1, x_2), \dots, R_k(x_k, x_1)$. We can then apply repeatedly Lemma Appendix G.8 to remove the relations $R_3, \dots, R_k$. More precisely, one application of Lemma Appendix G.8 will give the query $Q'' = S(x_1), R_1(x_1, x_2), \dots, R_{k-1}(x_{k-1}, x_1)$, and so on. In the end, we obtain the query $Q^0 = S(x_1), R_1(x_1, x_2), R_2(x_2, x_1)$, which is the NP-complete query H_2 (Proposition Appendix D.2). □

Proposition Appendix G.11. *Consider a 2-normalized query Q where a variable x appears in ≥ 3 binary predicates. Then,* Price(Q) *is NP-complete.*

Proof. Let the three binary predicates be $R_1(x, y_1), R_2(x, y_2), R_3(x, y_3)$ and consider the query Q' that we obtain by removing all the other variables apart from x, y_i and keeping these 3 occurrences of x along with a second occurrence of each y_i (Lemma Appendix G.7 justifies this). We now distinguish several cases.

If any of them are equal, let it be $y_1 = y_2$, then the query $Q''(x,y) = R_1(x,y_1), R_2(x,y_1), R_3(x)$ that occurs from Q' is NP-complete by Proposition Appendix D.2. If they are all different, let S_1, S_2, S_3 the relations they appear in Q'. If all S_i are different, then $Q'(x,y_1,y_2,y_3) = S_1(y_1), R_1(x,y_1), S_2(y_2), R_2(x,y_2), S_3(y_3), R_3(x,y_3)$, which is NP-complete by Proposition Appendix D.5. Finally, if two of S_i are equal (it cannot be 3, since the relations are at most binary), let it be that $S_1 = S_2$, we can obtain a query $Q''(x,y_1,y_2) = S_1(y_1,y_2), R_1(x,y_1), R_2(x,y_2), R_3(x)$, which is NP-complete by Proposition Appendix G.10. □

Finally, we complete the dichotomy by a syntactic characterization.

Proposition Appendix G.12. *Let Q be a 2-normalized query. Then, Q either:*

1. *is C_k or C_k^+*
2. *has a variable x that appears in ≥ 3 predicates.*
3. *is a* GChQ *query.*

Proof. Suppose that every variable of Q appears in at most 2 binary predicates. We distinguish two cases. In the first case, every variable x_i belongs in exactly 2 binary predicates. It is easy to see that Q is the cycle C_k plus some (or none) unary predicates, i.e. it is C_k or C_k^+.

Otherwise, there exists a variable x_1 that belongs in exactly one binary predicate (since Q is normalized, it can never be zero). Order the binary relations starting from the one that contains x_1 and following the joining variables. Notice that in the end of the process we will have seen all variables and all binary relations. Since all the other relations are unary, Q will be a GChQ. □

Projections. Finally, we show how to extend our result for CQs with projections.

Lemma Appendix G.13. *If Q is a boolean query,* Price(Q) $\sim$ Price(Q^f)*, where Q^f is the corresponding full query of Q.*

Proof. Let D be a database for which we price Q. If $Q(D)$ is false, then pricing Q is equivalent to pricing Q^f, because Q is false iff Q^f is empty. If $Q(D)$ is true, then to determine this we need to find the cheapest witness for $Q(D)$ being true. This can be done in PTIME though: compute $Q^f(D)$, then compute the price of each tuple in the answer (for each relation, compute the minimum priced selection) and find the minimum priced tuple.

Now, assume that Price(Q^f) is NP-hard. The crucial observation is that all the hardness reductions use instances D where the query result is empty. Hence, Price(Q) is NP-hard as well. If Price(Q^f) is in PTIME, then we distinguish two cases, depending on whether the query is true or false. By our previous observation, both cases are in PTIME, hence Price(Q) is also in PTIME. □

Lemma Appendix G.13, along with Proposition Appendix D.4 imply the following dichotomy for CQ's with projections.

Theorem Appendix G.14. *For a conjunctive query Q without self-joins and one connected component, let Q^f be the corresponding full CQ. Then:*

- *If Q is full or boolean,* Price(Q) $\sim$ Price(Q^f).

- *Else,* Price(Q) *is NP-complete.*

Proof. The first part follows from the dichotomy theorem for full CQs and Lemma Appendix G.13. For the second part, assume a query $Q(\bar{x}) = Q_1(\bar{x}, \bar{y})$, where $\bar{x}, \bar{y}$ are not empty (since it is not full nor boolean). Consider any two variables $x_0 \in \bar{x}$ and $y_0 \in \bar{y}$. Notice that both x_0, y_0 must exist in some atom with at least two attributes, since Q has one connected component. Let R, S such atoms for x_0, y_0 respectively. Moreover, since Q is one connected component, there will be w.l.o.g. a sequence of atoms $R_1, \ldots, R_\ell$ such that any two consecutive atoms R_i, R_{i+1} share a variable x_i, R and R_1 share a variable x_1 and R_ℓ and S share a variable x_ℓ. Applying Lemma Appendix G.7, we can remove any other variable from Q to obtain a query $Q'(\bar{x}') = R(x_0, x_1), R_1(x_1, x_2), \ldots, R_\ell(x_\ell, y_0)$, where $\bar{x}' \subseteq \bar{x}$, but still containing x_0. Finally, we can repeatedly apply Lemma Appendix G.8 to obtain the query $Q''(x_0) = R(x_0, y_0)$, which is the NP-complete query H_4 (Proposition Appendix D.4). $\square$