Query-Based Data Pricing

PARASCHOS KOUTRIS, PRASANG UPADHYAYA, MAGDALENA BALAZINSKA, BILL HOWE, and DAN SUCIU, University of Washington

Data is increasingly being bought and sold online, and Web-based marketplace services have emerged to facilitate these activities. However, current mechanisms for pricing data are very simple: buyers can choose only from a set of explicit views, each with a specific price. In this article, we propose a framework for pricing data on the Internet that, given the price of a few views, allows the price of any query to be derived automatically. We call this capability *query-based pricing*. We first identify two important properties that the pricing function must satisfy, the *arbitrage-free* and *discount-free* properties. Then, we prove that there exists a unique function that satisfies these properties and extends the seller's explicit prices to all queries. Central to our framework is the notion of query determinacy, and in particular *instance-based determinacy*: we present several results regarding the complexity and properties of it.

When both the views and the query are unions of conjunctive queries or conjunctive queries, we show that the complexity of computing the price is high. To ensure tractability, we restrict the explicit prices to be defined only on selection views (which is the common practice today). We give algorithms with polynomial time data complexity for computing the price of two classes of queries: chain queries (by reducing the problem to network flow), and cyclic queries. Furthermore, we completely characterize the class of conjunctive queries without self-joins that have PTIME data complexity, and prove that pricing all other queries is NP-complete, thus establishing a dichotomy on the complexity of the pricing problem when all views are selection queries.

Categories and Subject Descriptors: H.2.4 [**Database Management**]: Systems—*Relational databases*

General Terms: Algorithms, Economics, Theory

Additional Key Words and Phrases: Data pricing, arbitrage, query determinacy

ACM Reference Format:
Paraschos Koutris, Prasang Upadhyaya, Magdalena Balazinska, Bill Howe, and Dan Suciu. 2015. Query-based data pricing. J. ACM 62, 5, Article 43 (October 2015), 44 pages.
DOI: http://dx.doi.org/10.1145/2770870

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 (xignite.com) sells financial data, Gnip (gnip.com) provides data from social media, PatientsLikeMe (patientslikeme.com) sells anonymized, self-reported, patient statistics to pharmaceutical companies, and AggData (aggdata.com) aggregates various types of data available on the Web. To support and facilitate this online *data market*, Web-based marketplace services have recently emerged: The Windows Azure Marketplace (datamarket.azure.com) offers over 100 data sources from 42 publishers in 16 categories, and the Infochimps Data Marketplace (infochimps.com/marketplace) offers over 10,000 data sets from multiple vendors.

This work is supported in part by the NSF and Microsoft through NSF grant CCF-1047815 and also grant IIS-0915054.
Author's addresses: P. Koutris, email: pkoutris@cs.washington.edu.

0004-5411/2015/10-ART43 $15.00
DOI: http://dx.doi.org/10.1145/2770870

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 article, 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 (customlists.net) sells the American Business Database for \$399; a customer can also buy the subset of companies that have an email 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 this challenge, in this article 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[1]. 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* [Shapiro and Varian 1998] (reviewed in Section 7), 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 versions, and, as a consequence, appeals to large variety of buyers with a large variety of needs. Buyers with different needs are able to choose which queries to buy, based on the value that they associate to the data.

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 nonnegative 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 $p_1 + \cdots + p_{50} \geq p$. 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 $p \leq p_1 + \cdots + p_n$.

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. In general, there may be several arbitrage-free pricing functions that agree with the

[1]Throughout the article, we will follow the convention of calling the queries for which the seller sets explicit prices *views* and the queries the buyer asks *queries*.

prices that are explicitly set by the seller; the discount-free axiom requires that the pricing function should also be a *maximal* one.

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 between the views explicitly priced by the seller, 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 (2). This formula immediately gives 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), the decision version of the problem of computing the price of certain conjunctive queries is NP-hard in the size of the input database. However, we show that for a large class of full conjunctive queries (which include cyclic and chain queries), computing the price can be done efficiently (PTIME) when all explicit price points are selection queries. This class 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.

In order to show the tractability for this class of queries, we provide two algorithms. The first algorithm prices *chain queries* and is based on a nontrivial reduction to the Min-Cut problem in weighted graphs, which is the dual of the Max-Flow problem [Cormen et al. 2001], Section 5.5. The second algorithm prices *cyclic queries* and reduces the problem to a Bipartite Matching problem. Moreover, we provide several constructions that can reduce queries to either a cyclic or a chain query.

Third, we study the complexity of all conjunctive queries without self-joins. We prove a dichotomy of the data complexity of all conjunctive queries without self-joins, in PTIME or NP-complete, in Section 5.4: this is the main result of our article. In particular, we show that if a query can not be reduced to a chain or cyclic query, then it is NP-hard to price.

Our definition of arbitrage 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. If V determines Q, then a potential buyer interested in purchasing Q can purchase V instead, and derive from it the answer to Q: arbitrage occurs when the price of V is lower than that of Q. *Information-theoretic* determinacy, denoted $V \twoheadrightarrow Q$, is discussed by Segoufin and Vianu [2005] and by Nash et al. [2007, 2010] 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 restaurant chains 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 restaurant chains: 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 3.6). 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 5.6).

Lastly, we consider various practical generalizations of the pricing problem and discuss how to adapt our framework to handle them. We begin by showing why query containment fails as an abstraction for pricing. We next generalize selection-based pricing to the case where the views V are defined as a conjunction of selections over *multiple attributes* instead of just one. We then discuss the effect of allowing self-joins in the query Q. Lastly, we show how our framework can be easily extended to price certain types of sets of queries, $\{Q_1, Q_2, \ldots\}$, called *bundles*, instead of pricing just a single query Q at a time. Note that purchasing multiple queries simultaneously can be cheaper than buying them independently since purchased views may be shared across more than one query.

Organization. We begin by formally defining the basic pricing axioms and the general query pricing problem in Section 2. We introduce instance-based determinacy and analyze its complexity in Section 3. In Section 4, we discuss the complexity of pricing (based on instance-based determinacy). Section 5 presents our main contribution where we restrict the pricing problem to the practically useful and tractable case of pricing where views are selections and the queries are UCQs, and prove the dichotomy result that separates the problem of pricing the queries that are UCQs into a polynomial-time fragment and an NP-hard fragment. In Section 6, we briefly discuss several practical generalizations of the basic pricing problem as defined in Section 2. We present related literature and conclude in Section 7 and Section 8 respectively.

2. THE QUERY PRICING FRAMEWORK

In this section, we formally describe the pricing framework.

2.1. Notations

Fix a relational schema $\mathbf{R} = (R_1, \ldots, R_k)$; we denote a database instance[2] with $D = (R_1^D, \ldots, R_k^D)$, and the set of all database instances with $Inst_{\mathbf{R}}$ [Libkin 2004]. In this article we only consider monotone queries, and we denote by $\mathcal{L}$ a fixed query language. In particular, CQ and 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 CQ is *without self-joins* if each relation R_i occurs at most once in Q; for example, the query $Q(x, y) = R(x), S(x, y), R(y)$ has a self-join (since R occurs twice), whereas the query $Q(x, y) = R(x), S(x, y), T(y)$ is without self-joins. Also, a conjunctive query is *full* if all variables in the body appear in the head; as an example, the query $Q(x) = R(x, y)$ is not full (y does not appear in the head), whereas the query $Q(x, y) = R(x), S(x, y)$ is full. Moreover, a CQ is *boolean* if the head contains no variables. A boolean query returns a yes or no answer.

We also say that a body variable x is a *hanging variable* for Q if x appears only in a single atom. For example, for $Q(x, y) = R(x), S(x, y)$, the variable y is hanging (since it appears only in S), whereas x is not hanging.

Finally, we associate with each conjunctive query Q a hypergraph G as follows: for every body variable x, we introduce a new node $v(x) \in V(G)$ and for every atom $R(x_1, \ldots, x_k)$, we introduce a hyperedge $\{v(x_1), \ldots, v(x_k)\}$. A *connected component* of Q

[2]Throughout this article, we consider only *finite* databases.

refers to a connected component of the corresponding hypergraph G. For example, consider the query $Q(x, y, z) = R(x), S(x, y), T(z), U(z, w)$. Q has two connected components: $Q_1(x, y) = R(x), S(x, y)$ and $Q_2(z) = T(z), U(z, w)$.

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 $\mathbf{Q} : Inst_{\mathbf{R}} \to Inst_{\mathbf{R_Q}}$.

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 (which is a single query returning the emptyset regardless of the input). 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 $D \in Inst_{\mathbf{R}}$. A *pricing function* is a function $p_D : B(\mathcal{L}) \to \mathbb{R} \cup \{+\infty\}$.

The intuition is that if the user asks for the bundle $\mathbf{Q}$, then she has to pay the price $p_D(\mathbf{Q})$, where D is the database instance. If $p_D(\mathbf{Q}) = +\infty$, this is equivalent to saying that $\mathbf{Q}$ is not offered for sale. The positive infinity $+\infty$ behaves exactly as in the affinely extended real number system: for example, $a + \infty = +\infty$, and for any a, $a \le +\infty$.

The price is for an entire query bundle, not just for one query. For example, if a user needs to obtain 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 enforce that, in general, the pricing function is subadditive. In particular, the price of the bundle is at least as low as 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. The system then computes the pricing function on all queries in $\mathcal{L}$. Furthermore, this function must 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 $p_D(\mathbf{Q}) \le p_D(\mathbf{V})$, 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 can be defined in several ways.

Throughout this work, we will focus on one notion of determinacy, *instance-based determinacy*. We will discuss this in detail in Section 3, but for completeness we briefly give the definition here: $\mathbf{V}$ 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 $\mathbf{Q}(D) = \mathbf{Q}(D')$. It is possible that other notions of determinacy can be used for query pricing: for example one may use information-theoretic determinacy. To keep the framework general, we base our discussion on an abstract notion of determinacy, defined in the following. 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.

Definition 2.2. 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. $D \vdash \mathbf{ID} \twoheadrightarrow \mathbf{V}$

It is straightforward that instance-based determinacy satisfies the definition and hence is a determinacy relation. The definition implies the following property for determinacy relations.

LEMMA 2.3. *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 $\mathbf{V}, \mathbf{V} = \mathbf{V}$. □

As we will discuss in later sections, we often require that a determinacy relation satisfies a monotonicity property.

Definition 2.4 (*Monotonicity*). We say that a determinacy relation $\twoheadrightarrow$ is *monotone* for the query bundles $\mathbf{V}, \mathbf{Q}$ if, whenever $D_1 \subseteq D_2$ and $D_2 \vdash \mathbf{V} \twoheadrightarrow \mathbf{Q}$, then $D_1 \vdash \mathbf{V} \twoheadrightarrow \mathbf{Q}$.

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

Definition 2.5 (*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 $p_D(\mathbf{Q}) \leq \sum_{i=1}^{k} p_D(\mathbf{Q}_i)$.

Of course, even if $\mathbf{Q}_1, \ldots, \mathbf{Q}_k$ determine $\mathbf{Q}$, it may be nontrivial 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 article, however, we do not address the economic cost of the computation, focusing only on the information-theoretic aspect; that is, we assume that the only cost that matters is that of the data itself. Thus, if a pricing function is not arbitrage-free, 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 exist: for example, the trivial function $p_D(\mathbf{Q}) = 0$, for all $\mathbf{Q}$, is arbitrage-free; we will show nontrivial functions. First, we prove some basic properties.

PROPOSITION 2.6. *Any arbitrage-free pricing function p_D satisfies the following properties.*

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

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

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 Theorem 2.5: 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 nontrivial 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 2.7 (*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 $\{(\mathbf{V}_1, p_1), \ldots, (\mathbf{V}_m, p_m)\}$.

Definition 2.8 (*Validity*). A pricing function p_D for a database 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 2.9 (*Consistency*). A set of price points $\mathcal{S}$ for a database D is *consistent* if it admits a valid pricing function.

2.5. Axiom 2: Discount-Free

To see the intuition behind the second axiom, one can view the explicit price points in $\mathcal{S}$ 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 2.10 (*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 in the next section 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 and arbitrage-free price. It associates to any $\mathcal{S}$ (not necessarily consistent) a pricing function, which we call *arbitrage-price*. To introduce the formula, we need some notation. If $\mathbf{Q}_i$ for $i = 1, 2 \ldots, k$ are query 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 $p(\mathcal{C}) = \sum_{(\mathbf{V}_i, p_i) \in \mathcal{C}} p_i$.

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

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

The support of the bundle $\mathbf{Q}$ thus contains all the subsets of price points from $\mathcal{S}$ so that the corresponding views determine $\mathbf{Q}$.

Definition 2.11 (*Arbitrage-Price*). The *arbitrage-price* of a query bundle $\mathbf{Q}$ is

$$p_D^{\mathcal{S}}(\mathbf{Q}) = \begin{cases} \min_{\mathcal{C} \in supp_D^{\mathcal{S}}(\mathbf{Q})} p(\mathcal{C}) & \text{if } supp_D^{\mathcal{S}}(\mathbf{Q}) \neq \emptyset, \\ +\infty & \text{otherwise.} \end{cases} \tag{2}$$

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 $\mathbf{Q}$: purchase $\mathcal{C}$, then derive from it the answer to the query Q. It is also worth observing that for a query bundle $\mathbf{Q}$, $p_D^{\mathcal{S}}(\mathbf{Q})$ always has a finite price (and so is offered for sale) if and only if $D \vdash \mathbf{V}_1, \dots, \mathbf{V}_m \twoheadrightarrow \mathbf{Q}$. Indeed, if $\mathbf{Q}$ is determined by the set of views, then the support is not empty, and since $p_i < +\infty$ for all i, $p_D^{\mathcal{S}}(\mathbf{Q})$ must also be finite. On the other hand, if $\mathbf{Q}$ is not determined by the set of views, the support is empty and the price is $+\infty$; in this case, the query bundle is not offered for sale. We now prove the following.

LEMMA 2.12. *(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 $D \vdash \mathbf{Q}_1, \dots, \mathbf{Q}_k \twoheadrightarrow \mathbf{Q}$; we will prove that $p_D^{\mathcal{S}}(\mathbf{Q}) \leq \sum_i p_D^{\mathcal{S}}(\mathbf{Q}_i)$. Notice that if for some i we have that $supp_D^{\mathcal{S}}(\mathbf{Q}_i) = \emptyset$, then $p_D(\mathbf{Q}_i) = +\infty$ and the claim trivially holds. So now, for $i = 1, \dots, k$, let $\mathcal{C}_i^m = \arg\min_{\mathcal{C} \in supp_D^{\mathcal{S}}(\mathbf{Q}_i)} p(\mathcal{C})$, that is, the cheapest set of price points in the support of $\mathbf{Q}_i$. By the definition of arbitrage-price, $D \vdash \bigodot_{(\mathbf{V}_j, p_j) \in \mathcal{C}_i^m} \mathbf{V}_j \twoheadrightarrow \mathbf{Q}_i$ and $p_D^{\mathcal{S}}(\mathbf{Q}_i) = p(\mathcal{C}_i^m)$.

Let $\mathcal{C} = \bigcup_i \mathcal{C}_i^m \subseteq \mathcal{S}$ and $\mathbf{V}^m = \bigodot_{(\mathbf{V}_j, p_j) \in \mathcal{C}} \mathbf{V}_j$. 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[4]. It follows that $D \vdash \mathbf{V}^m \twoheadrightarrow \mathbf{Q}_i$, for every $i = 1, \dots, k$. By inductively applying Lemma 2.3(b), we derive $D \vdash \mathbf{V}^m \twoheadrightarrow \mathbf{Q}_1, \dots, \mathbf{Q}_k$ 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 nonnegative (Theorem 2.7). This proves that $p_D^{\mathcal{S}}$ is arbitrage-free. $\square$

The arbitrage-price is a fundamental formula because it allows us to check consistency, and, in that case, it gives the discount-free price. The following theorem captures how crucial arbitrage-price is for our pricing framework.

THEOREM 2.13. *Consider a set of price points $\mathcal{S}$. Let $p_D^{\mathcal{S}}$ denote the arbitrage-price function* (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, $p_D^{\mathcal{S}}$ is the unique, valid, and 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

[4]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.

of item 1 follows from two facts. First, $p_D^{\mathcal{S}}$ is arbitrage-free by Lemma 2.12(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 2.12(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 claim: if p_D is any valid pricing function for $\mathcal{S}$ then $p_i = p_D(\mathbf{V}_i) \leq p_D^{\mathcal{S}}(\mathbf{V}_i)$. 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. If $supp_D^{\mathcal{S}}(\mathbf{Q}) = \emptyset$, then $p_D(\mathbf{Q}) = +\infty$ and the claim holds trivially. So let $\mathcal{C} \in supp_D^{\mathcal{S}}(\mathbf{Q})$, 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

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

Since this holds for any $\mathcal{C}$ in the support of $\mathbf{Q}$:

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

This concludes the proof of the theorem. □

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 2.5: the theorem reduces this to a finite set.

Furthermore, it provides a simple but inefficient algorithm to compute the price of any query in our framework if we are equipped with a black box for computing determinacy: iterate over all 2^k subsets of $\mathcal{S}$, check whether a subset determines the query and keep the minimum cost subset; if no such subset exists, simply set the price to $+\infty$. In the rest of the article, we explore the black box of determinacy and also design algorithms to speed up the computation of prices.

3. INSTANCE-BASED DETERMINACY

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

Definition 3.1 (*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 $\mathbf{Q}(D') = \mathbf{Q}(D)$.

Our interest in instance-based determinacy is justified by the following proposition that is straightforward to show.

PROPOSITION 3.2. *For any* $\mathbf{V}, \mathbf{Q}$ *there exists a function* $f : Inst_{\mathbf{R_V}} \to Inst_{\mathbf{R_Q}}$ *such that, for all* 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* $D \vdash \mathbf{V} \twoheadrightarrow \mathbf{Q}$.

Example 3.3. Let $Q_1(x, y, z) = R(x, y), S(y, z)$, $Q_2(y, z, u) = S(y, z), T(z, u)$ and $Q(x, y, z, u) = R(x, y), S(y, z), T(z, u)$. Notice that for any database D, $D \vdash (Q_1, Q_2) \twoheadrightarrow Q$, because it suffices to define f as the function that joins $Q_1(D)$ and $Q_2(D)$; then $Q(D) = f(Q_1(D), Q_2(D))$. To decide whether Q_1 determines Q depends on the database D. For example, let D be a database instance s.t. $Q_1(D) = \emptyset$. Then $D \vdash Q_1 \twoheadrightarrow Q$, because we know that $Q(D) = \emptyset$: if f always returns the emptyset, then for any D' s.t. $Q_1(D) = Q_1(D')(= \emptyset)$ we have $Q(D') = f(Q_1(D'))$. On the other hand, if $D = \{R(a, b), S(b, c)\}$, then $D \vdash Q_1 \not\twoheadrightarrow Q$.

We now provide an alternate definition of instance-based determinacy that is based on the notion of *certain answers*.

Definition 3.4 (*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 identical to the definition of lossless views under the exact view assumption proposed in [Calvanese et al. 2002]. It is easy to check that the following alternative definitions of instance-based determinacy.

PROPOSITION 3.5. *Let* $\mathbf{V}$ *be a set of views,* D *a database,* Q *a query and* $\mathbf{E} = \mathbf{V}(D)$. *Then, the following are equivalent.*

(1) $\forall D', D''$, $\mathbf{V}(D') = \mathbf{V}(D'') = \mathbf{E}$ *implies* $Q(D') = Q(D'')$.
(2) $\forall D'$, $\mathbf{V}(D') = \mathbf{E}$ *implies* $Q(D') = cert_{Q,\mathbf{V}}(\mathbf{E})$.
(3) $\forall D'$, $\mathbf{V}(D') = \mathbf{E}$ *implies* $Q(D') = Q(D)$.

We next study the complexity of instance-based determinacy and show that it is decidable for a large class of queries.

THEOREM 3.6. *The combined complexity of the* INSTANCE-BASED DETERMINACY *problem,* $D \vdash \mathbf{V} \twoheadrightarrow \mathbf{Q}$, *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).*

We prove Theorem 3.6 by first determining the combined complexity and then by determining the data complexity.

LEMMA 3.7. *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 view 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 two databases D_1, D_2 such that the view $\mathbf{V}$ evaluates to E on both databases; $Q(D_1)$ contains the witness tuple t, while $Q(D_2)$ does not contain t.

To find the witness tuple, we enumerate over all possible tuples that are candidates for being a witness. Let A be the set of all constants that appear in E and let the arity of Q be k. Introduce k distinct constants $C = \{c_1, \ldots, c_k\}$ that do not appear in E. Then, it follows from the genericity of databases that we need only check for the witness in $(A \cup C)^k$. Thus there are at most $(|E| + k)^k$ distinct tuples that may be witness tuples, and these tuples are polynomially many in the size of the view extension E.

It is now easy to observe that it suffices to choose D_1, D_2 of polynomial size. Indeed, suppose that E contains n tuples and k is the maximum number of atoms over all views in $\mathbf{V}$. If there exists a database D with n tuples such that $\mathbf{V}(D) = E$ and $t \notin Q(D)$, then there must be a database $D' \subseteq D$ of size at most kn such that $\mathbf{V}(D') = E$ (since each tuple in E can be produced by at most k tuples). Since Q is monotone, it must also be that $t \notin Q(D')$; then D' is our D_2 of polynomial size. Similarly, if k' is the number of atoms in Q, and D is a database such that $\mathbf{V}(D) = E$ and $t \in Q(D)$, we can construct a database $D' \subseteq D$ of size at most $(n+1)\max(k, k')$ such that $\mathbf{V}(D') = E$ and $t \in Q(D')$; then D' is our D_1 of polynomial size.

Since query evaluation for UCQs has PTIME data complexity and NP-complete combined complexity, it follows that for finding a witness tuple, the data complexity is in NP and the combined complexity in NP^{NP}. Hence, the data complexity of Instance-Based Determinacy is in co-NP and the combined complexity in Π_2^P. □

Lemma 3.8. *The data complexity of* 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 [Abiteboul and Duschka 1998]. 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 $Q() = edge(X, Y), color(X, Z), color(Y, Z)$, that is, 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. □

4. THE COMPLEXITY OF PRICING

In this section, we discuss the complexity of pricing for instance-based determinacy in the case of CQs and UCQs.

Denote by Price($\mathcal{S}, \mathbf{Q}$) the decision version of the price computation problem: "given a database D and k, is the price $p_D^{\mathcal{S}}(\mathbf{Q})$ less than or equal to k"? Let us also denote by 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 4.1. *Suppose $\mathcal{S}, \mathbf{Q}$ consist of UCQs. Then, (a) the complexity of* Price($\mathbf{Q}$) *is in Σ_2^P and (b) the complexity of* Price($\mathcal{S}, \mathbf{Q}$) *is co-NP complete.*

Proof. For (a), to check whether $p_D^{\mathcal{S}}(\mathbf{Q}) \leq k$, 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 co-NP by Theorem 3.6) and that $\sum_i p_i \leq k$. For (b), instead of guessing, we can iterate over all subset of price points, since there is only a fixed number of them; hence it is co-NP. To prove co-NP hardness, consider the price points $\mathcal{S} = \{(\mathbf{V}, k)\}$. Then, $p_D^{\mathcal{S}}(\mathbf{Q}) \leq k$ if and only if $D \vdash \mathbf{V} \twoheadrightarrow \mathbf{Q}$, and recall from Theorem 3.6 that Instance-Based Determinacy is co-NP hard even for CQs. □

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 inquire about the price, then decide not to buy). We have an extensive discussion of tractability in Section 5, 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 PRICE-CONSISTENCY($\mathcal{S}$) be the problem of deciding whether a set $\mathcal{S}$ is consistent for a database D, and DETERMINACY($\mathbf{V}$, Q) the problem of checking determinacy $D \vdash \mathbf{V} \twoheadrightarrow Q$. The proof of Corollary 4.1 shows that the former problem is no more than exponentially worse than the latter. We prove here a weak converse:

PROPOSITION 4.2. *There is a polynomial time reduction from* DETERMINACY($\mathbf{V}$, Q) *to*[5] *the complement of* PRICE-CONSISTENCY($\mathcal{S}$).

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:

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

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 $kp + \epsilon = p_D(Q) \leq \sum_i p_D(V_i) = kp$, 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 2.13, 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 $D \vdash \mathbf{V} \twoheadrightarrow Q$. □

5. PRICING WITH SELECTIONS

The combined complexity for computing the price when the views and queries are UCQs is high: it is coNP-hard and in Σ_2^P. This is unacceptable in practice. In this section, we restrict the views on which the seller can set explicit prices and show that for a large class of queries, the price can be computed in polynomial time. This is the main result in the article, since it represents a quite practical framework for query-based pricing. We further show an even stronger result: for the class of conjunctive queries without self-joins, there exists a dichotomy of their pricing complexity into PTIME and NP-complete. The proof of this dichotomy is the most technically difficult result of the article and will be the focus of this section.

Organization. We first motivate and formally define the restriction to views as selections (Section 5.1). In Section 5.2, we study the complexity of instance-based determinacy if the price points are selections and show that for the class of UCQs the complexity is in polynomial time. In Section 5.4, we discuss the complexity of pricing and formally present the dichotomy result. The remaining part of the section presents the proof of the dichotomy, split into several steps.

5.1. The Pricing Framework with Selections

We restrict the views to *selection queries*, where the selection is on a single attribute. We will discuss in Section 6.3 how to extend several of our results in the case where the views are selections over multiple attributes.

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

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

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[6].

Example 5.1. Following the example from the introduction, CustomLists (customlists.net) sells the set of all businesses in any given state for \$199, thus it sells 50 selection views, one for each state.

Example 5.2. Infochimps (infochimps.com/marketplace) 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, 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 $Col_{R.X} = \{a_1, \dots, a_n\}$, called *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 $Col_{R.X}$[7] : 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. Thus, the explicit price points $\mathcal{S} = \{(V_1, p_1), (V_2, p_2), \dots\}$ are such that $V_i \in \Sigma$. We denote $p : \Sigma \to \mathbb{R}^+$ the partial function defined as: $p(V_i) = p_i$ if $(V_i, p_i) \in \mathcal{S}$. Finally, we say that a set of views $\mathbf{V} \subseteq \Sigma$ *fully covers* $R.X$ if $\Sigma_{R.X} \subseteq \mathbf{V}$.

Recall that 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 Price($\mathbf{Q}$), where the input consists of the database instance D, all columns $Col_{R.X}$, and the function p.

In order to give some more intuition about using selections as views, we discuss pricing with the following example.

Example 5.3. Consider $Q = R(x), S(x, y), T(y)$ over the database D in Figure 2(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.

[6]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.
[7]We will often refer to Col_x, where x is a variable that occurs in a query Q. In this case, Col_x is the intersection of all columns where the variable x appears in Q; it is easy to see that we can discard any other value, since it will never join. As an example, if $Q(x, y) = R(x), S(x, y)$, and our schema is $\{R(X), S(X, Y)\}$, $Col_x = Col_{R.X} \cap Col_{S.X}$.

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 $\mathbf{V} = \{\sigma_{R.X=a_1}, \sigma_{R.X=a_4}, \sigma_{S.Y=b_1}, \sigma_{S.Y=b_3}, \sigma_{T.Y=b_1}, \sigma_{T.Y=b_2}\}$. The reader may check that this is a minimal set that determines Q, hence the price of Q is $p_D^{\mathcal{S}}(Q) = 6$.

Let us also prove the following lemma.

LEMMA 5.4. *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 $R(X_1, \ldots, X_k)$. Since no attribute of R is fully covered, for every attribute X_i of R, there exists a selection $\sigma_{R.X_i=a_i} \notin \mathbf{V}$. W.l.o.g., let $X = X_1$ and consider the databases $D^+ = D \cup \{R(a, a_2, \ldots, a_k)\}$ and $D^- = D - \{R(a, a_2, \ldots, a_k)\}$. It is easy to see that $\mathbf{V}(D^+) = \mathbf{V}(D^-) = \mathbf{V}(D)$, 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 to the fact that $\mathbf{V}$ determines $\sigma_{R.X=a}$. □

The lemma gives us a simple criterion for checking whether $\mathcal{S}$ is consistent. By Theorem 2.13, 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 $\sigma_{R.X=a}$.

PROPOSITION 5.5 (CONSISTENCY FOR SELECTIONS). $\mathcal{S}$ *is consistent iff for every relation* R*, any* $\sigma_{R.X=a}$ *such that* $p(\sigma_{R.X=a})$ *is defined, and any attribute* Y *such that for every* $b \in Col_{R.Y}$*,* $p(\sigma_{R.Y=b})$ *is defined as*

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

This implies that consistency can be checked very efficiently in the case we use only selections. Moreover, the consistency is independent of the database instance.

5.2. The Complexity of Determinacy under Selections

We study here the complexity of determinacy when the views are selections. 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 [Abiteboul et al. 1995].

THEOREM 5.6. *Assume* $\mathbf{V} \subseteq \Sigma$. *Let* Q *be any monotone query that has PTIME data complexity. Then,* $D \vdash \mathbf{V} \twoheadrightarrow Q$ *can be decided in PTIME data complexity.*

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

$$R^{D^{min}} = \bigcup_{\sigma_{R.X_i=a} \in \mathbf{V}} \sigma_{R.X_i=a}(D), \tag{3}$$

$$R^{D^{max}} = R^{D^{min}} \cup \{t \in Col_{R.X_1} \times \cdots \times Col_{R.X_k} \mid t \text{ invisible in } R\}. \tag{4}$$

Notice that both databases are of polynomial size in the size of the columns and the input database. Moreover, they can be constructed from $\mathbf{V}(D)$, without having access to the underlying database D. Additionally, $D^{min} \subseteq D^{max}$ by definition. The following lemma establishes that both instances agree with D under the views $\mathbf{V}$.

LEMMA 5.7. $\mathbf{V}(D^{min}) = \mathbf{V}(D^{max}) = \mathbf{V}(D)$.

PROOF. We first examine D^{min}. For a view $\sigma_{R.A=c} \in \mathbf{V}$, consider a tuple $t_R \in \sigma_{R.A=c}(D^{min})$. 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 hence t_R belongs 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 why we have constructed D^{min} and D^{max}: they are the minimum and maximum databases respectively that agree with $\mathbf{V}(D)$.

LEMMA 5.8 (SANDWICH LEMMA). *For any database D' s.t. $\mathbf{V}(D') = \mathbf{V}(D)$, we have that $D^{min} \subseteq D' \subseteq D^{max}$.*

PROOF. We will first show that $D^{min} \subseteq D'$. For the sake of contradiction, suppose that for a relation R in D there exists a 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 for some relation R in D there exists a tuple $t_R \in R^{D'}$ such that $t_R \notin D^{max}$. We now distinguish two cases. In the first case, $t_R \in \sigma_{R.A=c}(D')$ for some $\sigma_{R.A=c} \in \mathbf{V}$. We can apply Lemma 5.7 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 all $\sigma_{R.A=c} \in \mathbf{V}$. 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 again. □

The lemma "sandwiches" every database that agrees with $\mathbf{V}(D)$ between the minimum and maximum database. We next leverage this property to show how we can efficiently characterize determinacy.

PROPOSITION 5.9. *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 5.8, $D^{min} \subseteq D' \subseteq D^{max}$. 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 5.8 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 5.7 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 checks whether $Q(D^{min}) = Q(D^{max})$, in which case it outputs yes, otherwise no. The validity of the

R

X
a_1

S

X	Y
a_2	b_1
a_1	b_1

$$Q(x,y) = R(x), S(x,y)$$
$$Col_x = \{a_1, a_2\}$$
$$Col_y = \{b_1, b_2\}$$

D^{min}

X

X	Y
a_1	b_1

D^{max}

X
a_1

X	Y
a_1	b_1
a_2	b_1
a_2	b_2

Fig. 1. Given D and $Q(x, y) = R(x), S(x, y)$, does the set of views $\mathbf{V} = \{\sigma_{R.X=a_2}, \sigma_{S.X=a_1}\}$ determine Q? By Proposition 5.9, we construct the databases D^{min} and D^{max}, as the figure shows. Indeed, as the construction claims, $\sigma_{R.X=a_2}(D) = \sigma_{R.X=a_2}(D^{min}) = \sigma_{R.X=a_2}(D^{max}) = \emptyset$ (similarly for $\sigma_{S.X=a_1}$). However, $Q(D^{min}) = \emptyset$, whereas $Q(D^{max}) = \{(a_1, b_1)\}$. Thus, $\mathbf{V}$ does not determine Q.

algorithm follows from Proposition 5.9. 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. This concludes the proof of the theorem. □

The example in Figure 1 presents our construction and the algorithm to decide determinacy when the views are selections.

Unfortunately, the PTIME complexity depends not only on the $|\mathbf{V}(D)|$, but also on the size of the columns (which is usually larger). We next show that, under stronger conditions than monotonicity and PTIME evaluation, we can construct an algorithm such that its complexity depends only on $|\mathbf{V}(D)|$.

Lemma 5.10. *Let Q be a positive Datalog query without inequalities. Then, there exists a database D^m of polynomial size in $|\mathbf{V}(D)|$ 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 $c_x \in Col_x$ that does not appear in $\mathbf{V}(D)$. Then, construct a function $f : Col \to Col$, 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 5.8, $Q(D^m)$ 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 $t \notin Q(D^{min})$. Since Q is a Datalog query without any inequalities, $f(t) \in Q(D^m)$. 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. □

This theorem implies that we can replace in our algorithm D^{max} with D^m, where the size of the latter depends only on $\mathbf{V}(D)$. Hence, the complexity of determinacy is now polynomial only in $|\mathbf{V}(D)|$.

5.3. Preliminaries

We prove now some basic results that will be used in proofs throughout this section.

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

LEMMA 5.11. *Consider a database D, a set of views $\mathbf{V} \subseteq \Sigma$ and a full CQ Q. 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 $t \in Q(D)$, for any projection t_R of t to 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 to some relation R in Q such that $t_R \notin R^D$ and $\mathbf{V}$ covers t_R.*

PROOF. We first show the "only if" direction. If $t \in Q(D)$ and $\mathbf{V}$ 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. If $t \notin Q(D)$ and for every atom R in Q 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 view extensions $\mathbf{V}(D)$ are not modified; however, $t \in Q(D^+)$, which means that $\mathbf{V}$ does not determine Q.

For the "if" direction, we again distinguish two cases. If $t \in Q(D)$, since $\mathbf{V}$ covers every tuple t_R for any R, any D' that agrees with $\mathbf{V}(D)$ discovers correctly that $t \in Q(D)$. If $t \notin Q(D)$, since $\mathbf{V}$ 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 $Q(D)$. □

PROPOSITION 5.12. *If $\mathbf{V} \subseteq \Sigma$ and $\mathbf{Q}$ is a bundle with 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, if for some database D we have $D \vdash \mathbf{V} \twoheadrightarrow Q$, then 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: $t_{R_1}, \dots, t_{R_k}$. 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 5.11. 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 5.11, 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 5.11, 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), (b) and invoking Lemma 5.11, we obtain the desired proposition. □

Definition 5.13. Given a database D and a query Q, we call a subset T^c of the database D *critical* for Q if $Q(D) \neq Q(D - T^c)$.

A critical set of tuples is thus a part of the database that is crucial for answering Q. 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 5.14. *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 noncritical subset T of D.*

PROOF. Since $D - T \subseteq D$, we can apply Proposition 5.12 (monotonicity) to obtain that $D - T \vdash \mathbf{V} \twoheadrightarrow Q$. Hence, it follows from Proposition 3.2 that there exists a function f such that for every D' s.t. $\mathbf{V}(D') = \mathbf{V}(D - T)$, $f(\mathbf{V}(D')) = Q(D - T)$. Now, consider any database D^0 such that $(\mathbf{V} - T)(D^0) = (\mathbf{V} - T)(D)$. We will show that $f((\mathbf{V} - T)(D^0)) = Q(D)$, which implies the determinacy relation. Indeed, note 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 noncritical set of views for Q. □

We next show a basic theorem that allows us to safely remove redundant views from a set that determines some query Q.

THEOREM 5.15 (REDUNDANT VIEWS). *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 all tuples $t \in \mathbf{V}^0(D)$ are such that either noncritical 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 noncritical tuples of T:

$$T^c = \{t \in T \mid Q(D) = Q(D - \{t\})\}.$$

We will prove the theorem in several steps as follows:

$$D \vdash \mathbf{V} \setminus \mathbf{V}^0 \twoheadrightarrow (\mathbf{V} \setminus \mathbf{V}^0) - T^c \twoheadrightarrow \mathbf{V} - T^c \twoheadrightarrow Q.$$

LEMMA 5.16. $D \vdash \mathbf{V} \setminus \mathbf{V}^0 \twoheadrightarrow (\mathbf{V} \setminus \mathbf{V}^0) - T^c$.

PROOF. We will show the more general result that, for any $\mathbf{V}' \subseteq \Sigma$, $D \vdash \mathbf{V}' \twoheadrightarrow \mathbf{V}' - T^c$. To prove this, we first show that for any $V \in \Sigma$, $D \vdash V \twoheadrightarrow V - T^c$. 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 $f(V(D)) = V(D) - T^c$.

By repeatedly applying the augmentation property for all the other $V' \in \mathbf{V}'$, we obtain that $D \vdash \mathbf{V}' \twoheadrightarrow \mathbf{V}' - T^c$. □

LEMMA 5.17. $D \vdash (\mathbf{V} \setminus \mathbf{V}^0) - T^c \twoheadrightarrow \mathbf{V} - T^c$.

PROOF. We first 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 the assumption of the theorem, $(\sigma_{R.y=c} - T^c)(D)$ contains only tuples that also belong to some view other than those in $\mathbf{V}^0$ (since every noncritical tuple belongs in T^c); hence, the view can be precisely reconstructed.

Applying repeatedly augmentation and transitivity, we obtain the result. □

LEMMA 5.18. $D \vdash (\mathbf{V} - T^c) \twoheadrightarrow Q$.

PROOF. We first show that T^c is not critical. For the sake of contradiction, assume 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 a tuple $t \in Q(D)$ s.t. $t \notin Q(D - T^c)$. However, since Q is a full query, 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.

We can now directly apply Lemma 5.14 to obtain that $D \vdash (\mathbf{V} - T^c) \twoheadrightarrow Q$. □

Combining the lemmas through the transitivity property of determinacy, we conclude that $D \vdash \mathbf{V} \setminus \mathbf{V}^0 \twoheadrightarrow Q$. □

As a corollary of Theorem 5.15, we can show the following lemma.

LEMMA 5.19. *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 5.15, it suffices to show that every tuple $t_R \in \sigma_{R.A=d}$ is either noncritical 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, $t_R \in \sigma_{R.B=d}(D)$. □

5.4. The Complexity of Pricing Under Selections

Having determined the complexity of determinacy when the price points are selections, we are equipped to explore the complexity of pricing.

COROLLARY 5.20 (COMPLEXITY). *Let* $\mathbf{Q}$ *be a bundle of monotone queries that have PTIME 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 $\mathbf{V} \subseteq \mathcal{S}$, 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 2.3) and that $\sum_{V \in \mathbf{V}} p(V) \leq k$. From Theorem 5.6, the check on determinacy can be done in polynomial time. □

The restriction to selection queries has lowered the complexity of price computation from Σ_2^P (Corollary 4.1) to NP. However, it is not possible to further lower the complexity of pricing for the class of CQs; indeed, as we will see, there are CQs where pricing is NP-complete. There are also CQs where pricing under selections is in PTIME. This raises an important question: can we syntactically characterize the pricing complexity for the class of CQs?

In the remaining section, we 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 single queries, not to query bundles: we will discuss bundles in Section 6.5.

We next define the class of normalized queries, which will be of interest to us, since we will show that the pricing of any conjunctive query w/o self-joins can be reduced to computing the price of a normalized query.

Definition 5.21. A *normalized query* is a full conjunctive query Q without self-joins such that Q has a single connected component and no constants, no multiple occurrences of a variable in the same atom and no hanging variables.

A normalized query where every atom is unary or binary is called *2-normalized*.

Example 5.22. The query $Q_1(x, y, z) = R(x), S(x, y, z), T(x), U(y, z)$ is a normalized query. The following queries are not normalized: $Q_2(x, y, z) = R(x), S(x, y, z), T(x)$ (z is hanging), $Q_3(x, y) = R(x), S(x, `a`, y)$ (a constant exists), $Q_4(x, y) = R(x), S(x, x, y)$ (x appears twice in S).

We identify two basic classes of normalized queries where pricing is in PTIME and provide algorithms for both: *chain queries* and *cyclic queries*.

Definition 5.23. A *chain query* is a 2-normalized conjunctive query $Q = R_0, R_1, \ldots, R_k$ such that any two consecutive atoms R_i, R_{i+1} share exactly one variable. Let us denote the class of chain queries by CHQ.

Some examples of CHQ queries are the following:

$$Q_1(x, y) = R(x), S(x, y), T(y),$$
$$Q_2(x, y, z) = R(x), S(x, y), T(y), U(y), V(y, z), W(z).$$

We present and analyze an algorithm that computes the price for any chain query in Section 5.5.

Definition 5.24. A cyclic query C_k is a 2-normalized conjunctive query of the form $C_k(x_1, \ldots, x_k) = R_1(x_1, x_2), \ldots, R_k(x_k, x_1)$.

The algorithm for computing C_k is described in Section 5.6. It is quite different from the reduction to MIN-CUT that we used for chain queries, suggesting that these two classes cannot be unified in a natural way. The class of queries C_k is also much more brittle than chain queries: adding a single unary predicate makes the query NP-hard. For example, see the query H_2 in Theorem 5.35: it is obtained by adding one unary predicate to C_2, and is NP-hard. By contrast, by adding a unary predicate to a chain query we still obtain a chain query.

We can now formally define the dichotomy result for the complexity of pricing.

THEOREM 5.25 (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 $Q_1, \ldots, Q_k$, then: if* PRICE(Q_i) *is in PTIME for all $i = 1, \ldots, k$, it is in PTIME; else, if at least one component Q_i is NP-complete, it is NP-complete.*
—*Else if Q is neither full nor boolean, it is NP-complete.*
—*Else if Q is a boolean query, let Q^f be the corresponding full query (add all variables to the head): the complexity of Q is the same as that of Q^f.*
—*Else if Q is a full CQ, let Q^n be obtained from Q by replacing each atom R with an atom R' such that R' contains exactly one occurrence from each nonhanging variable from R (Q^n is normalized): if Q^n is a chain or cyclic query, it is PTIME; otherwise, it is NP-complete.*

We provide an example of how to apply Theorem 5.25 as follows.

Example 5.26. Consider the following query $Q() = R(x, y), S(y, z, w), T(w, \text{‘}b\text{‘}), U(v)$. Note that Q has two connected components, $Q_1() = R(x, y), S(y, z, w), T(w, \text{‘}b\text{‘})$ and $Q_2() = U(v)$. Since Q_1 is boolean, it suffices to study the pricing complexity of the full query $Q_1^f(x, y, z, w) = R(x, y), S(y, z, w), T(w, \text{‘}b\text{‘})$. From this, consider the normalized query $Q_1^{fn}(y, w) = R^n(y), S^n(y, w), T^n(w)$ (we have removed the constants and hanging variables). Since Q_1^{fn} is a chain query, Q_1 is computable in polynomial time. Similarly, Q_2 can also be computed in polynomial time; thus, Q is also in PTIME.

On the other hand, pricing the query $Q(x, y, z) = R(x), S(x, x, y), T(y, x, z)$ in NP-complete. Indeed, the normalized query $Q^n(x, y) = R(x), S(x, y), T(y, x)$ is neither chain or cyclic, and hence it is hard to price.

The proof of Theorem 5.25 is split into four parts. In Section 5.5, we present the PTIME algorithm for chain queries and in Section 5.6 we present the algorithm for cyclic queries. In Section 5.7, we identify several basic CQs for which pricing is hard. Finally, in Section 5.8, we show how to reduce any CQ either to a hard query or to a query computable in polynomial time (cyclic or chain query); the latter part also concludes the dichotomy proof.

5.5. A PTIME Algorithm for Chain Queries

We prove here the following theorem.

THEOREM 5.27 (CHAIN QUERIES). *Assume that all explicit price points in $\mathcal{S}$ are selection queries. Then, the price of any chain query can be computed in PTIME (data complexity).*

We show that the price of chain query can be reduced to the MIN-CUT problem, which is the dual of the MAX-FLOW graph problem and can be solved in polynomial time [Cormen et al. 2001]. The running example for this section will be the chain query $Q(x, y) = R(x), S(x, y), T(y)$.

Given a chain query Q with atoms $R_0, \ldots, R_k$, 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 $Q_{[i:i-1]} = Col_{x_i}$. Define the *left-, middle-, and right-partial-answers*:

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

where Π is the projection operator.

We next describe the construction of the directed graph G. To construct the vertex set $V(G)$, we first introduce a source node s and a target node t. Moreover, for each attribute $R.X$ and constant $a \in Col_{R.X}$, we introduce two nodes: $v_{R.X=a}$ and $w_{R.X=a}$. The edges $E(G)$ are constructed as follows.

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[8] $p(\sigma_{R.X=a})$ 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 the edge

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

Capacity = ∞.
Skip edges. For all partial answers we create the edges

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

In all cases, capacity = ∞.

In particular, since $Lt_0 = Col_{x_0}$, $Md_{[i:i-1]} = Col_{x_i}$, $Rt_k = Col_{x_k}$, the construction will include the following edges, which are independent of the content of the database D and depend only on the columns:

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

We explain now the intuition behind the graph construction, and will also refer to the example of Figure 2. Notice that the view edges of finite capacity in G are in one-to-one correspondence with the views in $\mathcal{S}$, and the view edges in general are in one-to-one correspondence with the views in Σ. The main invariant is: *for every set of view edges C, 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:

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

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, $t = (u_1, u_2, \dots, u_k)$, where $u_1 \in Col_{x_1}, \dots, u_k \in Col_{x_2}$. 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 $i = 0, \dots, k$, denoting

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

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

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

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

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

X	Y
a_1	b_1
a_1	b_2
a_2	b_2

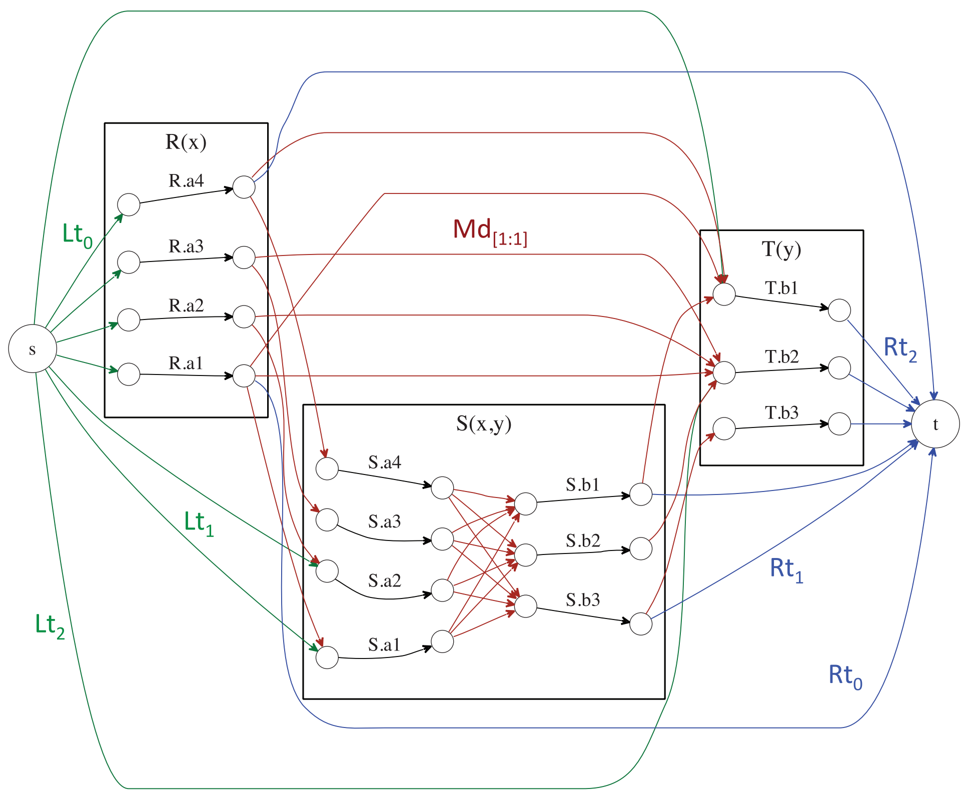
$Q_{[1:2]}(D)$

X	Y
a_1	b_1
a_4	b_1

$Q(D)$

X	Y
a_1	b_1

(a)



(b)

Fig. 2. (a) An example database D and query Q, along with the answers to the partial queries $Q_{[0:1]}$, $Q_{[1:2]}$, $Q_{[0:2]}$. (b) The flow graph G that corresponds to Q, D (see Theorem 5.28).

$a = u_i$ and[9] $b = u_{i+1}$, 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 2

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

the answer $(a_1, b_1) \in Q(D)$ leads to three $s - t$ paths.

$$s \xrightarrow{skip} v_{R.X=a_1} \xrightarrow{view} w_{R.X=a_1} \xrightarrow{skip} t$$
$$s \xrightarrow{skip} v_{S.X=a_1} \xrightarrow{view} w_{S.X=a_1} \xrightarrow{tuple} v_{S.Y=b_1} \xrightarrow{view} w_{S.Y=a_1} \xrightarrow{skip} t$$
$$s \xrightarrow{skip} v_{T.Y=b_1} \xrightarrow{view} w_{T.Y=b_1} \xrightarrow{skip} t$$

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 $a = u_i, b = u_{i+1}, c = u_j, d = u_{j+1}$:

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

leading to the following $s - t$ path:

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

To summarize, we prove the following.

THEOREM 5.28. *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.

PROOF. We prove that the reduction to MIN-CUT we described earlier is indeed valid. As we have previously discussed, there exists a one-to-one correspondence of views in Σ to view edges 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 view.

We start by showing that every set of views that determines Q corresponds to a set of edges in the graph that is a cut. □

LEMMA 5.29. *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 an $s - t$ path uniquely by listing the view edges it crosses, which correspond to selections (since any path visits alternatingly v, w nodes). Moreover, if a path crosses a selection $\sigma_{R_i.X=a}$, where R_i is binary, the path will also have to cross some selection $\sigma_{R_i.Y=b}$; in this case, we can say that the path crosses the tuple $R(a, b)$. Hence, we can equivalently describe an $s - t$ path in the graph as a sequence of tuples $t_1, t_2, \ldots, t_\ell$ (which may or may not occur in D).

The crucial observation is that we can add or remove the tuples $t_1, \ldots, t_\ell$ from the database without modifying the view extensions $\mathbf{V}(D)$, since the selections on these values are not present in $\mathbf{V}$. Consider the two databases $D^+ = D \cup \{t_1, \ldots, t_\ell\}$ and $D^- = D - \{t_1, \ldots, t_\ell\}$. 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} $(i = 1, \ldots, \ell - 1)$ and let $t_i \in R_j, t_{i+1} \in R_{j'}$, where $j < j'$. Let $a = t_i.x_{j+1}$ and $b = t_{i+1}.x_{j'}$. 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 produce an extra tuple for $Q(D^+)$ that will not be in D^-, a contradiction. □

For the converse direction, notice that we can assume w.l.o.g. that an $s - t$ cut contains only view edges. Indeed, if the cut has finite cost, it contains only view edges by construction; if its cost is ∞, then we can trivially obtain an equal cost cut (of infinity) by including all the possible view edges (since every $s - t$ path includes at least one such edge). We can now show the following.

LEMMA 5.30. *If a set of view edges S is a cut for G, then for $\mathbf{V} = \{V(e) \mid e \in S\}$, $D \vdash \mathbf{V} \twoheadrightarrow Q$.*

PROOF. In order to prove the lemma, we will apply Lemma 5.11. For this, we need to consider two cases.

First, assume that $t \in Q(D)$. Then, consider all the projections of t on the different atoms in Q: $t_{R_0}, \dots, t_{R_k}$. 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, for any $i = 0, \dots, k$, t_{R_i} will be covered by $\mathbf{V}$.

For the other case, let $t \notin Q(D)$. Again, consider the projections of t: $t_{R_0}, \dots, t_{R_k}$ and keep only these 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 is covered by $\mathbf{V}$. □

This concludes the proof that the reduction is indeed correct. □

5.6. A PTIME Algorithm for Cyclic Queries

We describe and analyze an algorithm with polynomial data complexity for the cyclic queries $C_k(x_1, \dots, x_k) = R_1(x_1, x_2), \dots, R_k(x_k, x_1)$. The algorithm reduces the pricing of a cyclic query to WEIGHTED BIPARTITE VERTEX COVER, which can be solved in polynomial time. We first extend the definition of a *full cover*. To simplify the presentation and take advantage of the symmetry of a cyclic query, we define $R_0 \equiv R_k$.

Definition 5.31 (*Full Cover*). Let variables x_i, x_j, where $1 \leq i \leq j \leq k$. For a set of views $\mathbf{V} \subseteq \Sigma$, we say that the pair (x_i, x_j) is *fully covered* by $\mathbf{V}$ if, for any $a \in Col_{R_{i-1}.x_i}$, $b \in Col_{R_j.x_j}$ 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 $\mathbf{V}$. Note that this is equivalent to saying that for every $a \in Col_{x_i}$, $\mathbf{V}$ contains one of $\sigma_{R_{i-1}.x_i=a}, \sigma_{R_i.x_i=a}$.

LEMMA 5.32. *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 $\sigma_{R_{i-1}.x_i=a_i}, \sigma_{R_i.x_i=a_i} \notin \mathbf{V}$. Let $T = \{R_1(a_1, a_2), \dots, R_k(a_k, a_1)\}$ and consider the databases $D^+ = D \cup T$ and $D^- = D - T$. By our construction, $\mathbf{V}(D^+) = \mathbf{V}(D^-)$. However, $(a_1, \dots, a_k) \in C_k(D^+)$, whereas $(a_1, \dots, a_k) \notin C_k(D^-)$, which contradicts the fact that $\mathbf{V}$ determines C_k. □

Lemma 5.32 guarantees that at least one variable x_i is fully covered. Let F be any fixing of this variable (which, due to symmetry, we can assume that it is w.l.o.g. x_1) and further a fixing of whether any pair of variables (x_i, x_j), $1 \leq i \leq j \leq k$, is fully covered or not. We will construct a weighted bipartite graph $G[F]$ for any such fixing F.

The Graph Construction. We construct a weighted bipartite graph $G[F] = (A, B, E)$. We introduce a node in $G[F]$ for every selection in Σ. The left partition A includes the left-selections in a relation (i.e., of the form $\sigma_{R_i.x_i=a}$), whereas B the right-selections (i.e., of the form $\sigma_{R_i.x_{i+1}=a}$). The weight of each node is equal to the price of the corresponding selection.

As for the edges, we distinguish the following cases.

Start edges. For every $a \in Col_{x_1}$, we add the edge $(\sigma_{R_1.x_1=a}, \sigma_{R_k.x_1=a})$.
Full Cover edges. If a 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 every tuple $t = (a_1, \dots, a_k) \in Col_{x_1} \times \cdots \times Col_{x_k}$, consider the projections of t at all relations: $t_{R_1}, \dots, t_{R_k}$.

(1) If $t \in C_k(D)$, for $i = 1, \dots, k$, we add the edge $(\sigma_{R_i.x_i=a_i}, \sigma_{R_i.x_{i+1}=a_{i+1}})$.
(2) If $t \notin C_k(D)$, 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 $P_t = \{(x_{i_1+1}, x_{i_2}), \dots, (x_{i_{\ell-1}+1}, x_{i_\ell})\}$. We 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.

For a graph G, let VC(G) be the cost of the minimum vertex cover of G. The algorithm for pricing cyclic queries (CYCLIC PRICE) assigns to C_k the price $\min_F\{VC(G[F])\}$.

PROPOSITION 5.33. CYCLIC PRICE *has polynomial data complexity.*

PROOF. First, note that the number of choices for F depends only on C_k and thus is constant for data complexity. Second, weighted bipartite vertex cover is in PTIME. □

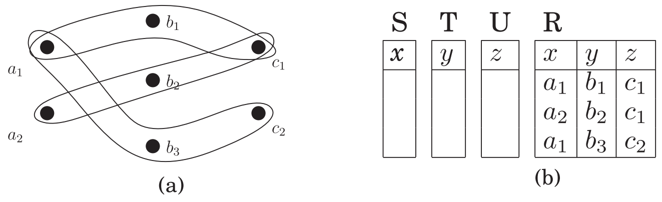
We next show the validity of the algorithm. For this, it suffices to show what follows.

PROPOSITION 5.34. *If the optimum solution to pricing C_k satisfies F, VC($G[F]$) equals the price of C_k.*

PROOF. For the one direction, let us 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 $D \vdash \mathbf{V} \twoheadrightarrow C_k$. Indeed, consider a tuple $t = (a_1, \dots, a_k)$. If $t \in C_k(D)$, by the construction of $G[F]$, any projection of t to an R_i, (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 C_k such that the tuples do not belong in D: $t_{R_{i_1}}, \dots, t_{R_{i_\ell}}$. If any 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 5.11 to conclude that $\mathbf{V}$ indeed determines C_k.

For the other direction, let us 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 $e \in E(G[F])$ that is not covered. First, note that e can not be one of the start edges, since $\mathbf{V}$ satisfies F and thus fully covers x_1. For the same reason, e can not be one of the full cover edges. Hence, e must be a tuple edge, which implies that the introduction of e can be traced back to a tuple t.

If $t = (a_1, \dots, a_k) \in C_k(D)$, by Lemma 5.11 every edge introduced by t would be covered by $\mathbf{V}$. Consequently, $t \notin C_k(D)$. Let $t_{R_{i_1}}, \dots, t_{R_{i_\ell}}$ be the tuple projections of t that do not appear in D. Consider any 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 $(a_{i_m+1}, a_{i_{m+1}}) \in Md_{[i_m:i_{m+1}]}$, 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 $\sigma_{R_{i_m}.x_{i_m+1}=a'_{i_m+1}}, \sigma_{R_{i_{m+1}}.x_{i_{m+1}}=a'_{i_{m+1}}} \notin \mathbf{V}$. Now, consider the tuple set $T = \{R_{i_1}(a_{i_1}, a'_{i_1+1}), R_{i_2}(a'_{i_2}, a'_{i_2+1}), \dots, R_{i_\ell}(a'_{i_\ell}, a_{i_\ell+1})\}$. Let $D^+ = D \cup T$ and $D^- = D - T$. Since e is not covered, the tuple $(a_{i_1}, a_{i_\ell+1})$ is also not covered by $\mathbf{V}$; this fact, together with our construction imply that $\mathbf{V}(D^+) = \mathbf{V}(D^-)$. However, notice that D^+ contains now one extra tuple from D^-, a contradiction. □



S	T	U	R		
x	y	z	x	y	z
			a_1	b_1	c_1
			a_2	b_2	c_1
			a_1	b_3	c_2

Fig. 3. An example of a 3-partite 3-uniform hypergraph, along with the corresponding database for the reduction to PRICE(H_1).

5.7. Hardness Results

In this section, we prove the NP-hardness for several CQs. These queries are the essential hard cases for pricing, and we will show in the next section that every other hard query reduces to one of them.

THEOREM 5.35 (NP-COMPLETE QUERIES). PRICE(Q) *is NP-complete (data complexity) when Q is any of the following queries:*

$$H_1(x, y, z) = R(x, y, z), S(x), T(y), U(z), \tag{5}$$

$$H_2(x, y) = R(x), S(x, y), T(x, y), \tag{6}$$

$$H_3(x) = R(x, y), \tag{7}$$

$$H_4(x_1, x_2, x_3, y) = R_1(x_1), S_1(x_1, y), R_2(x_2), S_2(x_2, y), R_3(x_3), S_3(x_3, y). \tag{8}$$

If Q is one of H_1, H_2, H_4 then the pricing complexity remains NP-complete even when the database instance D is restricted s.t. $Q(D) = \emptyset$.

We next show the hardness for pricing each of the given queries. We start by showing the hardness for PRICE(H_1), which is a special case of the following proposition.

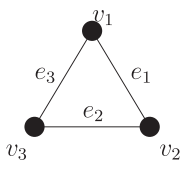
PROPOSITION 5.36. *Let Q be a full query with three variables x, y, z such that it contains the atom $R(x, y, z)$ and variable $i \in \{x, y, z\}$ belongs also in an atom $R_i \neq R$ (R_x, R_y, R_z are not necessarily different). Then,* PRICE(Q) *is NP-complete.*

PROOF. The hardness for pricing H_1 is obtained in the case where all R_x, R_y, R_z are different and contain only x, y, z respectively.

The reduction for Q is from 3-PARTITE 3-UNIFORM HYPERGRAPH VERTEX COVER (3P3UHVC), which is proven NP-complete in Gottlob and Senellart [2010]. In this, we are given a hypergraph $G(A, B, C, E)$ which is 3-partite (no vertices of the same part can belong in the same hyperedge). The vertex sets A, B, C denote the three parts. The graph is also 3-uniform, which means that each hyperedge contains exactly 3 vertices, one from each part. The decision version of the problems asks whether there exists a vertex cover of G of size $\leq k$; recall that a vertex cover is a set of vertices V such that every edge contains at least one vertex from V.

We now define the reduction. Let the columns be $Col_x = A$, $Col_y = B$, $Col_z = C$. We construct a database D where R_x^D, R_y^D, R_z^D are empty and $R^D = E$ (see Figure 3). 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 $i = x, y, z$. Note that this construction defines a straightforward one-to-one mapping from vertices 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 and for a set of nodes S, define also $\sigma(S) = \bigodot_{v \in S} \sigma(v)$. 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. Equivalently, we will show that $S = \{v_1, \ldots, v_k\}$ is a vertex cover for G if and only if $D \vdash \sigma(S), \Sigma_R \twoheadrightarrow Q$ (since the cost of the views $\sigma(S), \Sigma_R$ is



(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)

Fig. 4. An example graph G and the corresponding database for the reduction to PRICE(H_2).

exactly k and also w.l.o.g. any views of cost k will be of this form). Applying Lemma 5.11 and since $Q(D) = \emptyset$, we obtain that $D \vdash \sigma(S), \Sigma_R \twoheadrightarrow Q$ is equivalent to the following: for every tuple $(a, b, c) \in Col_x \times Col_y \times Col_z$ such that $(a, b, c) \in R^D$, at least one of the views $\sigma_{R_x.x=a}, \sigma_{R_y.y=b}, \sigma_{R_z.z=c}$ must belong in $\sigma(S)$. However, this is equivalent to the fact that for any edge $(a, b, c) \in E$, one of its vertices is covered by S. $\square$

PROPOSITION 5.37. 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. Given an undirected graph $G(V, E)$, the VC problem asks for the minimum number of vertices that cover every edge of G.

We construct an instance of H_2 as follows (See Figure 4). First, we take the columns to be $Col_x = V \cup \{v_0\}$ where $v_0 \notin V$ is a dummy value, and $Col_y = E$. Second, for every edge $e = (v_1, v_2) \in E$, we add (v_1, e) to S^D and (v_2, e) to T^D (here, we assume an arbitrary orientation of the edges that will be fixed throughout the reduction). Note that for each edge one endpoint contributes to S and the other to T. Finally, we add the dummy node v_0 to R^D. Note that $H_2(D) = \emptyset$, since $R^D = \{v_0\}$ and v_0 has no corresponding tuples to join: indeed, $\forall e \in Col_y$, $S(v_0, e)$ and $T(v_0, e)$ are both false.

To construct the price points $\mathcal{S}$, we price the selection queries as follows.

—$\forall v \in V$, $p(\sigma_{R.x=v}) = 1$ and $p(\sigma_{R.x=v_0}) = 0$.
—$\forall a \in Col_x$, $p(\sigma_{S.x=a}) = p(\sigma_{T.x=a}) = |E|$.
—$\forall e \in Col_y$, $p(\sigma_{S.y=e}) = p(\sigma_{T.y=e}) = 1$.

Note that given a set of selections $\mathbf{V}$ that determines H_2, 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 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 $v \in C$, $\sigma_{R.x=v}$ is selected and hence any database D' that agrees with $\sigma_{R.x=v} \in \mathbf{V}$ can not contain (v, e) (since $\sigma_{R.x=v}(D) = \emptyset$). 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, the construction of T implies that $(v, e) \notin \sigma_{T.y=e}(D)$ 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 $\sigma_{S.y=e}, \sigma_{T.y=e} \in \mathbf{V}$ prevent (v, e) (for any $v \in Col_x$) to belong to $H_2(D)$. And, so do the views $\sigma_{R.x=v_i}, \sigma_{T.y=e} \in \mathbf{V}$, since $\sigma_{T.y=e}(D) = \{(v_j, e)\}$ and $\sigma_{R.x=v_i}(D) = \emptyset$. 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): we now show that for each edge, at least one of its endpoints is selected in R. Suppose not and 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 5.38. PRICE(H_3) *is NP-complete, where*

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

PROOF. We prove the hardness of H_3 by reduction from SET COVER. Consider a universe U and a family of subsets $\mathcal{S} \subseteq 2^U$. Then, the SET COVER problem asks for the minimum size subset of $\mathcal{S}$ that covers every element of U.

For the reduction, let us define a schema $\{R(X, Y)\}$ such that $Col_X = U$ and $Col_Y = \mathcal{S}$. Then, consider the database D where $(a, S) \in R^D$ iff $a \in S$. Notice that $H_3(D) = U$. Let us set prices $p(\sigma_{R.X=a}) = |\mathcal{S}|$ for any $a \in Col_X = U$ and $p(\sigma_{R.Y=S}) = 1$ for any $S \in Col_Y = \mathcal{S}$. We will show that U has a set cover of size k if and only if H_3 can be determined by a set of views of price k.

For the one direction, consider a set cover $\{S_1, \dots, S_k\} \subseteq \mathcal{S}$ and let $\mathbf{V} = \{\sigma_{R.Y=S_i} \mid i = 1, \dots, k\}$. We will show that $D \vdash \mathbf{V} \twoheadrightarrow H_3$ (notice that $\mathbf{V}$ is of cost k). Consider a database D' such that $\mathbf{V}(D') = \mathbf{V}(D)$ and for the sake of contradiction assume that some $a \in U$ does not belong in $H_3(D')$. By the construction of $\mathbf{V}$, some set S covers a, hence $(a, S) \in \sigma_{R.Y=S}(D)$ and $\sigma_{R.Y=S} \in \mathbf{V}$. This implies that $(a, S) \in \sigma_{R.Y=S}(D')$ and consequently $a \in H_3(D')$, a contradiction.

For the converse direction, suppose that H_3 is determined by a set of views $\mathbf{V}$ of price k. We can assume w.l.o.g. that no selection from $R.X$ belongs in $\mathbf{V}$, since in this case $k \geq |\mathcal{S}|$ and we could instead buy all the views $\sigma_{R.Y=S}$ for a cost of $|\mathcal{S}|$ and determine H_3 trivially. Hence, $\mathbf{V}$ contains views of the form $\sigma_{R.Y=S_i}$ 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' that is constructed by removing from D all tuples of the form $R(a, *)$. It is easy to see that D', D agree on the views $\mathbf{V}$, since $\mathbf{V}$ contains no view of the form $\sigma_{R.Y=S}$ such that S contains a. Moreover, $a \notin H_3(D')$, whereas $a \in H_3(D)$; this contradicts the fact that $\mathbf{V}$ determines H_3. □

PROPOSITION 5.39. *Let Q be a full query with four variables y, x_1, x_2, x_3 such that it contains the atoms $S_1(x_1, y), S_2(x_2, y), S_3(x_3, y)$ and variable $x_i, i = 1, 2, 3$ belongs also in an atom $R_i \neq S_1, S_2, S_3$ (R_x, R_y, R_z are not necessarily different). Then,* PRICE(Q) *is NP-complete.*

PROOF. The hardness for pricing H_4 is obtained when R_1, R_2, R_3 are different and contain only x_1, x_2, x_3 respectively.

We prove the hardness of H_4 by applying the same idea presented in the hardness proof of H_1, namely by showing a reduction from 3P3UHVC. Similarly to the proof of Proposition 5.36, we will present a slightly more general proof, which will prove the hardness for two more queries: H_4', where $R_1(x_1), R_2(x_2)$ is replaced by $R_1(x_1, x_2)$, and H_4'', where $R_1(x_1), R_2(x_2), R_3(x_3)$ is replaced by $R_1(x_1, x_2, x_3)$. To do this, we will assume throughout the proof that any R_i may refer to the same relation.

Denote the hypergraph by $G(A, B, C, E)$. For the reduction, 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 $R_i^D = \emptyset$ for $i = 1, 2, 3$. 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 $H_4(D) = \emptyset$. Finally, we set the prices for the selections on S_i to be zero and for the selections of R_i to be 1. We prove that G has a vertex cover of size k *if and only if* H_4 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 a vertex $v \in C$, let $\sigma(v)$ be the corresponding selection and $\sigma(C) = \bigodot_{v \in C} \sigma(v)$.

As with the proof of Proposition 5.36, it suffices to show that $C = \{v_1, \dots, v_k\}$ is a vertex cover for G iff $D \vdash \sigma(C), \Sigma_S \twoheadrightarrow Q$ (again, note that the $\sigma(C), \Sigma_S$ costs exactly k and also w.l.o.g. any views of cost k will be of this form). Applying Lemma 5.11 and since $Q(D) = \emptyset$, we obtain that $D \vdash \sigma(C), \Sigma_S \twoheadrightarrow Q$ is equivalent to the following: for every tuple $(a, b, c, m) \in Col_{x_1} \times Col_{x_2} \times Col_{x_3} \times Col_y$ such that $(a, m) \in S_1^D$, $(b, m) \in S_2^D$, $(c, m) \in S_3^D$, at least one of the views $\sigma_{R_1.x_1=a}, \sigma_{R_2.x_2=b}, \sigma_{R_3.x_3=c}$ must belong in $\sigma(C)$. However, this is equivalent to the fact that for any edge $(a, b, c) \in E$, one of its vertices is covered by C. □

5.8. Proof of the Dichotomy Theorem

In this section, we complete the proof of the dichotomy theorem. We will first prove the dichotomy theorem for *full* conjunctive queries w/o self-joins. Then, we will show how to extend the dichotomy in the case of projections.

Let us first introduce some notation for the reductions. We denote PRICE(Q_1) $\prec$ PRICE(Q_2) if we can reduce PRICE(Q_1) to PRICE(Q_2) in polynomial time. We write PRICE(Q_1) $\sim$ PRICE(Q_2) if there is a PTIME reduction in both directions, that is, PRICE(Q_1) $\prec$ PRICE(Q_2) and PRICE(Q_2) $\prec$ PRICE(Q_1).

The proof consists of several steps, which we briefly describe here.

(1) We reduce the problem to a single connected component (Proposition 5.40).
(2) We reduce the problem to CQs without constants (Proposition 5.48), multiple occurrences of a variable in the same atom (Proposition 5.41), and hanging variables (Proposition 5.44): the remaining queries are *normalized*.
(3) Applying Proposition 5.36, we show that pricing a normalized query is NP-hard if it contains a ternary predicate.
(4) We finally apply Proposition 5.39 to show that if a query is not cyclic or chain, it is NP-hard.

Throughout this section, we will often need to show a result of the following form: if $D \vdash \mathbf{V} \twoheadrightarrow Q$, then $D' \vdash \mathbf{V}' \twoheadrightarrow Q'$. In order to prove this, it suffices to define a function f from $\mathbf{V}'(D')$ to $\mathbf{V}(D)$, and a function g from $Q(D)$ to $Q'(D')$. Indeed, since $\mathbf{V}$ determines Q, there exists some function h such that $Q(D) = h(\mathbf{V}(D))$. Then, $Q'(D') = g(Q(D)) = (g \circ h)(\mathbf{V}(D)) = (g \circ h \circ f)(\mathbf{V}'(D'))$, which shows that $\mathbf{V}'$ indeed determines Q' under D'.

5.8.1. Connected Components. 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 5.40. *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. Let us assume a database D and a set of price points $\mathcal{S}$. In order to show that $\textsc{Price}(Q) \prec \textsc{Price}(Q_1, Q_2)$, we show that the following equation holds:

$$p_D^{\mathcal{S}}(Q) = \begin{cases} \min\{p_D^{\mathcal{S}}(Q_1), p_D^{\mathcal{S}}(Q_2)\} & \text{if } Q_1(D) = Q_2(D) = \emptyset, \\ p_D^{\mathcal{S}}(Q_1) & \text{if } Q_1(D) = \emptyset,\ Q_2(D) \neq \emptyset, \\ p_D^{\mathcal{S}}(Q_2) & \text{if } Q_2(D) = \emptyset,\ Q_1(D) \neq \emptyset, \\ p_D^{\mathcal{S}}(Q_1) + p_D^{\mathcal{S}}(Q_2) & \text{else.} \end{cases} \tag{9}$$

First, consider the case where $Q_i(D) = \emptyset$ for some $i = 1, 2$. Notice that $Q_i(D)$ being empty implies that $Q(D) = \emptyset$ as well. Hence, $D \vdash Q_i \twoheadrightarrow Q$ and, by the arbitrage-free property, $p_D^{\mathcal{S}}(Q) \leq p_D^{\mathcal{S}}(Q_i)$.

Next, consider the case where $Q_1(D) \neq \emptyset$. We will show that, if $D \vdash \mathbf{V} \twoheadrightarrow Q$, then $D \vdash \mathbf{V} \twoheadrightarrow Q_2$, which implies that $p_D^{\mathcal{S}}(Q) \geq p_D^{\mathcal{S}}(Q_2)$. Denote by D_1 and D_2 the databases that are contained in D and correspond to the relations that appear in Q_1 and Q_2 respectively. Note that $D_1 \cup D_2 = D$. For the sake of the contradiction, assume that $\mathbf{V}$ does not determine Q_2. This implies that exists some database D'_2 such that $Q_2(D'_2) \neq Q_2(D_2)$ and $\mathbf{V}(D_2) = \mathbf{V}(D'_2)$. Let us now construct the database $D' = D_1 \cup D'_2$. It is easy to see that $\mathbf{V}(D) = \mathbf{V}(D')$. Also, let t_2 is the tuple which witnesses the nondeterminacy of Q_2. Since $Q_1(D)$ is nonempty, there exists $t_1 \in Q_1(D)$. It is easy to see that $t_1 \circ t_2$ witnesses the nondeterminacy of Q, a contradiction. We can show the same symmetrically in the case where $Q_2(D) \neq \emptyset$.

Combining the two inequalities, we prove case 2 (and symmetrically case 3) of (9).

In order to prove case 1 (where both Q_1, Q_2 are empty), assume that we have some $\mathbf{V}$ that determines neither Q_1 nor Q_2. Then, consider the tuples t_1, t_2 that witness the nondeterminacy 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 nonempty, 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 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 $p_{D_1}^{\mathcal{S}}(Q_1) = p_D^{\mathcal{S}'}(Q)$. $\square$

5.8.2. Multiple Occurrences. 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 a variable in the same atom of Q.

PROPOSITION 5.41. *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,* $\text{PRICE}(Q) \sim \text{PRICE}(Q')$.

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 $\text{PRICE}(Q') \prec \text{PRICE}(Q)$. 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 $\text{PRICE}(Q) \prec \text{PRICE}(Q')$. 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 5.19, $\mathbf{V}$ 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 5.14 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 $Q(D') = Q(D)$. □

The proposition says that pricing the query $Q(x, y) = R(x, x), S(x, y, x)$ is equivalent to pricing the query $Q'(x, y) = R'(x), S'(y, x)$.

5.8.3. Hanging Variables. Recall that a *hanging* variable is one that occurs in only one atom of Q (and by the previous discussion, we can assume it appears only once there).

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 5.42. *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) $D \vdash (\mathbf{V} \setminus \Sigma_{R.X}) \twoheadrightarrow \mathbf{Q}$ (in other words, every view in $\mathbf{V}$ referring to $R.X$ is redundant).*

PROOF. We will assume that $R.X$ is not fully covered and then show that $\Sigma_{R.X}$ (i.e., any view of the form $\sigma_{R.X=a}$) is redundant to $\mathbf{V}$. By Theorem 5.15, it suffices to show that a tuple $t_R \in \Sigma_{R.X}(D)$ is either noncritical 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. □

Lemma 5.43. *Let Q be a query with a hanging variable $R.x$, where $R(X_1, \dots, X_k, X)$. Given a database D, define $D_C = D \cup \bigcup_{t \in R^D} M(t)$. 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 5.12, 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 $Q(D^a)$. $\square$

Proposition 5.44. *Let Q^H be the query obtained from Q if we remove all the hanging variables. Then,* Price$(Q^H) \sim$ 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 $R'(X_1, \dots, X_k)$. 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 Price$(Q) \prec$ Price(Q^{-x}). Let D be a database and $\mathcal{S}$ price points for Q. From Lemma 5.43, 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 $(a_1, \dots) \in Q^{-x}(D')$ 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

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

Note that by Lemma 5.42, 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 the price of R' is zero). Indeed, we have a function to go from $Q(D_C)$ to $Q^{-x}(D)$ 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 Price$(Q^{-x}) \prec$ 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 now have 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 $\Sigma_{R.X}$, that is, $\sum_a p(\sigma_{R.Y=a})$. The second price corresponds to the case when $R.X$ is not covered at all, and is equal to the price of Q'.

Example 5.45. 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 $\min(p_1 + p(\Sigma_{R.X}), p_2)$.

5.8.4. Constants. We next show how to deal with constants (and constraints more general). The first proposition holds for any atomic constraint that can be computed in polynomial time, that is, we allow predicates like $x > 10$ or USER-DEFINED-PREDICATE(x), but not $x < y$.

PROPOSITION 5.46. *Suppose Q contains a variable x filtered by an atomic predicate $C(x)$. Let Q' be the query obtained by removing $C(x)$ from Q. If C is computable in PTIME,* PRICE(Q) $\prec$ 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', 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.

We next show that $p^{\mathcal{S}}_D(Q) = p^{\mathcal{S}'}_{D'}(Q')$. The key observation is that if for some $\mathbf{V} \subseteq \Sigma$, $D \vdash \mathbf{V} \twoheadrightarrow Q$, we can assume w.l.o.g. 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 noncritical and Theorem 5.15). 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 $D \vdash \mathbf{V} \twoheadrightarrow Q$, it follows that $D - T^C \vdash \mathbf{V} \twoheadrightarrow Q$, where T^C are the tuples of D filtered by C (from Proposition 5.12). Now, we can take $\mathbf{V}(D') = \mathbf{V}(D - T^C)$, and from that compute $Q(D - T^C) = Q(D) = Q'(D')$. □

The following example illustrates the given construction.

Example 5.47. 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 $Q'(x, y, z) = R(y), S(y, w, z), T(z)$.

PROPOSITION 5.48. *Suppose Q has a constant a in some atom $R(a, x_1, \ldots, x_m)$. Let Q' be the query where R is replaced with $R'(x_1, \ldots, x_m)$ such that a is removed. Then,* PRICE(Q) $\sim$ PRICE(Q').

PROOF. We first show that PRICE(Q) $\prec$ 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 5.46, 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 5.44 to obtain Q'. To prove that PRICE(Q') $\prec$ PRICE(Q), we can apply the same proof as in the second part of Proposition 5.44 to reduce Q' to Q'', where we have added to R' an extra attribute x_a with column $Col_{x_a} = \{a\}$. But now adding to Q'' the constraint $x_a = a$ gives an equivalent query, which is Q. □

As an example, pricing the query $Q(x, y) = R(x, y), S(x, a)$ is equivalent to pricing the query $Q'(x, y) = R(x, y), S'(x)$.

5.8.5. The Dichotomy for Normalized Queries. At this point, it suffices to characterize the complexity for pricing a *normalized* query to complete the dichotomy proof. We remind here that a *normalized* query is a full CQ without self-joins, which has one connected component, no constants, no hanging variables and no multiple occurrences of variables in the same atom.

We first present two useful lemmas for hardness reductions. The first lemma states that, given a hard query, we can add any variable in any atom (or even introduce a new 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 5.49. $\text{PRICE}(Q^{-R.x}) \prec \text{PRICE}(Q)$.

PROOF. Without loss of generality, we can assume that x is not hanging (otherwise the lemma holds trivially by Proposition 5.44).

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 $R'(X_1, \dots, X_k, Y)$. The column for Y is $Col_Y = Col_x$. We create D' from D as follows:

$$(a_1, \dots, a_k) \in R^D \Longrightarrow \forall b \in Col_Y : (a_1, \dots, a_k, b) \in R'^{D'}.$$

If R does not exist in $Q^{-R.x}$, add a new table $R'^{D'} = Col_x$.

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 relations remain unchanged, $t \in Q^{-R.x}(D)$. For the opposite direction, let $t \in Q^{-R.x}(D)$ and consider 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 construct $\mathcal{S}'$, we let the price of any selection on $R.x$ to be equal to the price of asking for the whole R, in other words $p^{\mathcal{S}}_D(R)$, 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 $p^{\mathcal{S}'}_{D'}(Q) = p^{\mathcal{S}}_D(Q^{-R.x})$, 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 $Q^{-R.x}(D) = Q(D')$. □

The second lemma is as follows.

LEMMA 5.50. *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,* $\text{PRICE}(Q') \prec \text{PRICE}(Q)$.

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 1-1 mapping between the answers, concludes the proof. □

PROPOSITION 5.51. *If a normalized query Q contains an atom with ≥ 3 attributes,* PRICE(Q) *is NP-complete.*

PROOF. Assume that Q contains an atom $R(x, y, z, \dots)$. Since Q is normalized, the variables x, y, z will be different. Applying repeatedly Lemma 5.49, we can remove all other variables and obtain a query Q' that contains only x, y, z. Since Q is normalized, the variables x, y, z are not hanging and thus will appear in other atoms in Q'. 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. By Lemma 5.49, PRICE(Q'') $\prec$ PRICE(Q). We can now apply Proposition 5.36 to show that PRICE(Q'') is NP-complete (and hence also PRICE(Q)). □

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

PROPOSITION 5.52. *For any query of the form*

$$C_k^+(x_1, \dots, x_k) = S_{i_1}(x_{i_1}), \dots, S_{i_\ell}(x_{i_\ell}), C_k(x_1, \dots, x_k),$$

where $k > 1$ and $\ell > 0$, PRICE(C_k^+) *is NP-complete.*

PROOF. First, we apply Lemma 5.49 to keep only one unary predicate, let it be $S(x_1)$. We can reduce then Q to the query $Q'(x_1, \dots, x_k) = S(x_1), R_1(x_1, x_2), \dots, R_k(x_k, x_1)$. Next, we apply repeatedly Lemma 5.50 to remove the relations $R_3, \dots, R_k$. Indeed, applying once Lemma 5.50 reduces Q' to $Q^2(x_1, \dots, x_{k-1}) = S(x_1), R_1(x_1, x_2), \dots, R_{k-1}(x_{k-1}, x_1)$. The next application will reduce Q^2 to $Q^3(x_1, \dots, x_{k-2}) = S(x_1), R_1(x_1, x_2), \dots, R_{k-2}(x_{k-2}, x_1)$, and so on. After $k-2$ applications, we have reduced Q to the query $Q^{k-2}(x_1, x_2) = S(x_1), R_1(x_1, x_2), R_2(x_2, x_1)$. This is exactly the query H_2, for which PRICE(H_2) is NP-complete by Proposition 5.37. □

PROPOSITION 5.53. *Let Q be a 2-normalized query where a variable x appears in ≥ 3 binary predicates. Then,* PRICE(Q) *is NP-complete.*

PROOF. Let the three binary predicates be $S_1(x_1, y)$, $S_2(x_2, y)$, $S_3(x_3, y)$. Also, consider the query Q' that is obtained by removing all the other variables apart from y, x_1, x_2, x_3 and keeping these 3 occurrences of y along with a second occurrence for each variable x_i. By Lemma 5.49, it suffices to show the hardness for Q'. Let R_1, R_2, R_3 be the relations where x_1, x_2, x_3 appear respectively. We now distinguish two cases, depending on whether the variables x_i are the same or not.

—Any two x_i are equal. W.l.o.g. let us assume that $x_1 = x_2 (= x)$. Then, we can reduce Q' to the query $Q''(x, y) = S_1(x, y), S_2(x, y), R_1(x)$, which is the hard query C_2^+ (Proposition 5.52).
—All x_i are different. Then, we prove hardness by applying Proposition 5.39.

This concludes the proof of the proposition. □

Equipped with this proposition, we can now provide a syntactic characterization for 2-normalized queries and thus conclude the dichotomy proof.

PROPOSITION 5.54. *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 binary predicates,
(3) is a chain query.

PROOF. Consider the graph G that is defined by the binary atoms of Q. If we assume that condition (2) does not hold, then in G every node has degree at most 2. It is easy to see that such a graph G is either a cycle or a line (G has one connected component). In the case of a cycle, Q is either C_k (no unary predicates) or C_k^+ (at least one unary predicate). In the case where G is a line, Q is a chain query. □

5.8.6. Projections. Finally, we show how to extend the dichotomy result for CQs with projections.

LEMMA 5.55. *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 the database for which we compute the price of 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, in order to determine Q it suffices to find a valuation that makes Q true. Thus, we are looking for the cheapest set of views from Σ such that the tuples in this set evaluate Q to true. We can compute this in polynomial time: we first compute $Q^f(D)$, and then for each tuple $t \in Q^f(D)$ compute the cheapest set of views that determines every projection t_R, where R is an atom in Q. Since the tuples t_R belong in at most $arity(R)$ views, this can be computed in polynomial time. Thus, we have showed how to reduce PRICE(Q) to PRICE(Q^f).

To show the reverse reduction, we need to prove that when PRICE(Q^f) is NP-hard, so is PRICE(Q). The crucial observation is that if PRICE(Q^f) is NP-hard, then by Theorem 5.35 and the structure of the reductions in this section, it also remains NP-hard if we restrict the databases D such that $Q^f(D) = \emptyset$. Since $Q^f(D) = \emptyset$ if and only if $Q(D)$ is false, this implies that the hardness results carry over to the boolean case. □

Lemma 5.55, along with Proposition 5.38 imply the following dichotomy for CQs with projections.

THEOREM 5.56 (DICHOTOMY FOR PROJECTIONS). *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 directly from Lemma 5.55. 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, x_0 and y_0 will be connected. In other words, there will be w.l.o.g. a sequence of atoms $R_1, \dots, R_\ell$ 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 5.49, we can remove any other variable from Q to obtain a query $Q'(\bar{x}') = R(x_0, x_1), R_1(x_1, x_2), \dots, R_\ell(x_\ell, y_0)$, where $\bar{x}' \subseteq \bar{x}$, but still containing x_0. Finally, we can repeatedly apply Lemma 5.50 to obtain the query $Q''(x_0) = R(x_0, y_0)$, which is the NP-complete query H_3 (Proposition 5.38). □

6. DISCUSSION

In this section, we present several results on extensions of our model.

6.1. 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 6.1. 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, that is, $T(D) = true$ 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 $p_D(T) \leq p_D(Q)$.

6.2. Information Leakage

In this section, we explore whether learning the price of a query without revealing the query answer leaks any information about the underlying database.

PROPOSITION 6.2. *Let D be a database and $\mathcal{S}$ price points with selections and non-zero prices. Moreover, assume that there exist two nonempty views $\sigma_{R.X=a}, \sigma_{S.Y=b} \in \mathcal{S}$ for two distinct relations R, S. Given only the prices of any conjunctive query w/o self-joins, a user can determine $V(D)$ for any $V \in \mathcal{S}$.*

PROOF. Let the relations be $R(X_1, \ldots, X_k)$ and $S(Y_1, \ldots, Y_m)$. Since $\mathcal{S}$ includes at least one nonempty selection from R and S, there exist tuples $t_1 \in R^D, t_2 \in S^D$ such that $p_1 = p(R(t_1.X_1, \ldots, t_1.X_k)) < +\infty$ and $p_2 = p(S(t_2.Y_1, \ldots, t_2.Y_m)) < +\infty$. Moreover, $p(R(t_1.X_1, \ldots), S(t_2.Y_1, \ldots)) = p_1 + p_2$. Observe that if $t_1 \notin R^D$ or $t_2 \notin S^D$, then that price would be p_1 or p_2, hence strictly smaller than $p_1 + p_2$. Thus, the user can learn that $t_1 \in R^D$ and $t_2 \in S^D$. Now, consider any potential tuple $t = R(a_1, \ldots, a_k)$ such that its price is $p < +\infty$ (its price is finite if and only if there exists an attribute X_i such that $\sigma_{R.X_i=a_i} \in \mathcal{S}$). Then, we have that $p(R(a_1, \ldots, a_k), S(t_2.Y_1, \ldots)) = p + p_2$ if $t \in R$, otherwise it is p. Similarly from potential tuples from S. Hence, the user can determine $V(D)$ for any $V \in \mathcal{S}$. $\square$

The proposition tells us that under very weak assumptions a user can derive from the query prices all the query answers for the views that are for sale. In order to overcome this issue, a pricing system can choose not to reveal any prices without a purchase, or alternatively charge the buyer some amount for learning the price.

6.3. Selections on Multiple Attributes

In Section 5, we studied the case when the price points can only be selections on single attributes. A natural question is whether one can extend our results when selections on two or more attributes are used as price points: for instance, $\sigma_{R.X=a,R.Y=b}$. In this section, we will present some result in this direction.

A first observation is that adding selections on multiple attributes will make pricing at least as hard as having selections only on one attribute (indeed, pricing the multi-selections in a very high price reduces the problem to single-selection price points). However, the data complexity of pricing conjunctive queries will be still in NP.

Unfortunately, computing the price becomes intractable even for very simple queries. The next proposition shows that using selections on 3 attributes is always prohibitive.

PROPOSITION 6.3. *If we allow selections on 3 attributes as price points, pricing the query $Q(x, y, z) = R(x, y, z)$ is NP-hard.*

PROOF. The reduction is from 3-partitite 3-uniform hypergraph vertex cover (3P3UHVC). Given a hypergraph $G = (A, B, C, E)$, we construct a database D such that $R^D = \bar{E}$, that is, the complement of the graph G. The columns of X, Y, Z are A, B, C respectively. As for the price points, we price every selection on a single attribute to 1, and a selection $\sigma_{R.X=a,R.Y=b,R.Z=c}$ to 0 if $(a, b, c) \notin E$, else we price it to 1.

Now, notice that if a set of views determines Q, we can replace any 3-selection $\sigma_{R.X=a,R.Y=b,R.Z=c}$ such that $(a, b, c) \in E$ with either $\sigma_{R.X=a}$, $\sigma_{R.Y=b}$ or $\sigma_{R.Z=c}$ with the same cost of 1 and still determine Q. This holds still any of the single attribute selections cover a superset of the tuples that the 3-selection covers. Thus, we can assume w.l.o.g. that any solution consists only of single selections plus the free edges that correspond to $\bar{E}$.

It is now easy to see that any set vertex cover defines a corresponding equal cost set of 1-selections that, together with the 3-selections from $\bar{E}$ determine Q, and vice versa. □

Hence, using selections on 3 attributes makes even the simplest queries infeasible to compute. We next discuss the implications of adding selections on two attributes (2-selections) to the complexity of pricing. Even in this case, we run into intractability very soon.

PROPOSITION 6.4. *If 2-selections are allowed, pricing the query $Q(x, y, z, w) = R(x, y), S(x, z), T(x, w)$ is NP-hard.*

PROOF. The reduction is again from 3P3UHVC. Given a hypergraph $G = (A, B, C, E)$, we first give to each potential edge a unique ID (which corresponds to variable x). Then, for each edge $(a, b, c) \notin E$, with ID m, we introduce the tuples $R(m, a), S(m, b), T(m, c)$. As for the prices points $\mathcal{S}$, we price the selections on $R.y$, $S.z$, $T.w$ to 1, and the selections on $R.x$, $S.x$, $T.x$ to $|A|$, $|B|$, $|C|$ respectively. This ensures that we can assume w.l.o.g. that no selection from $R.x$, $S.x$, $T.x$ needs to be purchased. Finally, we price a 2-selection on m, v to zero if (m, v) is in D or the edge with ID m does not include vertex v; else, we price it to 1.

Let $D \vdash \mathbf{V} \twoheadrightarrow Q$. By our construction, we can assume w.l.o.g. that any set of views that $\mathbf{V}$ contains all the zero priced 2-selections and a set C of selections on $R.y$, $S.z$, $T.w$. We will show that C corresponds to a vertex cover for G. Indeed, consider an edge $(a, b, c) \in E$. Since none of the 2-selections $\sigma_{R.x=m,R.y=a}$, $\sigma_{S.x=m,S.z=b}$, $\sigma_{T.x=m,T.w=c}$ will be free, $\mathbf{V}$ must cover one of the selections $\sigma_{R.y=a}$, $\sigma_{S.z=b}$, $\sigma_{T.w=c}$, thus covering the edge.

For the other direction, consider a vertex cover C of G and a possible answer $t = (m, a, b, c)$. If $t \in Q(D)$, the views $\sigma_{R.x=m,R.y=a}$, $\sigma_{S.x=m,S.z=b}$, $\sigma_{T.x=m,T.w=c}$ will be free, and hence the tuple t will be always discovered. Otherwise, $t \notin Q(D)$. Notice that if edge m does not include any of the vertices a, b, c, then we know it since this selection is priced to zero. Hence, m is a potential edge with vertices a, b, c. Since it is not an answer, all of $(m, a), (m, b), (m, c)$ do not belong in D and $(a, b, c) \in E$. Since all edge are covered, one of the selections on a, b, c is chosen and hence we discover again that the tuple is correctly not an answer. □

Notice that the given example also suggests that we cannot apply the reductions we used for 1-selections in order to simplify the structure of a query: indeed, for the given query, y, z, w are all hanging variables, but cannot be removed.

Fortunately, there exists a large class of conjunctive queries for which pricing with 2-selections is still feasible, and this is the class of chain queries (Definition 5.23). Indeed, there is a very elegant way to modify the construction of the flow graph such that 2-selections are taken into account. For every binary relation R, instead of setting the capacity of the *tuple edges* $(w_{R.X=a}, v_{R.Y=b})$ to infinity, we simply set it to $p(\sigma_{R.X=a,R.Y=b})$. It is easy to see that the proof for the validity of the construction can be easily extended for this case.

PROPOSITION 6.5. *Even if 2-selections are allowed as price points, pricing any* ChQ *query is in PTIME.*

To sum up, pricing conjunctive queries in the presence of multi-selections as price points leads to NP-hard problems even in the case of queries that could be trivially priced in polynomial time with 1-selections. However, the tractability for pricing remains for chain queries. We conjecture that there exists a dichotomy for pricing complexity for multi-selections as well; we leave this as an open problem.

6.4. Self-Joins

The dichotomy theorem (Theorem 5.25) holds only for conjunctive queries without self-joins. In this section, we present some partial results towards extending this dichotomy for any conjunctive query. We leave open the problem of whether a dichotomy exists for this case. We start the discussion on self-joins by showing the hardness of pricing the following query.

PROPOSITION 6.6. PRICE(H_J) *is NP-complete, where*

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

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 $H_J(D) = \emptyset$. We now show that G has a vertex cover of size k if and only if H_J can be determined by a set of views with cost k.

For the one direction, assume that G has a vertex cover $C = \{v_1, \dots, v_k\}$ 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 $D \vdash \mathbf{V}^C \twoheadrightarrow H_J$. 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, $H_J(D') = \emptyset$. For the sake of contradiction, assume that $(a, b) \in H_J(D')$. 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 it 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_J 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 $\{v_1, \dots, v_k\}$ is a vertex cover for G. Indeed, suppose that it is not a vertex cover. Then, there exists an edge $(a, b) \in E$ 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 $\mathbf{V}(D) = \mathbf{V}(D')$. However, $H_J(D') = \{(a, b)\} \neq \emptyset$, which contradicts the fact that $\mathbf{V}$ determines H_J. $\square$

This proposition suggests that self-joins complicate the computation of prices: indeed, if any of the R atoms in H_J was replaced by another atom R', the price of the query would be computable in polynomial time. Furthermore, several of the reductions proved in the previous section do not hold if the query contains a self-join. For example, consider the query $Q(x, y) = R(x), R(y)$. If we blindly apply Proposition 5.40, we would infer that the price of Q is twice the price of $R(x)$ (whereas the price of Q is exactly equal to the price of $R(x)$). Indeed, we cannot split Q into two independent connected components, since there is a dependency between the views of R that we purchase for each component.

In order to further support that a dichotomy for queries with self-joins is challenging, we show next that the problem of pricing query bundles can be reduced to pricing queries with self-joins.

THEOREM 6.7. *For a bundle $\mathbf{Q} = \{Q_1, Q_2\}$, where Q_1, Q_2 are conjunctive queries, there exists a conjunctive query Q (possibly with self-joins) s.t.* PRICE($\mathbf{Q}$) $\prec$ PRICE(Q).

PROOF. Suppose we want to price $\mathbf{Q}$ for a database D and price points $\mathcal{S}$. If Q_1, Q_2 have variables $\bar{x}_1, \bar{x}_2$ respectively (we can assume w.l.o.g. that $\bar{x}_1, \bar{x}_2$ are disjoint, otherwise they can be renamed), let $Q(\bar{x}_1, \bar{x}_2) = Q_1, Q_2$. To construct the corresponding database D', let a be a new constant added to every column and for any relation $R(X_1, \dots, X_k)$, introduce a new tuple $R(a, \dots, a)$; the tuples added guarantee that the query answer is always nonempty. Finally, the price points for $\mathcal{S}'$ remain the same as in $\mathcal{S}$, with the addition that any selection on constant a is priced to zero. Let Σ_a be all the selections on constant a.

We next show that $D \vdash \mathbf{V} \twoheadrightarrow Q_1, Q_2$ if and only if $D' \vdash \mathbf{V}, \Sigma_a \twoheadrightarrow Q$; this suffices to prove the theorem, since Σ_a is free. First, notice that there exists a mapping between the answers $\mathbf{Q}(D)$ and $Q(D')$. Indeed, notice that the answers in $\mathbf{Q}(D)$ completely determine $Q(D')$ (since we know that $Q_2(D')$, $Q_1(D')$ are not empty), and vice versa. Moreover, it it easy to see that one can compute $\mathbf{V}(D)$ from $\mathbf{V}(D')$ and vice versa. $\square$

6.5. Query Bundles

In Section 5, we discussed how to price *single* queries when the price points are selections. An interesting question is whether any of our techniques to compute the price in polynomial time apply in the case of a collection of conjunctive queries $\mathbf{Q}$.

We will show that there exists a class of query bundles that generalize the notion of a chain query and can be also priced in polynomial time.

Definition 6.8. A CHQ query bundle is a set $\mathbf{Q}$ of CHQ queries, 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 $\mathbf{Q} = \{Q_1(y, z) = S(y), R(y, z), U(z), Q_2(x, y) = S(x), T(x, y), W(y)\}$ satisfies the definition (since Q_1, Q_2 share only the prefix S) and thus is a CHQ bundle. For this class of bundles, computing the price is in polynomial time.

PROPOSITION 6.9. *If the price points are selection views, pricing a* CHQ *query bundle* $\mathbf{Q}$ *is in PTIME.*

PROOF. In order to compute a price for a $\mathbf{Q}$, we construct a flow graph $G[\mathbf{Q}]$ as in the case of single CHQ queries. The construction is as follows: for each query Q in the bundle $\mathbf{Q}$, we construct separately the flow graph $G[Q]$: the construction is feasible, since Q is a CHQ query. Then, we combine the graphs $G[Q]$ for $Q \in \mathbf{Q}$ by merging nodes with the same name and then edges that have the same endpoints. For example, for our example bundle $\mathbf{Q}$, the source and target nodes will be merged; moreover, all the nodes that correspond selections on $S(X)$ will be merged. Hence, the path that goes from s to $v_{R.X=a}$ to $w_{R.X=a}$, where $a \in Col_{R.X}$, will be common for both Q_1 and Q_2.

To prove the correctness of this construction, it suffices to show that any path from $G[Q]$ will exist in $G[\mathbf{Q}]$ and also that any path in $G[\mathbf{Q}]$ exists in some flow graph $G[Q]$. The first part comes directly from our construction, since no edge is removed. As for the second part, consider any path P in $G[\mathbf{Q}]$. As we have shown in Section 5.5, this path corresponds to a sequence of tuples $t_{R_1}, \ldots, t_{R_\ell}$. It suffices to show now that the sequence of relations $R_1, \ldots, R_\ell$ corresponds to a chain query in the bundle $\mathbf{Q}$. Suppose not; then, there exists an index i, where $1 < i < \ell$ such that $R_1, \ldots, R_i$ is a prefix of a query $Q \in \mathbf{Q}$ and $R_1, \ldots, R_{i+1}$ is not a prefix of any query. This implies that there exists some query $Q' \in \mathbf{Q}$ where R_i, R_{i+1} are consecutive atoms.

Hence, we have the two queries Q, Q' that share the atom R_i. Since they both belong in a ChQ bundle, it must be that they share R_i as part of a shared prefix or suffix. Note that it can not be part of a shared prefix, since then $R_1, \ldots, R_{i+1}$ would be a prefix for Q', a contradiction. Moreover, it cannot be a shared suffix, since then Q would contain R_i, R_{i+1}. This concludes our proof. $\square$

7. RELATED WORK

In this section, we present and discuss related work on data pricing.

Previous Work. While the interaction between data management and economics has been studied in the database research community before [Dash et al. 2009; Stonebraker et al. 1996], to the best of our knowledge, this work is the first to study the problem of data pricing and is part of a wider effort to formalize and develop a flexible data marketplace. This effort was initiated with a short vision paper that we recently published [Balazinska et al. 2011], followed by Koutris et al. [2012a] and a demonstration of an implementation of our framework in Koutris et al. [2012b]. Our pricing framework was also adapted in Li and Miklau [2012], where the authors study the pricing of aggregate linear queries.

Online Data Selling. There exist several independent vendors selling data on the web (Gnip, PatientsLikeMe, AggData, Xignite). Furthermore, Amazon Cloud users can sell their S3 data for a profit [Amazon].

On the other hand, digital market services for data have recently emerged in the cloud (Azure Datamarket, Infochimps) these data marketplaces 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, the seller can set prices on APIs (modeled as selection queries) or entire datasets. Data consumers pay monthly subscriptions that enable a maximum number of queries (i.e., API calls) per month. Alternatively, a data provider can set a price for users to download the entire dataset. The Azure DataMarket uses data subscriptions with query limits: a group of records returned by a query and that can fit on a page (currently 100) is called a *transaction*. Each subscription of a data buyer is associated with a maximum number of transactions per month. Apollo Mapping (apollomapping.com) sells access to satellite imagery. The approach that we develop in this article extends these pricing methods with the ability to interpolate prices for *arbitrary queries* over a seller's database.

Pricing Information. There exists a rich literature on pricing information products (e.g., [Jain and Kannan 2002; Shapiro and Varian 1998]). We were mostly influenced by Shapiro and Varian [1998], who argue that the price of *information products* is quite different from that of *physical goods*, and propose 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). Information products offer vast economies of scale, but can also lead to devastating price wars and ruin. Traditional pricing mechanisms based on the production cost are bound to fail when applied to information products. 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.

Query Determinacy. The classic notion of determinacy (information-theoretic determinacy) was extensively studied by Nash, Segoufin and Vianu [Segoufin and Vianu 2005; Nash et al. 2007, 2010], who have investigated both the decidability question, and the subtle relationship between determinacy and rewritability. In information-theoretic determinacy, for two query bundles $\mathbf{V}$, $\mathbf{Q}$ we say that $\mathbf{V}$ *determines* $\mathbf{Q}$, in notation $\mathbf{V} \twoheadrightarrow \mathbf{Q}$, if for all $D_1, D_2, \mathbf{V}(D_1) = \mathbf{V}(D_2)$ implies $\mathbf{Q}(D_1) = \mathbf{Q}(D_2)$. It is straightforward to see that this definition also satisfies the properties of Definition 2.2. 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 nonmonotone, 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 [Segoufin and Vianu 2005; Nash et al. 2007, 2010] 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. The fact that information-theoretic determinacy is very hard to decide is an argument against using it for pricing. However, several classes of CQ queries where determinacy is well-behaved have been found: path queries [Afrati 2007], syntactic restrictions of FO and UCQ which are called packed FO and UCQ [Marx 2007] and monadic views [Nash et al. 2010]. Determinacy has also been examined in the restricted setting of aggregate queries [Grumbach and Tininini 2003].

In our article 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* [Calvanese et al. 2002] under the exact view assumption. The definition is based on the notion of *certain answers* [Abiteboul and Duschka 1998]. We note that instance-specific reasoning also arises in data security and authorization views: in that context, Zhang and Mendelzon [2005] study *conditional query containment*, where the containment is conditioned on a particular view output.

Technical Content. Finally, we should mention that, on the surface, our complexity results for pricing seem related to complexity results for computing *responsibility* [Meliou et al. 2010]. 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.

8. CONCLUSION

We have presented in this article 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 then determined automatically. We gave several results: an explicit formula for computing the price, a polynomial time algorithm for a large class of Conjunctive Queries and a dichotomy theorem for Conjunctive Queries without self-joins. We also discussed several extensions of our results. Finally, we presented several results on instance-based determinacy, which may be of independent interest.

Our work has left several open questions:

—Can the dichotomy result to PTIME or NP-complete be extended to pricing super-classes of Conjunctive Queries without self-joins: CQs with self-joins, CQ bundles, Unions of Conjunctive Queries? We have presented some steps towards this direction in Section 6, but we do not have a full dichotomy for either of these cases.

—We have shown that the data complexity for INSTANCE-BASED DETERMINACY is co-NP complete, but the combined complexity of the problem is not known (we know that is between co-NP complete and Π_2^p).

Interesting future work also includes considering data pricing in more general settings. For example, we can consider pricing with 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 direction is the interaction between pricing and privacy. Most of the literature on data privacy [Dwork 2011] 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 [Schneier 2000]. Setting a price for private data is one form of such control that we plan to investigate.

REFERENCES

S. Abiteboul and O. M. Duschka. 1998. Complexity of answering queries using materialized views. In *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*. ACM Press, 254–263.

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

F. N. Afrati. 2007. Rewriting conjunctive queries determined by views. In MFCS. 78–89.

Amazon. Using Amazon S3 Requester Pays with DevPay. docs.amazonwebservices.com/AmazonDevPay/latest/DevPayDeveloperGuide/index.html?S3RequesterPays.html.

M. Balazinska, B. Howe, and D. Suciu. 2011. Data markets in the cloud: An opportunity for the database community. http://cloud-data-pricing.cs.washington.edu/balazinska-pvldb11.pdf.

D. Calvanese, G. D. Giacomo, M. Lenzerini, and M. Y. Vardi. 2002. Lossless regular views. In *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*. L. Popa, Ed., ACM, 247–258.

T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. 2001. *Introduction to Algorithms* 2nd Ed. MIT Press and McGraw-Hill Book Company.

D. Dash, V. Kantere, and A. Ailamaki. 2009. An economic model for self-tuned cloud caching. In *Proceedings of the IEEE 25th International Conference on Data Engineering (ICDE'09)*. 1687–1693.

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

G. Gottlob and P. Senellart. 2010. Schema mapping discovery from data instances. *J. ACM* 57, 2.

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

S. Jain and P. K. Kannan. 2002. Pricing of information products on online servers: Issues, models, and analysis. *Management Sci.* 48, 9, 1123–1142.

P. Koutris, P. Upadhyaya, M. Balazinska, B. Howe, and D. Suciu. 2012a. Query-based data pricing. In *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*. M. Benedikt, M. Krötzsch, and M. Lenzerini, Eds., ACM, 167–178.

P. Koutris, P. Upadhyaya, M. Balazinska, B. Howe, and D. Suciu. 2012b. Querymarket demonstration: Pricing for online data markets. *Proc. VLDB Endow.* 5, 12, 1962–1965.

C. Li and G. Miklau. 2012. Pricing aggregate queries in a data marketplace. In *Proceedings of the International Workshop on Web and Databases*.

L. Libkin. 2004. *Elements of Finite Model Theory*. Springer.

M. Marx. 2007. Queries determined by views: pack your views. In *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*. L. Libkin, Ed., ACM, 23–30.

A. Meliou, W. Gatterbauer, K. F. Moore, and D. Suciu. 2010. The complexity of causality and responsibility for query answers and non-answers. *Proc. VLDB Endow.* 4, 1, 34–45.

A. Nash, L. Segoufin, and V. Vianu. 2007. Determinacy and rewriting of conjunctive queries using views: A progress report. In *Proceedings of the International Conference on Database Theory*. 59–73.

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

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

L. Segoufin and V. Vianu. 2005. Views and queries: determinacy and rewriting. In *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*. C. Li, Ed., ACM, 49–60.

C. Shapiro and H. R. Varian. 1998. Versioning: The smart way to sell information. *Harvard Business Rev.* 76, 106–114.

M. Stonebraker et al. 1996. Mariposa: A wide-area distributed database system. *VLDB J.* 5, 1, 048–063.

Z. Zhang and A. O. Mendelzon. 2005. Authorization views and conditional query containment. In *Proceedings of the International Conference on Database Theory*. 259–273.

Received October 2012; revised January 2014; accepted May 2015