Enumeration Complexity of Conjunctive Queries with Functional Dependencies

Nofar Carmeli[1] · Markus Kröll[2]

Abstract

We study the complexity of enumerating the answers of Conjunctive Queries (CQs) in the presence of Functional Dependencies (FDs). Our focus is on the ability to list output tuples with a constant delay in between, following a linear-time preprocessing. A known dichotomy classifies the acyclic self-join-free CQs into those that admit such enumeration, and those that do not. However, this classification no longer holds in the common case where the database exhibits dependencies among attributes. That is, some queries that are classified as hard are in fact tractable if dependencies are accounted for. We establish a generalization of the dichotomy to accommodate FDs; hence, our classification determines which combination of a CQ and a set of FDs admits constant-delay enumeration with a linear-time preprocessing. In addition, we generalize a hardness result for cyclic CQs to accommodate unary FDs, and further conclusions of our development include a dichotomy for enumeration with linear delay. Finally, we show that all our results apply also for CQs with disequalities and in the presence of cardinality dependencies that generalize FDs.

Keywords Enumeration · Complexity · Conjunctive queries · Functional dependencies

✉ Nofar Carmeli
  snofca@cs.technion.ac.il

  Markus Kröll
  kroell@dbai.tuwien.ac.at

[1] Technion, Haifa, Israel

[2] TU Wien, Vienna, Austria

1 Introduction

When evaluating a non-boolean Conjunctive Query (CQ) over a database, the number of results can be huge. Since this number may be larger than the size of the database itself, we need to use specific measures of enumeration complexity to describe the hardness of such a problem. In this perspective, the best we can hope for is to constantly output results, in such a way that the delay between them is unaffected by the size of the database instance. For this to be possible, we need to allow a precomputation phase, as linear time preprocessing is necessary to read the input instance before deciding the existence of a first result.

A known dichotomy determines when the answers to self-join-free acyclic CQs can be enumerated with constant delay after linear time preprocessing [3]. This class of enumeration problems, denoted by $DelayC_{lin}$, can be regarded as the most efficient class of nontrivial enumeration problems and therefore current work on query enumeration has focused on this class [5, 11, 18]. Bagan et al. [3] show that a subclass of acyclic queries, called *free-connex*, are exactly those that are enumerable in $DelayC_{lin}$, under the common assumption that boolean matrix multiplication cannot be solved in quadratic time. An acyclic query is called free-connex if the query remains acyclic when treating the head of the query as an additional atom. This and all other results in this paper hold under the RAM model [13].

The above mentioned dichotomy only holds when applied to databases with no additional assumptions, but oftentimes this is not the case. In practice, there is usually a connection between different attributes, and *Functional Dependencies* (FDs) and *Cardinality Dependencies* (CDs) are widely used to model situations where some attributes imply others. As the following example shows, these constraints also have an immediate effect on the complexity of enumerating answers for queries over such a schema.

Example 1 For a list of actors and the production companies they work with, consider the query

$$Q(actor, production) \leftarrow \text{Cast}(movie, actor), \text{Release}(movie, production).$$

At first glance, it appears as though this query is not in $DelayC_{lin}$, as it is acyclic but not free-connex. Nevertheless, if we take the fact that a movie has only one production company into account, we have the FD Release: $1 \rightarrow 2$, and the enumeration problem becomes easy: we only need to iterate over all tuples of Cast and replace the *movie* value with the single *production* value that the relation Release assigns to it. This can be done in linear time by first building a lookup table from Release in linear time.

Example 1 shows that the dichotomy by Bagan et al. [3] does not hold in the presence of FDs. In fact, we believe that dependencies between attributes are so common in real life that ignoring them in such dichotomies can lead to missing a significant portion of the tractable cases. Therefore, to get a realistic picture of the enumeration complexity of CQs, we have to take dependencies into account. The goal of this work is to generalize the dichotomy to fully accommodate FDs.

Towards this goal, we introduce an extension of a query Q according to the FDs. The extension is called the FD-extended query, and denoted Q^+. In this extension, each atom, as well as the head of the query, contains all variables that can be implied by its variables according to the FDs. This way, instead of classifying every combination of CQ and FDs directly, we encode the dependencies within the extended query, and use the classification of Q^+ to gain insight regarding Q. This approach draws inspiration from the proof of a dichotomy in the complexity of *deletion propagation*, in the presence of FDs [14]. However, the problem and consequently the proof techniques are fundamentally different.

The FD-extension is defined in such a way that if Q is satisfied by an assignment, then the same assignment also satisfies the extension Q^+, as the underlying instance is bound by the FDs. In fact, we can show that enumerating the solutions of Q under FDs can be reduced to enumerating the solutions of Q^+. Therefore, tractability of Q^+ ensures that Q can be efficiently solved as well. By using the positive result in the known dichotomy, Q^+ is tractable w.r.t enumeration if it is free-connex. Moreover, it can be shown that the structural restrictions of acyclicity and free-connex are closed under FD-extensions. Hence, the class of all queries Q such that Q^+ is free-connex is a proper extension of the class of free-connex queries. We denote the classes of queries Q such that Q^+ is acyclic or free-connex as FD-acyclic or FD-free-connex, respectively.

To reach a dichotomy, we now need to answer the following question: Is it possible that Q can be enumerated efficiently even if it is not FD-free-connex? To show that an enumeration problem is not within a given class, enumeration complexity has few tools to offer. One such tool is a notion of completeness for enumeration problems [10]. However, this notion focuses on problems with a complexity corresponding to higher classes of the polynomial hierarchy. So in order to deal with this problem, Bagan et al. [3] reduced the matrix multiplication problem to enumerating the answers to any query that is acyclic but not free-connex. This reduction fails, however, when dependencies are imposed on the data, as the constructed database instance does not necessarily satisfy the underlying dependencies.

As it turns out, however, the structure of the FD-extended query Q^+ allows us to extend this reduction to our setting. By carefully expanding the reduced instance such that on the one hand, the dependencies hold and on the other hand, the reduction can still be performed within linear time, we establish a dichotomy. That is, we show that the tractability of enumerating the answers of a self-join-free query Q in the presence of FDs is exactly characterized by the structure of Q^+: Given an FD-acyclic query Q, we can enumerate the answers to Q within the class $\text{D}elayC_{lin}$ iff Q is FD-free-connex.

The resulting extended dichotomy, as well as the original one, brings insight to the case of acyclic queries. Concerning unrestricted CQs, providing even a first solution of a query in linear time is impossible in general. This is due to the fact that the parameterized complexity of answering boolean CQs, taking the query size as the parameter, is W[1]-hard [17]. This does not imply, however, that there are no cyclic queries with the corresponding enumeration problems in $\text{D}elayC_{lin}$. The fact that no such queries exist requires an additional proof, which was presented by Brault-Baron [6]. This result holds under the assumption that a generalized version

of the triangle finding problem is not solvable within linear time [19]. As before, this proof does no longer apply in the presence of FDs. Moreover, it is possible for Q to be cyclic and Q^+ acyclic. In fact, Q^+ may even be free-connex, and therefore tractable in $DelayC_{lin}$. We show that, under the same assumptions used by Brault-Baron [6], the evaluation problem for a self-join-free CQ in the presence of unary FDs where Q^+ is cyclic cannot be solved in linear time. As linear time preprocessing is not enough to achieve the first result, a consequence is that enumeration within $DelayC_{lin}$ is impossible in that case. This covers all types of self-join-free CQs and shows a full dichotomy for the case of unary FDs. Moreover, we show how our results can be easily used to yield additional results, such as a dichotomy for the evaluation of CQs with linear delay.

The results we present here are not limited to CQs and FDs. CQs with disequalities are an extension of CQs, allowing to restrict the satisfying assignments and demand that some pairs of variables map to different values. We prove that our results apply to the more general query class: CQs with disequalities.

Another way of generalizing our results is not to extend the class of queries, but the class of dependencies. Cardinality Dependencies (CDs) [2, 7] are a generalization of FDs, denoted $(R_i : A \rightarrow B, c)$. Here, the right-hand side does not have to be unique for every assignment to the left-hand side, but there can be at most c different values to the variables of B for every value of the variables of A. FDs are in fact a special case of CDs where $c = 1$. Constraints of that form appear naturally in many applications. For example: a movie has only a handful of directors, there are at most 200 countries, and a person is typically limited to at most 5000 friends in (some) social networks. We show that all results described in this paper also apply to CDs.

Contributions Our main contributions are as follows.

- We extend the class of queries that are known to be in $DelayC_{lin}$ by incorporating FDs. This extension is the class of FD-free-connex CQs.
- We establish a dichotomy for the enumeration complexity of self-join-free FD-acyclic CQs. Consequently, we get a dichotomy for self-join-free acyclic CQs in the presence of FDs.
- We show a lower bound for FD-cyclic CQs. In particular, we get a dichotomy for all self-join-free CQs in the presence of unary FDs with respect to $DelayC_{lin}$, and a similar dichotomy regarding enumeration with linear delay.
- We show the extension of all our results to CDs and CQs with disequalities.

A short version of this article appeared in the 21st International Conference on Database Theory (ICDT18) [9]. This article has the following additions. First, it includes all proofs omitted from the short version. Second, it includes a new section, Section 7, proving that all results in this paper hold also for CQs with disequalities. In addition, we add more thorough discussions both on the computational model and on the complexity assumption used in Section 5.

This work is organized as follows: in Section 2 we provide definitions and state previous results that we will use. Section 3 introduces the notion of FD-extended queries and establishes the equivalence between a query and its FD-extension. The dichotomy for acyclic CQs is shown in Section 4, while Section 5 shows a lower

bound for cyclic queries under unary FDs. Section 6 shows that all results from the previous sections extend to CDs, and Section 7 does the same for CQs with disequalities. Concluding remarks are given in Section 8.

2 Preliminaries

In this section we provide preliminary definitions as well as state results that we will use throughout this paper.

2.1 Schemas and Functional Dependencies

A *schema* $\mathcal{S}$ is a pair $(\mathcal{R}, \Delta)$ where $\mathcal{R}$ is a finite set $\{R_1, \ldots, R_n\}$ of *relational symbols* and Δ is a set of *Functional Dependencies* (FDs). We denote the *arity* of a relational symbol R_i as $\text{arity}(R_i)$. An FD $\delta \in \Delta$ has the form $R_i \colon A \to B$, where $R_i \in \mathcal{R}$ and A, B are non-empty with $A, B \subseteq \{1, \ldots, \text{arity}(R_i)\}$.

Let *dom* be a finite set of constants. A database I over schema $\mathcal{S}$ is called an *instance* of $\mathcal{S}$, and it consists of a finite relation $R_i^I \subseteq dom^{\text{arity}(R_i)}$ for every relational symbol $R_i \in \mathcal{R}$, such that all FDs in Δ are *satisfied*. An FD $\delta = R_i \colon A \to B$ is said to be satisfied if, for all tuples $u, v \in R_i^I$ that are equal on the indices of A, u and v are equal on the indices of B. In what follows, we will often assume w.l.o.g. that all FDs are of the form $R_i \colon A \to b$, where $b \in \{1, \ldots, \text{arity}(R_i)\}$, as we can replace an FD of the form $R_i \colon A \to B$ where $|B| > 1$ by the set of FDs $\{R_i \colon A \to b \mid b \in B\}$. If $|A| = 1$, we say that δ is a *unary* FD.

2.2 Conjunctive Queries

Let *var* be a set of variables disjoint from *dom*. A *Conjunctive Query* (CQ) over schema $\mathcal{S} = (\mathcal{R}, \Delta)$ is an expression of the form $Q(\mathbf{p}) \leftarrow R_1(\mathbf{v}_1), \ldots, R_m(\mathbf{v}_m)$, where $R_1, \ldots, R_m$ are relational symbols of $\mathcal{R}$, the tuples $\mathbf{p}, \mathbf{v}_1, \ldots, \mathbf{v}_m$ hold variables, and every variable in $\mathbf{p}$ appears in at least one of $\mathbf{v}_1, \ldots, \mathbf{v}_m$. We often denote this query as $Q(\mathbf{p})$ or even Q. Define the variables of Q as $var(Q) = \bigcup_{i=1}^{m} \mathbf{v}_i$, and define the *free variables* of Q as $free(Q) = \mathbf{p}$. We call $Q(\mathbf{p})$ the head of Q, and the atomic formulas $R_i(\mathbf{v}_i)$ are called *atoms*. We further use atoms(Q) to denote the set of atoms of Q. A CQ is said to contain *self-joins* if some relation symbol appears in more than one atom.

For the *evaluation* $Q(I)$ of a CQ Q with free variables $\mathbf{p}$ over a database I, we define $Q(I)$ to be the set of all *mappings* $\mu|_{\mathbf{p}}$ such that μ is a homomorphism from $R_1(\mathbf{v}_1), \ldots, R_m(\mathbf{v}_m)$ into I, where $\mu|_{\mathbf{p}}$ denotes the restriction (or projection) of μ to the variables $\mathbf{p}$. The problem $\text{DECIDE}_\Delta \langle Q \rangle$ is, given a database instance I over a schema $(\mathcal{R}, \Delta)$, determining whether such a mapping exists.

Let $R(\mathbf{v})$ be an atom of a CQ. We say that a tuple $t \in R^I$ assigns a variable x with the value c if for every index i such that $\mathbf{v}[i] = x$ we have that $t[i] = c$. We say that a tuple $t_a \in R_a^I$ agrees with a tuple $t_b \in R_b^I$ on the value of a variable x if for every vectors $\mathbf{v_a}, \mathbf{v_b}$ such that $R_a(\mathbf{v_a})$ and $R_b(\mathbf{v_b})$ are atoms of the CQ, and for every pair of indices i_a, i_b such that $\mathbf{v_a}[i_a] = \mathbf{v_b}[i_b] = x$, we have that $t_a[i_a] = t_b[i_b]$. Given a

query Q over a schema $\mathcal{S} = (\mathcal{R}, \Delta)$, we often identify an FD $\delta \in \Delta$ as a mapping between variables. That is, if δ has the form $R_i : A \to b$ for $A = \{a_1, \ldots, a_{|A|}\}$, we sometimes denote it by $R_i : \{\mathbf{v}_i[a_1], \ldots, \mathbf{v}_i[a_{|A|}]\} \to \mathbf{v}_i[b]$, where $\mathbf{u}[k]$ is the k-th variable of $\mathbf{u}$. To distinguish between these two representations, we usually denote subsets of integers by $A, B, C, \ldots$, integers by $a, b, c, \ldots$, and variables by letters from the end of the alphabet.

2.3 Hypergraphs

A *hypergraph* $\mathcal{H} = (V, E)$ is a pair consisting of a set V of *vertices*, and a set E of non-empty subsets of V called *hyperedges* (sometimes *edges*). A *join tree* of a hypergraph $\mathcal{H} = (V, E)$ is a tree T where the nodes are the hyperedges of $\mathcal{H}$, and the *running intersection* property holds: for all $u \in V$ the set $\{e \in E \mid u \in e\}$ forms a connected subtree in T. A hypergraph $\mathcal{H}$ is said to be *acyclic* if there exists a join tree for $\mathcal{H}$. Two vertices in a hypergraph are said to be *neighbors* if they appear in the same edge. A *clique* of a hypergraph is a set of vertices, which are pairwise neighbors in $\mathcal{H}$. A hypergraph $\mathcal{H}$ is said to be *conformal* if every clique of $\mathcal{H}$ is contained in some edge of $\mathcal{H}$. A *chordless cycle* of $\mathcal{H}$ is a tuple $(x_1, \ldots, x_n), n \geq 3$, such that the set of neighboring pairs of variables of $\{x_1, \ldots, x_n\}$ is exactly $\{\{x_i, x_{i+1}\} \mid 1 \leq i \leq n - 1\} \cup \{\{x_n, x_1\}\}$. It is well known (see [4]) that a hypergraph is acyclic iff it is conformal and contains no chordless cycles.

A *pseudo-minor* of a hypergraph $\mathcal{H} = (V, E)$ is a hypergraph obtained from $\mathcal{H}$ by a finite series of the following operations:

1. *vertex removal*: removing a vertex from V and from all edges in E that contain it.
2. *edge removal*: removing an edge e from E provided that some other $e' \in E$ contains it.
3. *edge contraction*: replacing all occurrences of a vertex v (within every edge) with a vertex u, provided that u and v are neighbors.

2.4 Classes of CQs

To a CQ Q we associate a hypergraph $\mathcal{H}(Q) = (V, E)$ where the vertices V are the variables of Q and every hyperedge E is a set of variables occurring in a single atom of Q, that is $E = \{\{v_1, \ldots, v_n\}\} \mid R_i(v_1, \ldots, v_n) \in \text{atoms}(Q)\}$. With a slight abuse of notation, we also identify atoms of Q with edges of $\mathcal{H}(Q)$. A CQ Q is said to be *acyclic* if $\mathcal{H}(Q)$ is acyclic, and it is said to be *free-connex* if both Q and $(V, E \cup \{\text{free}(Q)\})$ are acyclic.

A *head-path* for a CQ Q is a sequence of variables $(x, z_1, \ldots, z_k, y)$ with $k \geq 1$, such that:

1. $\{x, y\} \subseteq \text{free}(Q)$
2. $\{z_1, \ldots, z_k\} \subseteq V \setminus \text{free}(Q)$
3. It is a *chordless path* in $\mathcal{H}(Q)$, that is, two succeeding variables appear together in some atom, and no two non-succeeding variables appear together in an atom.

For a head-path $P = (z_0, z_1, \ldots, z_k, z_{k+1})$, the *length* of P is defined as $k + 1$. Bagan et al. [3] showed that an acyclic CQ has a head-path iff it is not free-connex.

2.5 Computational Model

In this work, we only consider finite structures. We thus always assume that an input w is a string over a (fixed) finite alphabet, and we denote by $||w||$ the size of w, which is the length of the string. Using data complexity for most of our problems, the input is measured only by the size of the database instance I. Let I be a database over a schema $\mathcal{S} = (\mathcal{R}, \Delta)$. Flum et al. describe a reasonable encoding $||I||$ of the database as a string over a finite alphabet [12] with $||R^I|| = \text{arity}(R)|R^I|$ and $||I|| = 1 + |dom| + |\mathcal{R}| + \sum_{R \in \mathcal{R}} ||R^I||$. When we say linear time, we mean that the number of operations is $\mathcal{O}(||I||)$.

In this paper we adopt the *Random Access Machine* (RAM) model with uniform cost measure. For an input of size n, every register is of length $\mathcal{O}(\log(n))$. Operations such as addition of the values of two registers or concatenation can be performed in constant time. In contrast to the Turing model of computation, the RAM model with uniform cost measure can retrieve the content of any register via its unique address in constant time. This enables the construction of large lookup tables that can be queried in constant time.

We use a variant of the RAM model named DRAM [13], where the values stored in registers are at most n^c for some fixed integer c. As a consequence, the amount of available memory is polynomial in n. Some previous work mentioned in this paper use a more restrictive variant of the model, called a DLINRAM [13], where the values are at most $cn/\log(n)$, and therefore the available memory is linear in the size of the input. All results presented in this article except for the positive results for cardinality dependencies (that use Lemma 7) also apply with the DLIN-RAM model. The lower bounds shown on the DRAM model imply the same lower bounds for the more restrictive DLINRAM model. The positive results in the presence of FDs use the reduction described in Theorem 2. There, the set of keys is a projected subset of the input relations, so the memory used is bound by the size of the input.

Grandjean proved that sorting strings in the DLINRAM model can be done in time $\mathcal{O}(n/\log n)$, where n is the size of the input containing strings encoded in some fixed alphabet and separated by some special symbol [13]. We can use this method to sort relations. To construct the input to the sorting algorithm, we first translate the values from dom to a possibly smaller domain dom_R, containing only the values that appear in R^I. Note that $|dom_R| \leq ||R^I||$. Then, we translate these values to binary (since we are required to use a fixed alphabet), where each value takes $\log(dom_R)$ bits. The size of the input to the sorting problem is $n = ||R^I|| \cdot \log(|dom_R|) + (|R^I| - 1)$. Therefore, we can sort the tuples of a relation in time $\mathcal{O}(n/\log n) = \mathcal{O}(||R^I||)$. That is, it is possible to sort relations within linear time [18].

2.6 Enumeration Complexity

Given a finite alphabet Σ and binary relation $R \subseteq \Sigma^* \times \Sigma^*$, we denote by $\text{ENUM}\langle R \rangle$ the *enumeration problem* of given an instance $x \in \Sigma^*$, to output all $y \in \Sigma^*$ such that $(x, y) \in R$. For a class $\mathcal{C}$ of enumeration problems, we say that $\text{ENUM}\langle R \rangle \in \mathcal{C}$, if there is a RAM that, on input $x \in \Sigma^*$, outputs all $y \in \Sigma^*$ with $(x, y) \in R$ without

repetition such that the first output is computed in time $p(|x|)$ and the delay between any two consecutive outputs after the first is $d(|x|)$, where:

- For $\mathcal{C} = \text{D}elayC_{lin}$, we have $p(|x|) \in O(|x|)$ and $d(|x|) \in O(1)$.
- For $\mathcal{C} = \text{D}elayLin$, we have $p(|x|), d(|x|) \in O(|x|)$.

Let $\text{ENUM}\langle R_1 \rangle$ and $\text{ENUM}\langle R_2 \rangle$ be enumeration problems. There is an *exact reduction* from $\text{ENUM}\langle R_1 \rangle$ to $\text{ENUM}\langle R_2 \rangle$, denoted $\text{ENUM}\langle R_1 \rangle \leq_e \text{ENUM}\langle R_2 \rangle$, if there are mappings σ and τ such that the following conditions hold.

- $\{\tau(y) \mid y \in \Sigma^* \text{ with } (\sigma(x), y) \in R_2\} = \{y' \in \Sigma^* \mid (x, y') \in R_1\}$ in multiset notation.
- for every $x \in \Sigma^*$, the mapping $\sigma(x)$ is computable in $\mathcal{O}(|x|)$;
- for every $y \in \Sigma^*$ with $(\sigma(x), y) \in R_2$, $\tau(y)$ is computable in constant time;

Intuitively, σ is used to map instances of $\text{ENUM}\langle R_1 \rangle$ to instances of $\text{ENUM}\langle R_2 \rangle$, and τ is used to map solutions to $\text{ENUM}\langle R_2 \rangle$ to solutions of $\text{ENUM}\langle R_1 \rangle$. The notation $\text{ENUM}\langle R_1 \rangle \equiv_e \text{ENUM}\langle R_2 \rangle$ means that $\text{ENUM}\langle R_1 \rangle \leq_e \text{ENUM}\langle R_2 \rangle$ and $\text{ENUM}\langle R_2 \rangle \leq_e \text{ENUM}\langle R_1 \rangle$. An enumeration class $\mathcal{C}$ is said to be *closed under exact reduction* if for every $\text{ENUM}\langle R_1 \rangle$ and $\text{ENUM}\langle R_2 \rangle$ with $\text{ENUM}\langle R_2 \rangle \in \mathcal{C}$ and $\text{ENUM}\langle R_1 \rangle \leq_e \text{ENUM}\langle R_2 \rangle$, we have that $\text{ENUM}\langle R_1 \rangle \in \mathcal{C}$. Bagan et al. [3] proved that $\text{D}elayC_{lin}$ is closed under exact reduction. The same proof holds for any meaningful enumeration complexity class that guarantees generating all unique answers with at least linear preprocessing time and at least constant delay between answers.

2.7 Enumerating Answers to CQs

For a CQ Q over a schema $\mathcal{S} = (\mathcal{R}, \Delta)$, we denote by $\text{ENUM}_\Delta\langle Q \rangle$ the enumeration problem $\text{ENUM}\langle R \rangle$, where R is the binary relation between instances I over $\mathcal{S}$ and sets of mappings $Q(I)$. We consider the size of the query as well as the size of the schema to be fixed. Bagan et al. [3] showed that a self-join-free acyclic CQ is in $\text{D}elayC_{lin}$ iff it is free-connex:

Theorem 1 [3] *Let Q be an acyclic CQ without self-joins over a schema $\mathcal{S} = (\mathcal{R}, \emptyset)$.*

1. *If Q is free-connex, then $\text{ENUM}_\emptyset\langle Q \rangle \in DelayC_{lin}$.*
2. *If Q is not free-connex, then $\text{ENUM}_\emptyset\langle Q \rangle \notin DelayC_{lin}$, assuming the product of two $n \times n$ boolean matrices cannot be computed in time $\mathcal{O}(n^2)$.*

3 FD-Extended CQs

In this section, we formally define the extended query Q^+. Intuitively, the extension is based on treating the FDs as dependencies between variables. Assume that a variable x implies a variable y via an FD $R: x \to y$. Given an assignment for x in some answer to the query, we know the unique assignment of y according to R. Therefore, we can add the y value to every tuple that has an x value. Note however that when we extend the relations with these known values, the data still conforms to the FDs.

Thus, the process of extension does not remove the FDs. Instead, it ensures that the extended query holds in its structure the added value that the FDs provide. In Definition 1 we formalize the extension of the schema and the query, and in Theorem 2 we formalize the extension of the instance.

We also discuss the relationship between Q and Q^+: their equivalence w.r.t. enumeration and the possible structural differences between them. As a result, we obtain that if Q^+ is in a class of queries that allows for tractable enumeration, then Q is tractable as well.

The *extension* of an atom $R(\mathbf{v})$ according to an FD $S: A \to b$ and an atom $S(\mathbf{u})$ is possible if $\mathbf{u}[A] \subseteq \mathbf{v}$ but $\mathbf{u}[b] \notin \mathbf{v}$. In this case, $\mathbf{u}[b]$ is added to the variables of R. The *FD-extension* of a query is defined by iteratively extending all atoms as well as the head according to every possible dependency in the schema, until a fixpoint is reached. The schema extends accordingly: the arities of the relations increase as their corresponding atoms extend, and the FDs apply in every relation that contains all relevant variables. Dummy variables are added to adjust to the change in arity in case of self-joins.

Definition 1 ((FD-Extended Query)) Let $Q(\mathbf{p}) \leftarrow R_1(\mathbf{v_1}), \ldots, R_m(\mathbf{v_m})$ be a CQ over a schema $\mathcal{S} = (\mathcal{R}, \Delta)$. We define two types of extension steps:

- The extension of an atom $R_i(\mathbf{v_i})$ according to an FD $R_j: A \to b$.
  Prerequisites: $\mathbf{v_j}[A] \subseteq \mathbf{v_i}$ and $\mathbf{v_j}[b] \notin \mathbf{v_i}$.
  Effect: The arity of R_i increases by one, $R_i(\mathbf{v_i})$ is replaced by $R_i(\mathbf{v_i}, \mathbf{v_j}[b])$. In addition, every $R_k(\mathbf{v_k})$ such that $R_k = R_i$ and $k \neq i$ is replaced with $R_k(\mathbf{v_k}, t_k)$, where t_k is a fresh variable in every such step.
- The extension of the head $Q(\mathbf{p})$ according to an FD $R_j: A \to b$.
  Prerequisites: $\mathbf{v_j}[A] \subseteq \mathbf{p}$ and $\mathbf{v_j}[b] \notin \mathbf{p}$.
  Effect: The head is replaced by $Q(\mathbf{p}, \mathbf{v_j}[b])$.

The *FD-extension* of Q is the query $Q^+(\mathbf{q}) \leftarrow R_1^+(\mathbf{u_m}), \ldots, R_m^+(\mathbf{u_m})$, obtained by performing all possible extension steps on Q according to FDs of Δ until a fixpoint is reached. The extension is defined over the schema $\mathcal{S}^+ = (\mathcal{R}^+, \Delta_{Q^+})$, where $\mathcal{R}^+$ is $\mathcal{R}$ with the extended arities, and Δ_{Q^+} is

$$\{R_i^+: C \to d \mid \exists (R_j: A \to b) \in \Delta, \exists R_i^+(\mathbf{u_i}) \in \mathrm{atoms}(Q^+), \exists R_j(\mathbf{v_j}) \in \mathrm{atoms}(Q),$$
$$\text{s.t. } \mathbf{u_i}[C] = \mathbf{v_j}[A] \text{ and } \mathbf{u_i}[d] = \mathbf{v_j}[b]\}.$$

Given a query, its FD-extension is unique up to a permutation of the added variables and renaming of the new variables. As the order of the variables and the naming make no difference w.r.t. enumeration, we can treat the FD-extension as unique.

Example 2 Consider a schema with $\Delta = \{R_1: 1 \to 2, R_3: 2, 3 \to 1\}$, and the query $Q(x) \leftarrow R_1(x, y), R_2(x, z), R_2(u, z), R_3(w, y, z)$. As the FDs are $x \to y$ and $yz \to w$, the FD-extension is

$$Q^+(x, y) \leftarrow R_1^+(x, y), R_2^+(x, z, y, w), R_2^+(u, z, t_1, t_2), R_3^+(w, y, z).$$

We first apply $x \rightarrow y$ on the head, and then $x \rightarrow y$ and consequently $yz \rightarrow w$ on $R_2(x, z)$. These two FDs are now in the schema also for R_2, and the FDs of the extension are $\Delta_{Q^+} = \{R_1^+ : 1 \rightarrow 2, R_2^+ : 1 \rightarrow 3, R_2^+ : 3, 2 \rightarrow 4, R_3^+ : 2, 3 \rightarrow 1\}$.

We later show that the enumeration complexity of a CQ Q over a schema with FDs only depends on the structure of Q^+, which is implicitly given by Q and its schema. Therefore, we introduce the notions of acyclic and free-connex queries for FD-extensions:

Definition 2 Let Q be a CQ over a schema $\mathcal{S} = (\mathcal{R}, \Delta)$, and let Q^+ be its FD-extension.

-   We say that Q is *FD-acyclic*, if Q^+ is acyclic.
-   We say that Q is *FD-free-connex*, if Q^+ is free-connex.
-   We say that Q is *FD-cyclic*, if Q^+ is cyclic.

The following proposition shows that the classes of acyclic queries and free-connex queries are both closed under constructing FD-extensions.

Proposition 1 *Let Q be a CQ over a schema $\mathcal{S} = (\mathcal{R}, \Delta)$.*

-   *If the query Q is acyclic, then it is FD-acyclic.*
-   *If the query Q is free-connex, then it is FD-free-connex.*

Proof We prove that if Q is acyclic, then Q^+ is also acyclic, by constructing a join-tree of $\mathcal{H}(Q^+)$ given one of $\mathcal{H}(Q)$. The same proof can be applied to a join tree containing the head to show that if Q is free-connex, then so is Q^+. Denote by $Q = Q_0, Q_1, \ldots, Q_n = Q^+$ a sequence of queries such that Q_{i+1} is the result of extending all possible relations of Q_i according to a single FD $\delta \in \Delta$. By induction, it suffices to show that if $\mathcal{H}(Q_i)$ has a join tree, then $\mathcal{H}(Q_{i+1})$ has one too. So consider an acyclic $Q_i(\mathbf{p}) \leftarrow R_1(\mathbf{v}_1), \ldots, R_m(\mathbf{v}_m)$ extended to $Q_{i+1}(\mathbf{q}) \leftarrow R_1(\mathbf{u}_1), \ldots, R_m(\mathbf{u}_m)$ according to the FD $\delta = R_j : \mathbf{x} \rightarrow y$, and let $T_i = (V_i, E_i)$ be a join tree of $\mathcal{H}(Q_i)$. We claim that the same tree (but with the extended atoms), is a join tree for Q_{i+1}. Formally, define $T_{i+1} = (V_{i+1}, E_{i+1})$ such that $V_{i+1} = \{R_k(\mathbf{u}_k) \mid 1 \leq k \leq m\}$ and $E_{i+1} = \{(R_k(\mathbf{u}_k), R_l(\mathbf{u}_l)) \mid (R_k(\mathbf{v}_k), R_l(\mathbf{v}_l)) \in E_i\}$. Next we show that the running intersection property holds in T_{i+1}, and therefore it is a join tree of $\mathcal{H}(Q_{i+1})$.

Every new variable introduced in the extension appears only in one atom, so the subtree of T_{i+1} containing such a variable contains one node and is trivially connected. For any other variable $w \neq y$, the attribute w appears in the same atoms in Q and Q^+. Therefore, the subgraph of T_{i+1} containing w is isomorphic to the subgraph of T_i containing w, and since T_i is a join tree, it is connected. It is left to show that the subtree of T_{i+1} containing y is connected. Since R_j is an atom in Q containing δ, it corresponds to vertices in T_i and T_{i+1} containing $\mathbf{x} \cup \{y\}$. Let R_k be some vertex in T_{i+1} containing y. We will show that all vertices $S_1, \ldots, S_r$ on the path between R_k and R_j contain y. If y appears in the vertex R_k in T_i, then it also appears in

$S_1, \ldots, S_r$ since T_i is a join tree. Since the extension doesn't remove occurrences of variables, y appears in these vertices in T_{i+1} as well. Otherwise, y was added to R_k via δ, so R_k contains $\mathbf{x}$. Since T_i is a join tree, the vertices $S_1, \ldots, S_r$ all contain the variables $\mathbf{x}$. Thus by the definition of Q_{i+1}, y is added to each of $S_1, \ldots, S_r$ (if it was not already there) in T_{i+1}. Thus also the subtree of T_{i+1} containing y is connected. Therefore T_{i+1} is indeed a join tree. □

Example 1 shows that the converse of the proposition above does not hold. This means that, by Theorem 1, there are queries Q such that evaluating Q^+ is in $DelayC_{lin}$, but evaluating Q cannot be done with the same complexity if we do not assume the FDs. The following theorem shows that, when relying on the FDs, evaluating Q^+ is equally hard to evaluating Q.

Theorem 2 *Let Q be a CQ over a schema $\mathcal{S} = (\mathcal{R}, \Delta)$, and let Q^+ be its FD-extended query. Then, $\mathrm{ENUM}_\Delta \langle Q \rangle \equiv_e \mathrm{ENUM}_{\Delta_{Q^+}} \langle Q^+ \rangle$.*

Proof Consider a query $Q(\mathbf{p}) \leftarrow R_1(\mathbf{v}_1), \ldots, R_m(\mathbf{v}_m)$ and its FD-extension $Q^+(\mathbf{q}) \leftarrow R_1^+(\mathbf{u}_1), \ldots, R_m^+(\mathbf{u}_m)$. We show the two parts of the equivalence. □

Claim 1 $\mathrm{ENUM}_\Delta \langle Q \rangle \leq_e \mathrm{ENUM}_{\Delta_{Q^+}} \langle Q^+ \rangle$.

Construction Given an instance I for $\mathrm{ENUM}_\Delta \langle Q \rangle$, we construct an instance $\sigma(I)$ for $\mathrm{ENUM}_{\Delta_{Q^+}} \langle Q^+ \rangle$ with two phases: cleaning and extension. In the cleaning phase, we remove tuples that interfere with the extended dependencies. For every dependency $\delta = R_j \colon X \to y$ and every atom $R_k(\mathbf{v}_k)$ that contains the corresponding variables (i.e., $X \cup \{y\} \subseteq \mathbf{v}_k$), we correct R_k according to δ: we only keep tuples of R_k^I that agree with some tuple of R_j^I over the values of $X \cup \{y\}$. We denote the cleaned instance by I_0. The cleaning phase can be done in linear time by first sorting both R_j^I and R_k^I according to $X \cup \{y\}$, and then performing one scan over both of them. Next, we perform the extension phase. We follow the extension of the schema as described in Definition 1 and extend the instance accordingly. This phase results in a sequence of instances $I_0, I_1, \ldots, I_n = \sigma(I)$ that correspond to a sequence of queries $Q = Q_0, Q_1, \ldots, Q_n = Q^+$ such that each query is the result of extending an atom or the head of the previous query according to an FD. If in step i the head was extended, we set $I_{i+1} = I_i$. Now assume some relation R_k is extended according to some FD $R_j \colon X \to y$. For each tuple $t \in R_k^{I_i}$, if there is no tuple $s \in R_j^{I_i}$ that agrees with t over the values of X, then we remove t altogether. Otherwise, we copy t to $R_k^{I_{i+1}}$ and assign y with the same value that s assigns it. The extension phase takes linear time for each step. Since the number of FDs is constant in data complexity, the overall construction takes linear time. Note that this construction ensures that the extended dependencies hold in $\sigma(I)$. Given an answer $\mu|_{\mathrm{free}(Q^+)} \in Q^+(\sigma(I))$, we set $\tau(\mu) = \mu|_{\mathrm{free}(Q)}$. This projection only requires constant time.

Correctness We now show that $Q(I) = \{\mu|_{\mathrm{free}(Q)} : \mu|_{\mathrm{free}(Q^+)} \in Q^+(I^+)\}$. First, if $\mu|_{\mathrm{free}(Q^+)}$ is an answer of $Q^+(I^+)$, then μ is a homomorphism from Q^+ to

I^+. Since all tuples of I^+ appear (perhaps projected) in I, then μ is also a homomorphism from Q to I, and $\mu|_{\text{free}(Q)} \in Q(I)$. It is left to show the opposite direction: if $\mu|_{\text{free}(Q)} \in Q(I)$ then $\mu|_{\text{free}(Q^+)} \in Q^+(I^+)$. We show by induction on $Q = Q_0, Q_1, \ldots, Q_n = Q^+$ that $\mu|_{\text{free}(Q_i)} \in Q_i(I_i)$. The induction base holds since in the cleaning phase we did not remove "useful" tuples. Since $\mu|_{\text{free}(Q)} \in Q(I)$, there exist tuples, one of each relation of the query, that agree on the values of $X \cup \{y\}$ (they all assign them with the values μ assigns them). These tuples were not removed in the cleaning phase, and therefore $\mu|_{\text{free}(Q)} \in Q(I_0)$. Next assume that $\mu|_{\text{free}(Q_i)} \in Q_i(I_i)$, and we want to show that $\mu|_{\text{free}(Q_{i+1})} \in Q_{i+1}(I_{i+1})$. This claim is trivial in case the head was extended. Note also that there cannot be two distinct answers $\mu|_{\text{free}(Q_{i+1})}$ and $\mu'|_{\text{free}(Q_{i+1})}$ in $Q_{i+1}(I_{i+1})$ such that $\mu|_{\text{free}(Q_i)} = \mu'|_{\text{free}(Q_i)}$, as the added variable is bound by the FD to have only one possible value. Now consider the case where an atom $R_k(\mathbf{v}_k)$ was extended according to an FD $R_j \colon X \to y$ since $X \subseteq \mathbf{v}_k$. The tuple $\mu(\mathbf{v}_k) \in R_k^{I_i}$ was extended with the value $\mu(y)$ due to the tuple $\mu(\mathbf{v}_j) \in R_j^{I_i}$ that agrees with it on the values of X, and so $\mu(\mathbf{v}_k, y) \in R_k^{I_{i+1}}$. In case of self-joins, other atoms with the relation R_k are extended with a new and distinct variable. Such variables will be mapped to this value $\mu(y)$ as well. Overall, we have that μ (extended by mappings of the fresh variables) is also a homomorphism in $Q_{i+1}(I_{i+1})$.

Claim 2 $\text{ENUM}_{\Delta_{Q^+}}\langle Q^+ \rangle \leq_e \text{ENUM}_\Delta\langle Q \rangle$.

Construction Given an instance I^+ for $\text{ENUM}_{\Delta_{Q^+}}\langle Q^+ \rangle$, we construct an instance $\sigma(I^+)$ for $\text{ENUM}_\Delta\langle Q \rangle$ with three phases: cleaning, building a lookup table and projection. In the cleaning phase, we remove tuples that do not contribute to the answer set of Q^+ in order to prevent additional answers from appearing in Q after the projection. This can be seen as unifying the restrictions of different FDs in Δ_{Q^+} that originate in the same FD in Δ. For every FD $R_j \colon X \to y$ in Δ and every atom $R_k^+(\mathbf{u}_k)$ such that $X \cup \{y\} \subseteq \mathbf{u}_k$, we remove all tuples $t \in R_k^{+I^+}$ that agree with some tuple $s \in R_j^{+I^+}$ over X but disagree with s over y. The cleaning phase can be done in linear time by first sorting both $R_k^{+I^+}$ and $R_j^{+I^+}$ according to X. Next, we construct a lookup table T to later reconstruct the assignments to $\text{free}(Q^+) \setminus \text{free}(Q)$. For every $y \in \text{free}(Q^+) \setminus \text{free}(Q)$ added to the head due to an FD $R_j \colon X \to y$, denote by $\mathbf{x}$ a vector containing the variables of X in lexicographic order. For every tuple in $R_j^{+I^+}$ that assigns y and $\mathbf{x}$ with the values y_0 and $\mathbf{x}_0$ respectively, we set $T(\mathbf{x}, \mathbf{x}_0, y) = y_0$. Note that due to the FD, a key cannot map to two different values. We conclude the construction by projecting the relations of I^+ according to the schema of Q. These steps result in the construction of an instance $\sigma(I^+)$ and a lookup table T in linear time. Note that Δ hold in $\sigma(I^+)$ since Δ_{Q^+} contains them.

Given $\mu|_{\text{free}(Q)} \in Q(I)$, we define $\tau(\mu|_{\text{free}(Q)}) = \mu|_{\text{free}(Q)} \cup \nu_\mu$, where the mapping $\nu_\mu \colon \text{free}(Q^+) \setminus \text{free}(Q) \to dom$ uses the lookup table: For every $y \in \text{free}(Q^+) \setminus \text{free}(Q)$ added due to some FD $R_j \colon X \to y$, we set $\nu_\mu(y) = T[(\mathbf{x}, \mu(\mathbf{x}), y)]$. Note that τ is computable in constant time since we use the lookup table $|\text{free}(Q^+) \setminus \text{free}(Q)|$ times, and each access takes constant time.

Correctness We first claim that the lookup table succeeds in reconstructing the values for the missing head variables: if $\mu|_{\text{free}(Q)} \in Q(\sigma(I^+))$, then $\tau(\mu|_{\text{free}(Q)}) = \mu|_{\text{free}(Q^+)}$. By definition, for every $y \in \text{free}(Q)$, $\tau(\mu(y)) = \mu(y)$. We need to show the same for $y \in \text{free}(Q^+) \setminus \text{free}(Q)$. In this case, y was added to the head due to some FD $R_j \colon X \to y$, and $\tau(\mu(y))$ is defined to be $\nu_\mu(y) = T[(\mathbf{x}, \mu(\mathbf{x}), y)]$. Since μ is a homomorphism into $\sigma(I^+)$, there exists some tuple in $R_j^{\sigma(I^+)}$ that assigns $\mathbf{x}$ and y with $\mu(\mathbf{x})$ and respectively $\mu(y)$. This tuple is a projection of a tuple in $R_j^{I^+}$ that assigns $\mathbf{x}$ and y with the same values. Due to this tuple, when constructing the lookup table, we set $T[(\mathbf{x}, \mu(\mathbf{x}), y)] = \mu(y)$.

We now show that $Q^+(I^+) = \{\mu|_{\text{free}(Q^+)} : \mu|_{\text{free}(Q)} \in Q(\sigma(I^+))\}$. The first direction is that if $\mu|_{\text{free}(Q^+)} \in Q^+(I^+)$, then $\mu|_{\text{free}(Q)} \in Q(\sigma(I^+))$. Indeed, if μ maps the relations $R_i^+(\mathbf{u}_i)$ to the tuples s_i respectively, these tuples agree on the value of y for every FD $X \to y$ in Δ_{Q^+}, and so none of them were removed in the cleaning phase. After the projection, μ maps $R_i(\mathbf{v}_i)$ to the same (perhaps projected) tuples in $R_i^{\sigma(I^+)}$, and so it is still a homomorphism in Q. The second direction is that if $\mu|_{\text{free}(Q)} \in Q(\sigma(I^+))$, then $\mu|_{\text{free}(Q^+)} \in Q^+(I^+)$. We know that μ is a homomorphism from Q to $\sigma(I^+)$. For every $1 \leq i \leq m$, denote by t_i the tuple $t_i = \mu(\mathbf{v}_i)$. These tuples are projections of tuples in I^+. Denote by s_i the tuple in $R_i^{I^+}$ whose projection is t_i. If an atom R_k was extended due to an FD $R_j \colon X \to y$, then t_j and s_k must agree on y, otherwise this s_k would have been deleted in the cleaning phase. Therefore, s_k assigns y with $\mu(y)$. We conclude that $\mu|_{\text{free}(Q^+)} \in Q^+(I^+)$.

As described in Corollary 1, a direct consequence of Theorem 2 is that FD-extensions can be used to expand tractable enumeration classes. This is due to the fact that $\text{ENUM}_\Delta\langle Q\rangle \leq_e \text{ENUM}_{\Delta_{Q^+}}\langle Q^+\rangle$. The opposite direction is used in Section 4 to show the lower bounds required for a dichotomy.

Corollary 1 *Let $\mathcal{C}$ be an enumeration class that is closed under exact reduction. Let Q be a CQ and let Q^+ be its FD-extension. If $\text{ENUM}_{\Delta_{Q^+}}\langle Q^+\rangle \in \mathcal{C}$, then $\text{ENUM}_\Delta\langle Q\rangle \in \mathcal{C}$.*

Proof According to Theorem 2, $\text{ENUM}_\Delta\langle Q\rangle \leq_e \text{ENUM}_{\Delta_{Q^+}}\langle Q^+\rangle$. Since $\mathcal{C}$ is closed under exact reduction, if $\text{ENUM}_{\Delta_{Q^+}}\langle Q^+\rangle \in \mathcal{C}$, then $\text{ENUM}_\Delta\langle Q\rangle \in \mathcal{C}$. □

Since free-connex queries are in $\text{D}elayC_{lin}$ and $\text{D}elayC_{lin}$ is closed under exact reduction, we get the following corollary.

Corollary 2 *Let Q be a CQ over a schema $\mathcal{S} = (\mathcal{R}, \Delta)$. If Q is FD-free-connex, then $\text{ENUM}_\Delta\langle Q\rangle \in \text{D}elayC_{lin}$.*

Proof Since Q^+ is free-connex, and according to Theorem 1, we have that $\text{ENUM}_\emptyset\langle Q^+\rangle \in \text{D}elayC_{lin}$. Given an instance over the schema $(\mathcal{R}, \Delta_{Q^+})$, the same instance is also valid over $(\mathcal{R}, \emptyset)$, so using the identity mapping shows that $\text{ENUM}_{\Delta_{Q^+}}\langle Q^+\rangle \leq_e \text{ENUM}_\emptyset\langle Q^+\rangle$. Therefore, $\text{ENUM}_{\Delta_{Q^+}}\langle Q\rangle \in \text{D}elayC_{lin}$, and from Corollary 1 we get that $\text{ENUM}_\Delta\langle Q\rangle \in \text{D}elayC_{lin}$. □

We can now revisit Example 1. The query $Q(x, y) \leftarrow R_1(z, x), R_2(z, y)$ is not free-connex. Therefore, ignoring the FDs, according to Theorem 1 it is not in $DelayC_{lin}$. However, given $R_2 \colon z \rightarrow y$, its FD-extension is $Q^+(x, y) \leftarrow R_1^+(z, y, x), R_2^+(z, y)$. As it is free-connex, evaluating Q^+ is in $DelayC_{lin}$ by Corollary 2.

4 A Dichotomy for Acyclic CQs

In this section, we characterize which self-join-free FD-acyclic queries are in $DelayC_{lin}$. We use the notion of FD-extended queries defined in the previous section to establish a dichotomy stating that enumerating the answers to an acyclic query is in $DelayC_{lin}$ iff the query is FD-free-connex. The positive case for the dichotomy is described in Corollary 2, and this section concludes the negative case. We prove the following theorem:

Theorem 3 *Let Q be a self-join-free FD-acyclic CQ over the schema $\mathcal{S} = (\mathcal{R}, \Delta)$.*

– *If Q is FD-free-connex, then $\text{ENUM}_\Delta \langle Q \rangle \in DelayC_{lin}$.*
– *If Q is not FD-free-connex, then $\text{ENUM}_\Delta \langle Q \rangle \notin DelayC_{lin}$, assuming that the product of two $n \times n$ boolean matrices cannot be computed in time $\mathcal{O}(n^2)$.*

Note that the restriction of considering only self-joins-free queries is required only for the negative side. This assumption is standard [3, 6, 14], as it allows to assign different atoms with different relations independently. The hardness result described here builds on that of Bagan et al. [3] for databases that are assumed not to have FDs, and it relies on the hardness of *the boolean matrix multiplication problem*. This problem is defined as the evaluation $\text{ENUM}_\emptyset \langle \Pi \rangle$ of the query $\Pi(x, y) \leftarrow A(x, z), B(z, y)$ over the schema $(\{A, B\}, \emptyset)$ where $A, B \subseteq \{1, \ldots, n\}^2$. It is strongly conjectured that this problem is not computable in $\mathcal{O}(n^2)$ time, and the best currently known algorithms require $\mathcal{O}(n^\omega)$ time for some $2.37 < \omega < 2.38$ [1, 15].

The original proof describes an exact reduction $\text{ENUM}_\emptyset \langle \Pi \rangle \leq_e \text{ENUM}_\emptyset \langle Q \rangle$. Since Q is acyclic but not free-connex, it contains a head-path $(x, z_1, \ldots, z_k, y)$. For a given an instance of the matrix multiplication problem, an instance of $\text{ENUM}_\emptyset \langle Q \rangle$ is constructed, where the variables x, y and $z_1, \ldots, z_k$ of the head-path encode the variables x, y and z of Π, respectively. All other variables of Q are assigned constants. This way, A is encoded by an atom containing x and z_1, and B is encoded by an atom containing z_k and y. Atoms containing some z_i and z_{i+1} propagate the value of z. Since x and y are in $free(Q)$, but z_i are not, the answers to Q correspond to those of Π. As no atom of Q contains both x and y, the instance can be constructed in linear time. Constant delay enumeration for Q following a linear time preprocessing would result in the computation of the answers of Π in $\mathcal{O}(n^2)$ time.

FDs restrict the relations that can be assigned to atoms. This means that the reduction cannot be freely performed on databases with FDs, and the proof no longer holds. The following example illustrates where the reduction fails in the presence of FDs.

Example 3 The CQ from Example 1 has the form $Q(x, y) \leftarrow R_1(z, x), R_2(z, y)$ with the single FD $\Delta = \{R_2 \colon z \to y\}$. In the previous section, we show that it is in $\mathsf{D}elay\mathsf{C}_{lin}$, so the reduction should fail. Indeed, it would assign R_2 with the same relation as B of the matrix multiplication problem, but this may have two tuples with the same z value and different y values. Therefore, the construction does not yield a valid instance of $\mathrm{ENUM}_\Delta\langle Q \rangle$.

We now provide a modification of this construction to show an exact reduction from $\mathrm{ENUM}_\emptyset\langle\Pi\rangle$ to $\mathrm{ENUM}_{\Delta_{Q^+}}\langle Q^+ \rangle$. Any violations of the FDs are fixed by carefully picking more variables other than those of the head-path to take the roles of x, y and z of the matrix multiplication problem. This is done by introducing the sets V_x, V_y and V_z which are subsets of $var(Q)$. We say that a variable β *plays the role* of α, if $\beta \in V_\alpha$. To clarify the reduction, we start by describing a restricted case, where all FDs are unary. The basic idea in the case of general FDs remains the same, but it requires a more involved construction of the sets V_α.

4.1 Unary Functional Dependencies

For the unary case, we define the sets V_x, V_y and V_z to hold the variables that iteratively imply x, y and some z_i, respectively. That is, for $\alpha \in \{x, y, z_1, \dots, z_k\}$ we set $V_\alpha := \{\alpha\}$ and apply $V_\alpha := V_\alpha \cup \{\gamma \in var(Q) \mid \gamma \to \beta \in \Delta_{Q^+} \wedge \beta \in V_\alpha\}$ until a fixpoint is reached. We then define $V_z := V_{z_1} \cup \dots \cup V_{z_k}$.

The Reduction Let $I = (A^I, B^I)$ be an instance of $\mathrm{ENUM}_\emptyset\langle\Pi\rangle$. We define $\sigma(I)$ by describing the relation R^I for every atom $R(\mathbf{v}) \in atoms(Q^+)$. If $var(R) \cap V_y = \emptyset$, then every tuple $(a, c) \in A^I$ is copied to a tuple in R^I. Variables in V_x get the value a, variables in V_z get the value c, and variables that play no role are assigned a constant $\perp$. That is, we define $R^{\sigma(I)}$ to be $\{(f(v_1, a, c), \dots, f(v_k, a, c)) \mid (a, c) \in A^I\}$, where:

$$f(v_i, a, c) = \begin{cases} a & \text{if } v_i \in V_x \setminus V_z, \\ c & \text{if } v_i \in V_z \setminus V_x, \\ (a, c) & \text{if } v_i \in V_x \cap V_z, \\ \perp & \text{otherwise.} \end{cases}$$

If $var(R) \cap V_y \neq \emptyset$, we show that $var(R) \cap V_x = \emptyset$, see the sketch for the proof for the well-definedness of the reduction below. In this case we define the relation similarly with B^I. Given a tuple $(c, b) \in B^I$, the variables of V_y get the value b, and those of V_z are assigned with c.

Example 4 Consider the query $Q^+(x, y, v) \leftarrow R(u, x, z), S(v, y, z)$ with FDs $\Delta_{Q^+} = \{R \colon u \to x, R \colon u \to z, S \colon y \to v\}$. Using the head-path (x, z, y), the reduction sets $V_x = \{x, u\}$, $V_y = \{y\}$ and $V_z = \{z, u\}$. Given an instance I of the matrix multiplication problem with relations A^I and B^I, every tuple $(a, c) \in A^I$ results in a tuple $((a, c), a, c) \in R^{\sigma(I)}$, and every tuple $(c, b) \in B^I$ results in a tuple $(\perp, b, c) \in S^I$.

We now outline the correctness of this reduction.

Well-defined reduction: For an atom R, either $var(R) \cap V_y = \emptyset$ or $var(R) \cap V_x = \emptyset$. That is, no atom contains variables from both V_x and V_y. Due to the definition of Q^+, this atom would otherwise also contain both x and y. However, they cannot appear in the same relation according to the definition of a head-path. The reduction is therefore well defined, and it can be constructed in linear time via copy and projection.

Preserving FDs: The construction ensures that if an FD $\gamma \to \alpha$ exists, then γ has all the roles of α. Therefore, either α has no role and corresponds to the constant $\bot$, or every value that appears in α also appears in γ. In any case, all FDs are preserved.

1-1 mapping of answers: If a variable of V_z had appeared in the head of Q^+, then by the definition of Q^+, some z_i would have been in the head as well. This cannot happen according to the definition of a head-path. Therefore, $V_z \cap \text{free}(Q^+) = \emptyset$, and the head only encodes the x and y values of the matrix multiplication problem, so two different solutions to $\text{ENUM}_{\Delta_{Q^+}}\langle Q^+ \rangle$ must differ in either x or y, and correspond to different solutions of $\text{ENUM}_\emptyset \langle \Pi \rangle$. For the other direction, the head necessarily contains the variables x and y. Therefore, two different solutions to $\text{ENUM}_\emptyset \langle \Pi \rangle$ correspond to different solutions of $\text{ENUM}_{\Delta_{Q^+}} \langle Q^+ \rangle$.

4.2 General Functional Dependencies

Next we show how to lift the idea of this reduction to the case of general FDs. In the case of unary FDs, we ensure that the construction does not violate a given FD $\gamma \to \alpha$, by simply encoding the values of α to γ. In the general case, when allowing more than one variable on the left-hand side of an FD $\gamma_1, \ldots, \gamma_k \to \alpha$, we must be careful when choosing the variables γ_j to which we copy the values of α. Otherwise, as the following example shows, we will not be able to construct the instance in linear time.

Example 5 Consider the query $Q(x, y) \leftarrow R_1(x, z, t_1), R_2(z, y, t_1, t_2)$ over a schema with the FD $R_2 : t_1 t_2 \to y$. Note that $Q = Q^+$ is acyclic but not free-connex, and that (x, z, y) is a head-path in $\mathcal{H}(Q^+)$. To repeat the idea shown in the unary case and ensure that the FDs still hold, the variable on the right-hand side of every FD is encoded to the variables on the left-hand side. If we encode y to t_1, then R_1 would contain the encodings of x, y and z. This means that its size will not be linear in that of the matrix multiplication instance, and we cannot hope for linear time construction. On the other hand, if we choose to encode y only to t_2, the reduction works.

In the following central lemma, we describe how to carefully pick the variables to which we assign roles in a way that meets the requirements we need for the reduction. We prove requirements 1 and 2 to guarantee a one-to-one mapping between the results of the two problems. Requirement 3 enables linear time construction, while requirement 4 is used to show that all FDs are preserved. The idea is that we consider the join-tree of Q^+ and define a partition of its atoms. We then define V_x and V_y to

hold variables that appear only in different parts of the tree, ensuring that no atom contains variables of each. The running intersection property of a join-tree is then used to guarantee that the sets are inclusive enough to correct all FD violations.

Lemma 1 *Let Q be a self-join-free CQ over a schema $\mathcal{S} = (\mathcal{R}, \Delta)$, such that Q^+ is acyclic but not free-connex. Further let $(x, z_1, \ldots, z_k, y)$ be a head-path of Q^+. Then, there exist sets of variables V_x, V_y and V_z such that:*

1. $x \in V_x$, $y \in V_y$, $\{z_1, \ldots z_k\} \subseteq V_z$.
2. $V_z \cap \text{free}(Q^+) = \emptyset$.
3. *For every $R \in \text{atoms}(Q^+)$: $var(R) \cap V_y = \emptyset$ or $var(R) \cap V_x = \emptyset$.*
4. *For every $U \to v \in \Delta_{Q^+}$ s.t. $v \in V_\alpha$ with $\alpha \in \{x, y, z\}$: $U \cap V_\alpha \neq \emptyset$.*

Proof We first define a partition of the atoms of Q into two or three sets: T_x, T_y and possibly T_{mid}. Let T be a join tree of $\mathcal{H}(Q^+)$, and denote the hyperedges on the head-path by $e(x, z_1), \ldots, e(z_k, y)$. Note that, by definition, each hyperedge of the head-path is a vertex of T and an atom of Q^+. By the running intersection property of T and since the path is cordless, we can conclude that there is a simple path P from $e(x, z_1)$ to $e(z_k, y)$ in T, such that $e(z_1, z_2), \ldots, e(z_{k-1}, z_k)$ lie on that path in the order induced by the head-path. Let sep_x be the first atom on the path P that does not contain x. This exists because $e(z_k, y)$ does not contain x, as the head-path is chordless. Similarly, let sep_y be the last atom on P that does not contain y. Let T_x be the set of atoms v such that the unique path from v to $e(x, z_1)$ in T does not go through sep_x. Similarly, let T_y be the set of atoms w such that the unique path from w to $e(z_k, y)$ in T does not go through sep_y. Next set $T_{\text{mid}} = V(T) \setminus (T_x \cup T_y)$. Note that $e(x, z_1) \in T_x$ and $e(z_k, y) \in T_y$, but T_{mid} may be empty (this happens in the case that the head-path is of length two). By definition, the atoms of Q^+ are exactly $T_x \cup T_{\text{mid}} \cup T_y$, and next we show that this union is disjoint. Figure 1 depicts the established partition. □

Claim 3 The sets T_x and T_y are disjoint.

Proof of the Claim Assume by contradiction that there is some $v \in T_x \cap T_y$. Let P_x be the unique simple path from v to $e(x, z_1)$, and recall that since $v \in T_x$ it does not go through sep_x. Similarly let P_y be the unique simple path from v to $e(z_k, y)$ that does not go through sep_y.

We first claim that there exists some atoms w that appears in all three paths P, P_x and P_y. Take w to be the first atom on P_x that is also in P and set P_x^w to be the simple path from v to w. Such an atom w exists because the last atom of P_x is $e(x, z_1)$ which
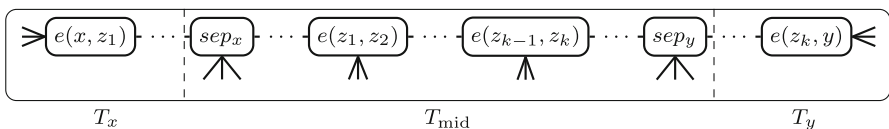
Fig. 1 Join tree T of $\mathcal{H}(Q^+)$ for head-paths of length greater than 2. The subtrees T_x, T_y and T_{mid} are disjoint, and are separated by the nodes sep_x and sep_y

is in P. Further set P^w to be the simple path from w to $e(z_k, y)$. Concatenating the paths P_x^w and P^w, we obtain a simple path from v to $e(z_k, y)$. Since the simple paths in a tree are unique, this is exactly P_y, and so w is also in P_y.

Our second claim is that if an atom u is in both P and P_x, then it contains the variable x. Assume by contradiction that such an atom u does not contain x. Then u is an atom on P not containing x, and by definition of sep_x, the simple path from u to $e(x, z_1)$ contains sep_x. As this path is a subpath of P_x, P_x contains sep_x, in contradiction to the fact that $v \in T_x$. Similarly, if an atom is in both P and P_y, then it contains y.

Combining the two claims, we have an atom w containing both x and y, in contradiction to the fact that a head-path is chordless by definition. Therefore we conclude that T_x and T_y are indeed disjoint.

Now we are ready to define the sets of variables V_x, V_y and V_z. We define V_x recursively to contain x and variables that imply those of V_x, but without variables that appear outside of T_x. V_y is defined symmetrically. V_z contains $z_1, \ldots, z_k$ and variables that imply those of V_z but without free variables. Formally,

$$\text{Implies}(V) = \{u \in var(Q) \mid \exists U \to w \in \Delta_{Q^+} \text{ with } w \in V \text{ and } u \in U\}$$

for $V \subseteq var(Q)$, and we define via fixpoint iteration the following:

- V_x: base $V_x := \{x\}$; rule $V_x := (V_x \cup \text{Implies}(V_x)) \setminus var(T_y \cup T_{\text{mid}})$
- V_y: base $V_y := \{y\}$; rule $V_y := (V_y \cup \text{Implies}(V_y)) \setminus var(T_x \cup T_{\text{mid}})$
- V_z: base $V_z := \{z_1, \ldots z_k\}$; rule $V_z := (V_z \cup \text{Implies}(V_z)) \setminus \text{free}(Q^+)$

We now prove that V_x, V_y and V_z meet the requirements of the lemma. Requirements 1 and 2 follow immediately from the definition of the sets. To prove requirement 3, let $R \in \text{atoms}(Q^+)$. If $R \in T_x$, then by definition of V_y we have that $var(R) \cap V_y = \emptyset$. Otherwise, $R \in T_y \cup T_{\text{mid}}$, and similarly $var(R) \cap V_x = \emptyset$. It is left to show requirement 4.

Let $\delta = U \to v \in \Delta_{Q^+}$ where $v \in V_\alpha$. We first show the case of $\alpha = z$. If $U \cap V_z = \emptyset$, then $U \subseteq \text{free}(Q^+)$, and by the definition of Q^+, $v \in \text{free}(Q^+)$, which is a contradiction to the definition of V_z. Now we prove the case where $\alpha = x$. The case $\alpha = y$ is symmetric. Denote by $e(U, v)$ an atom containing all variables of δ. As $v \in V_x$, we know that $e(U, v) \notin T_y \cup T_{\text{mid}}$, therefore $e(U, v) \in T_x$. Assume by contradiction that $U \cap V_x = \emptyset$. Let $u \in U$. By definition of V_x, this means that $u \in var(e_u)$ for some $e_u \in T_y \cup T_{\text{mid}}$. As T_x, T_y and T_{mid} are disjoint, we have that $e_u \notin T_x$, which means that the path between e_u and $e(x, z_1)$ goes through sep_x. This means that the path from e_u to $e(U, v)$ goes through sep_x too, otherwise the concatenation of this path with the path from $e(U, v)$ to $e(x, z_1)$ would result in a path from e_u to $e(x, z_1)$ not going through sep_x. By the running intersection property, $u \in var(sep_x)$. Since this is true for all $u \in U$, it follows that $v \in var(sep_x)$ by definition of Q^+, contradicting the fact that $v \in V_x$. $\qquad\square$

With the sets V_x, V_y, V_z at hand, we can now perform the reduction in the presence of general FDs.

Lemma 2 *Let Q be a self-join-free CQ over a schema $\mathcal{S} = (\mathcal{R}, \Delta)$. If Q^+ is acyclic and not free-connex, then* $\text{ENUM}_\emptyset\langle\Pi\rangle \leq_e \text{ENUM}_{\Delta_{Q^+}}\langle Q^+\rangle$.

Proof Let $I_{A,B} = (A^I, B^I)$ be an instance of $\text{ENUM}_\emptyset\langle\Pi\rangle$ over the domain $\mathcal{D} = \{1, \ldots, n\}$. We define an instance $\sigma(I_{A,B})$ of $\text{ENUM}_{\Delta_{Q^+}}\langle Q^+\rangle$ based on the sets V_x, V_y and V_z from Lemma 1 and the relations A^I and B^I. Since Q^+ is acyclic but not free-connex, it contains some head-path $(x, z_1, \ldots, z_k, y)$.

To define the instance $\sigma(I_{A,B})$, we first fix the functions f_A and f_B:

$$
f_A(v, a, c) = \begin{cases} a & : v \in V_x \setminus V_z \\ c & : v \in V_z \setminus V_x \\ (a, c) & : v \in V_x \cap V_z \\ \bot & : \text{otherwise} \end{cases}, \qquad f_B(v, b, c) = \begin{cases} b & : v \in V_y \setminus V_z \\ c & : v \in V_z \setminus V_y \\ (b, c) & : v \in V_y \cap V_z \\ \bot & : \text{otherwise} \end{cases}
$$

We partition all relational atoms of Q^+ into two sets: $\mathcal{R}_A^+$ and $\mathcal{R}_B^+$. The set $\mathcal{R}_A^+$ is defined as $\{R^+ \in \text{atoms}(Q^+) \mid var(R^+) \cap V_y = \emptyset\}$ and $\mathcal{R}_B^+$ is $\text{atoms}(Q^+) \setminus \mathcal{R}_A^+$. To obtain an instance $\sigma(I_{A,B})$ of $\text{ENUM}_{\Delta^+}\langle Q^+\rangle$, we apply f_A to the atoms in $\mathcal{R}_A^+$ using the values of A^I, while the atoms in $\mathcal{R}_B^+$ use f_B and B^I. That is, if $R^+(u_1, \ldots, u_m) \in \mathcal{R}_A^+$, then $(R^+)^{\sigma(I_{A,B})}$ is defined to be $\{(f_A(u_1, a, c), \ldots, f_A(u_m, a, c)) \mid (a, c) \in A^I\}$. Otherwise, $R^+(u_1, \ldots, u_m)$ is in $\mathcal{R}_B^+$, and $(R^+)^{\sigma(I_{A,B})} = \{(f_B(v_1, b, c), \ldots, f_B(v_p, b, c)) \mid (c, b) \in B^I\}$. The mapping τ is defined as the projection onto the variables x and y. Note that the instance can be constructed in linear time, and the projection can be computed in constant time.

We now claim that $\sigma(I_{A,B})$ is a database over the schema $(\mathcal{R}^+, \Delta_{Q^+})$, as all the FDs of Δ_{Q^+} are satisfied. Let $\delta = R_j^+: U \to v \in \Delta_{Q^+}$. If $v \notin V_x \cup V_y \cup V_z$, then δ holds as v is assigned the value $\bot$ in every tuple in $(R_j^+)^{\sigma(I_{A,B})}$. Next assume that $v \in V_x \setminus V_z$. By point 4 of Lemma 1, there is some $u \in U$ such that $u \in V_x$. Thus in every tuple in $(R_j^+)^{\sigma(I_{A,B})}$, if v is assigned the value a, then u is either assigned the value a or (a, c) for some $c \in \{1, \ldots, n\}$ and in either case δ is satisfied. The proof for the cases where $v \in V_z \setminus (V_x \cup V_y)$ and $v \in V_y \setminus V_z$ is similar. Next assume that $v \in V_x \cap V_z$. By point 4 of Lemma 1, there are some $u_1, u_2 \in U$ such that $u_1 \in V_x$ and $u_2 \in V_z$ and for every tuple in $(R_j^+)^{\sigma(I_{A,B})}$, if v is assigned the value (a, c), then u_1 is either assigned the value a or (a, c), u_2 is either assigned the value c or (a, c), and so δ is satisfied. The case $v \in V_y \cap V_z$ is similar. Note that the case where $v \in V_x \cap V_y$ cannot occur due to point 3 of Lemma 1.

The structure of the head-path $(x, z_1 \ldots, z_k, y)$ guarantees that all answers to $\text{ENUM}_{\Delta_{Q^+}}\langle Q^+\rangle$ correspond to those of $\text{ENUM}_\emptyset\langle\Pi\rangle$ and vice versa. Indeed, let $\mu|_{\{x,y\}} \in \Pi(I_{A,B})$. We define $v_\mu : var(Q^+) \to \mathcal{D}$ as follows.

$$
v_\mu(v) = \begin{cases} \mu(x) & : v \in V_x \setminus V_z, \\ \mu(y) & : v \in V_y \setminus V_z, \\ \mu(z) & : v \in V_z \setminus (V_x \cup V_y), \\ (\mu(x), \mu(z)) & : v \in V_z \cap V_x, \\ (\mu(y), \mu(z)) & : v \in V_z \cap V_y, \\ \bot & : \text{otherwise.} \end{cases}
$$

By definition, $f_A(v, \mu(x), \mu(z)) = v_\mu(v)$ and $f_B(v, \mu(y), \mu(z)) = v_\mu(v)$. As $\mu|_{\{x,y\}} \in \Pi(I_{A,B})$, we have that $(\mu(x), \mu(z)) \in A^I$ and $(\mu(z), \mu(y)) \in B^I$.

Consider any atom $R^+(u_1, \ldots, u_m)$ of Q^+. If $R^+(u_1, \ldots, u_m) \in \mathcal{R}_A^+$, then $v_\mu(u_1, \ldots, u_m) = (f_A(u_1, \mu(x), \mu(z)), \ldots, f_A(u_m, \mu(x), \mu(z))) \in (R^+)^{\sigma(I_{A,B})}$. If $R^+(u_1, \ldots, u_m) \in \mathcal{R}_B^+$, we have that $var(R^+) \cap V_x = \emptyset$ by point 3 of Lemma 1. Then, $v_\mu(u_1, \ldots, u_m) = (f_B(u_1, \mu(z), \mu(y)), \ldots, f_B(u_m, \mu(z), \mu(y)))$ is in $(R^+)^{\sigma(I_{A,B})}$. Therefore, $v_\mu|_{\text{free}(Q^+)} \in Q^+(\sigma(I_{A,B}))$. Since $x \in V_x \setminus V_z$ and $y \in V_y \setminus V_z$, we have that $v_\mu(x) = \mu(x)$ and $v_\mu(y) = \mu(y)$, and so $\tau(v_\mu) = \mu|_{\{x,y\}}$. Moreover, any answer $\mu'|_{\text{free}(Q^+)} \in Q^+(\sigma(I_{A,B}))$ that has $\tau(\mu'|_{\text{free}(Q^+)}) = \mu|_{\{x,y\}}$ assigns $\mu(x)$ to variables in $V_x \setminus V_z$, assigns $\mu(y)$ to variables in $V_y \setminus V_z$ and assigns $\perp$ to variables not in $V_x \cup V_y \cup V_z$. By points 2 and 3 of Lemma 1, $V_z \cap \text{free}(Q^+) = \emptyset$ and $V_x \cap V_y = \emptyset$, so these are the only variables in $\text{free}(Q^+)$. Therefore $\mu'|_{\text{free}(Q^+)} = v_\mu|_{\text{free}(Q^+)}$.

Next assume that $\mu'|_{\text{free}(Q^+)} \in Q^+(\sigma(I_{A,B}))$. Let R_x^+ be an atom containing x and z_1 (such an atom exists by the definition of the head-path). By point 3 of Lemma 1 we have $var(R^+) \cap V_y = \emptyset$, and $R^+ \in \mathcal{R}_A^+$. By points 1 and 2 of Lemma 1 and since x is a free variable, we have $x \in V_x \setminus V_z$ and $z_1 \in V_z$. Thus there exists some $(a, c) \in A^I$ such that $\mu'(x) = f_A(x, a, c) = a$ and $\mu'(z_1) = f_A(z_1, a, c) \in \{a, (a, c)\}$. Similarly, there exists some $(c', b) \in B^I$ such that $\mu'(y) = f_B(y, b, c') = b$ and $\mu'(z_k) = f_B(z_k, b, c') \in \{c', (b, c')\}$. It remains to show that $c = c'$. We show by induction on i that $\mu'(z_i)$ is either c or of the form (t, c) with some value t. We know this fact for $i = 1$ since this is how we define c. If $k > 1$, then consider an atom $R_i^+(\mathbf{v}_i)$ containing $\{z_{i-1}, z_i\}$. Then μ' maps $\mathbf{v}_i$ to some tuple $t \in R_i^+$ that assigns z_{i-1} with a value of the form c or (t, c). Since $z_i \in V_z$, t also assigns z_i with a value of such a form, so $\mu'(z_i)$ is of the form c or (t, c) too. This show that $c = c'$. Therefore, $\tau(\mu') \in \Pi(I_{A,B})$. Moreover, since τ is simply a projection, $\tau(\mu')$ is uniquely defined. $\square$

By combining Theorem 2 and Lemma 2, we have the exact reduction $\text{ENUM}_\emptyset\langle\Pi\rangle \leq_e \text{ENUM}_{\Delta_{Q^+}}\langle Q^+\rangle \leq_e \text{ENUM}_\Delta\langle Q\rangle$. Therefore having $\text{ENUM}_\Delta\langle Q\rangle$ in $DelayC_{lin}$ would mean that $\text{ENUM}_\emptyset\langle\Pi\rangle \in DelayC_{lin}$, which contradicts the conjecture of the lower bound for matrix multiplication. This concludes the proof of Theorem 3. Note that Theorem 3 does not contradict the dichotomy of Theorem 1: if for a given query Q we have that Q^+ is acyclic but not free-connex, then Q is not free-connex by Proposition 1.

5 Cyclic CQs

In the previous section, we established a classification of FD-acyclic CQs, but we did not consider FD-cyclic queries. A known result states that, under certain assumptions, self-join-free cyclic queries are not in $DelayC_{lin}$ [6]. In this section, we therefore explore how FD-extensions can be used to obtain some insight on the implications of this result in the presence of FDs. We show that (under the same assumptions) self-join-free FD-cyclic queries that contain only unary FDs cannot be evaluated in linear time. For schemas containing only unary FDs, this extends the dichotomy presented in the previous section to all CQs, and also proves a dichotomy for the queries that can be enumerated in linear delay. We will prove the following theorem using the $\text{TETRA}(k)$ problem, which is defined below.

Theorem 4 *Let Q be a self-join-free CQ over a schema $\mathcal{S} = (\mathcal{R}, \Delta)$, where Δ only contains unary FDs. If Q is FD-cyclic, then $\mathrm{DECIDE}_\Delta\langle Q\rangle$ cannot be solved in linear time, assuming that the $\mathrm{TETRA}(k)$ problem cannot be solved in linear time for any $k \geq 3$.*

As before, the initial hardness proof for cyclic queries no longer holds in the presence of FDs, and we present a modified reduction that satisfies the FDs. We start by describing the hypothesis used to obtain the conditional lower bounds. $\mathrm{TETRA}(k)$ is defined as the hypergraph with the vertices $\{1, \ldots, k\}$ and the edges $\{\{1, \ldots, k\} \setminus \{i\} \mid i \in \{1, \ldots, k\}\}$. Let $\mathcal{H}$ be a hypergraph. With a slight abuse of notation, we also denote by $\mathrm{TETRA}(k)$ the decision problem of whether $\mathcal{H}$ contains a subhypergraph isomorphic to $\mathrm{TETRA}(k)$.

$\mathrm{TETRA}(3)$ is the problem of deciding whether a graph contains a triangle, and it is strongly believed not to be solvable within time linear in the size of the graph [19]. The generalization of this assumption is that $\mathrm{TETRA}(k)$ cannot be solved in time linear in the size of the graph for any $k \geq 3$. This assumption is stronger than the one used in Section 4, as $\mathrm{TETRA}(3)$ can be reduced to the matrix multiplication problem [19]. However, we do have reasons to also believe the $\mathrm{TETRA}(k)$ assumption for any $k \geq 4$. The $(\ell, k)-$Hyperclique Hypothesis [16] states that, in a k-uniform hypergraph of n vertices, $n^{k-o(1)}$ time is required to find a set of ℓ vertices such that each of it subsets of size k forms a hyperedge. Solving $\mathrm{TETRA}(k)$ in linear time would contradict the $(\ell, k)-$Hyperclique Hypothesis. Indeed, an algorithm that decides whether a hypergraph contains a $\mathrm{TETRA}(k)$ in linear time runs in at most n^{k-1} many steps, and thus can also detect a $(k-1)$-hyperclique in a k-uniform graph in time n^{k-1}, which is less than $n^{k-o(1)}$.

We will show that if Q^+ is cyclic and only unary FDs are present, the problem $\mathrm{TETRA}(k)$ for some k can be reduced to $\mathrm{DECIDE}_{\Delta_{Q^+}}\langle Q^+\rangle$. In the following definition, $\mathcal{H}^b$ is a pseudo-minor isomorphic to some $\mathrm{TETRA}(k)$. To perform the said reduction, we will use this pseudo-minor on a graph describing our query.

Definition 3 Let $\mathcal{H}$ be a cyclic hypergraph. We denote by $\mathsf{T}et_{pm}(\mathcal{H})$ the pairs of pseudo-minors $(\mathcal{H}^a, \mathcal{H}^b)$ of $\mathcal{H}$ such that:

1. $\mathcal{H}^a$ is obtained by a (possibly empty) set of vertex removal and edge removal operations on $\mathcal{H}$.
2. $\mathcal{H}^b$ is obtained by a (possibly empty) set of edge contraction and edge removal operations on $\mathcal{H}^a$.
3. $\mathcal{H}^b$ is isomorphic to $\mathrm{TETRA}(k)$ for some $k \geq 3$.
4. Either $\mathcal{H}^a = \mathcal{H}^b$ or $\mathcal{H}^a$ is a chordless cycle.

Given a query Q, we define $\mathsf{T}et_{pm}(Q) = \mathsf{T}et_{pm}(\mathcal{H}(Q))$.

Brault-Baron [6, Theorem 11] showed that a cyclic hypergraph $\mathcal{H}$ admits some $\mathrm{TETRA}(k)$ as a pseudo-minor. We describe the proof here in our terminology.

Lemma 3 ([6], Theorem 11) *Let $\mathcal{H}$ be a hypergraph. If $\mathcal{H}$ is cyclic, then $\mathsf{T}et_{pm}(\mathcal{H})$ is non-empty.*

Proof If $\mathcal{H}$ has a chordless cycle C as an induced subgraph, then removing vertices not in C followed by performing all possible edge removals results in a chordless cycle $\mathcal{H}^a$. Then, $\mathcal{H}^b$ isomorphic to TETRA(3) is obtained by a repeated use of edge-contraction followed by performing all possible edge removals. In this case, $(\mathcal{H}^a, \mathcal{H}^b) \in \mathsf{Tet}_{pm}(\mathcal{H})$. If $\mathcal{H}$ does not contain a chordless cycle, since it is not acyclic, it is non-conformal. Consider its smallest non-conformal clique. The clique is not contained in any edge (since it is non-conformal), and it is a TETRA(k) because of its minimality. Therefore, removing all vertices other than the clique, and then performing all possible edge removals, results in a graph $\mathcal{H}^a = \mathcal{H}^b$ isomorphic to some TETRA(k). Again, $(\mathcal{H}^a, \mathcal{H}^b) \in \mathsf{Tet}_{pm}(\mathcal{H})$. $\qquad\square$

For the reduction we present next, we first need to show that for an FD-cyclic query Q, no pseudo-minor in $\mathsf{Tet}_{pm}(Q^+)$ contains all variables of any FD $X \rightarrow y$. In the following we assume that Δ only contains non-trivial FDs, meaning $y \notin X$.

Lemma 4 *Let Q be a self-join-free FD-cyclic CQ over a schema $\mathcal{S} = (\mathcal{R}, \Delta)$. Let $(\mathcal{H}^a, \mathcal{H}^b) \in \mathsf{Tet}_{pm}(Q^+)$ and $\mathcal{H}^a = (V, E)$. For every non-trivial $X \rightarrow y \in \Delta_{Q^+}$, we have $X \cup \{y\} \not\subseteq V$.*

Proof We start with an observation regarding the FDs. Note that in $\mathcal{H}(Q^+)$ some edge contains the vertices $X \cup \{y\}$, and by the construction of Q^+, every edge that contains X must also contain y. These properties still hold after any sequence of vertex removals and edge removals as long as none of the vertices $X \cup \{y\}$ are removed. Therefore if none of the vertices $X \cup \{y\}$ were removed, there must be an edge in $\mathcal{H}^a$ containing $X \cup \{y\}$, and every edge in $\mathcal{H}^a$ containing X also contains y.

We distinguish two cases. In the first case, $\mathcal{H}^b = \mathcal{H}^a$ is a TETRA(k) obtained from $\mathcal{H}(Q^+)$ by a sequence of vertex and edge removals. If $X \cup \{y\} \subseteq V$, then by the definition of TETRA(k) it should contain the edge $V \setminus \{y\}$. Such an edge cannot exist since it contains all of X but not y. Therefore, such a TETRA(k) cannot contain all of $X \cup \{y\}$, and in this case we conclude that $X \cup \{y\} \not\subseteq V$. In the second case, $\mathcal{H}^a$ is a cycle, and $\mathcal{H}^b$ is a TETRA(3) obtained by performing edge contraction steps on it. If none of $X \cup \{y\}$ were removed, some edge of $\mathcal{H}^a$ must contain all of them. Since all edges are of size 2 it must be that $|X| = 1$. Denote $X = \{x\}$. Since we consider a cycle containing both x and y, there should be at least one edge containing x but not containing y. Since we showed it is not possible, such a cycle cannot contain all of $X \cup \{y\}$. $\qquad\square$

We are now ready to establish the reduction. Given a pseudo-minor of $\mathsf{Tet}_{pm}(Q^+)$ isomorphic to some TETRA(k), we can reduce the problem of checking whether a hypergraph contains a subhypergraph isomorphic to TETRA(k) to finding a boolean answer to Q^+.

Lemma 5 *Let Q be a self-join-free FD-cyclic CQ over a schema $\mathcal{S} = (\mathcal{R}, \Delta)$, where Δ only contains unary FDs. Let $(\mathcal{H}^a, \mathcal{H}^b) \in \mathsf{Tet}_{pm}(Q^+)$ such that $\mathcal{H}^b$ is isomorphic to TETRA(k) for some k. Then, there is a linear time reduction TETRA(k) $\leq_m$ DECIDE$_{\Delta_{Q^+}}\langle Q^+ \rangle$.*

Proof Given an input hypergraph $\mathcal{G}$ for the TETRA(k) problem, we define an instance I of DECIDE$_{\Delta_{Q^+}}\langle Q^+\rangle$. We consider a sequence of pseudo-minors $\mathcal{H}(Q^+) = \mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_m = \mathcal{H}^b$, each pseudo-minor is obtained by performing one operation over the previous one, where $\mathcal{H}_j = \mathcal{H}^a$ for some $1 \leq j \leq m$. We treat hypergraphs as describing CQs. That is, to the hypergraph $\mathcal{H}_i$ we associate a query Q_i such that $\mathcal{H}(Q_i) = \mathcal{H}_i$. Every edge e of $\mathcal{H}_i$ corresponds to an atom in Q_i with a relational symbol R_e^i, and the vertices of e are its variables. We assume that the variables in every atom are sorted by some total order. In the following, we construct instances I_i to these queries. It is possible to define a instance I_m such that deciding $Q_m(I_m)$ solves TETRA(k) [6, Lemma 20]. We describe how to inductively build an instance $I_1 = I$ such that deciding $Q^+(I)$ solves the same problem.

Constructing I We first define I_m. For every edge e of $\mathcal{H}_m$, we define a relation R_e^m that contains all edges of $\mathcal{G}$ that have the same size as e. A tuple of R_e^m consists of the vertices of such an edge sorted by some total order on the vertices of $\mathcal{G}$. We now define I_i given I_{i+1}. We distinguish three cases according to the type of pseudo-minor operation that leads from $\mathcal{H}_i$ to $\mathcal{H}_{i+1}$.

- *edge removal*: For every $e'' \in \mathcal{H}_{i+1}$, set $R_{e''}^i = R_{e''}^{i+1}$. Then, let e be the edge removed, and let e' be an edge containing it. Set R_e^i to be a copy of $R_{e'}^{i+1}$ projected accordingly.
- *edge contraction*: Let v be the vertex replaced by its neighbor u. For any edge $e \in \mathcal{H}_i$ contracting to an edge $e' \in \mathcal{H}_{i+1}$, set R_e^i to be a copy of $R_{e'}^{i+1}$, and assign the attribute v a copy of the value of u in every tuple. Then, if $u \notin e$, project u out of R_e^i. For every other edge $e'' \in \mathcal{H}_i$, set $R_{e''}^i = R_{e''}^{i+1}$.
- *vertex removal*: Let v be the vertex removed, and let $e \in \mathcal{H}_i$ be an edge containing v resulting in an edge $e' \in \mathcal{H}_{i+1}$. Expand $R_{e'}^{i+1}$ to R_e^i by copying $R_{e'}^{i+1}$, and assign v with a constant $\perp$ in every tuple. Next apply the following FD-correction steps on v:

  1. In every tuple, concatenate to the value of v the values of variables it implies. These variables are defined via fixpoint iteration by ImpliedBy(v) = $\{v\}$ and ImpliedBy(v) = $\{w \mid t \to w \in \Delta_{Q^+}, t \in$ ImpliedBy(v)$\}$. For each $w \in$ ImpliedBy(v) $\setminus \{v\}$, if $R^i(\mathbf{u})$ is an atom such that $\mathbf{u}[k] = v$ and $\mathbf{u}[j] = w$, then in every tuple $t \in R^i$, replace $t[k]$ with $(t[k], t[j])$.
  2. After the value of v is determined, concatenate the value of v to the variables implying it. These variable are defined by Implies(v) = $\{v\}$ and Implies(v) = $\{u \mid u \to t \in \Delta_{Q^+}, t \in$ Implies(v)$\}$. For each variable $u \in$ Implies(v) $\setminus \{v\}$, if $R^i(\mathbf{u})$ is an atom such that $\mathbf{u}[k] = v$ and $\mathbf{u}[j] = u$, then in every tuple $t \in R^i$, replace $t[j]$ with $(t[j], t[k])$.

  For every edge $e'' \in \mathcal{H}_i$ not containing v, set $R_{e''}^i = R_{e''}^{i+1}$.

The overall construction of the instance I can be done in linear time, since there is a constant number of pseudo-minor operations, each requiring a linear number of computational steps.

I is an instance of $\mathcal{S}$ We show that I satisfies the FDs in Δ_{Q^+} by induction: we claim that for each $\mathcal{H}_i$ all FDs $x \rightarrow y$ such that $x, y \in V(\mathcal{H}_i)$ are satisfied. According to Lemma 4, $\mathcal{H}^a$ and therefore all of $\mathcal{H}_j, \ldots, \mathcal{H}_m$ do not contain all variables of any FD. Therefore our claim trivially holds for $\mathcal{H}_j, \ldots, \mathcal{H}_m$. We now prove our claim for $\mathcal{H}_i$ where $i \leq j - 1$. Consider an FD $\delta = x \rightarrow y$ such that $x, y \in V(\mathcal{H}_i)$. There are three cases:

- If $x, y \in V(\mathcal{H}_{i+1})$, then by the induction assumption δ is satisfied in $\mathcal{H}_{i+1}$. If $\mathcal{H}_{i+1}$ is obtained by edge removal, then the only new relation in $\mathcal{H}_i$ is a projection of a relation of $\mathcal{H}_{i+1}$, and therefore δ is satisfied in all relations. Otherwise, $\mathcal{H}_{i+1}$ is obtained by vertex removal. If the value of y is the same in R_e^i as in R_e^{i+1}, we are done by the induction assumption. Otherwise, y is changed due to the second FD-correction step, and the vertex removed is some z such that $y \rightarrow z$. In this case, since x transitively implies z, both x and y are concatenated with the same values, and δ is still satisfied.
- If $x \notin V(\mathcal{H}_{i+1})$, then $\mathcal{H}_{i+1}$ is obtained by the removal of the vertex x, and the first FD-correction step ensures that x contains a copy of the values of y in every tuple where they both appear. Therefore δ is satisfied.
- If $y \notin V(\mathcal{H}_{i+1})$, then $\mathcal{H}_{i+1}$ is obtained by the removal of the vertex y. The second FD-correction step ensures that x contains a copy of the values of y, and δ is satisfied.

Correctness We know [6, Lemma 20] that there is a solution to $Q_m(I_m)$ iff there exists a subhypergraph of $\mathcal{G}$ isomorphic to $\mathcal{H}_t$, and in fact every mapping μ that can be used for the evaluation corresponds to such a subhypergraph. We claim that every mapping used for evaluating $Q_{i+1}(I_{i+1})$ corresponds to a mapping that can be used for $Q_i(I_i)$, and vice versa. This was already shown in case $\mathcal{H}_{i+1}$ is obtained by $\mathcal{H}_i$ via edge contraction [6, Lemma 15] or edge removal [6, Lemma 14], and it was shown for vertex removal [6, Lemma 13] if we simply assign the new vertex with a constant and skip the FD-correction steps. Let $\mathcal{H}_{i+1}$ be a pseudo-minor obtained from $\mathcal{H}_i$ via vertex removal, and denote by I_i^0 the instance constructed from I_{i+1} as described but without the FD-correction steps. It is left to show that a mapping μ^0 that satisfies $Q_i(I_i^0)$ corresponds to a mapping μ that satisfies $Q_i(I_i)$, and vice versa. This will conclude that $\mathcal{G}$ has a subhypergraph isomorphic to $\mathcal{H}_t$ iff $Q^+(I) \neq \emptyset$.

First we claim that whenever we perform an FD-correction step, if x implies y and x is in some atom, then y is in this atom too. This will help us show that we perform the same changes over x in all relations. Since Q^+ is an FD-extension and only unary FDs are present, we are guaranteed that if x implies y, then y is present in every edge of $\mathcal{H}(Q^+)$ where x appears. This property is preserved under vertex removal and edge removal operations (as long as x and y are not removed), which are the only operations that can be performed between $\mathcal{H}(Q^+)$ and any pseudo-minor on which we perform vertex-removal.

We now show that given μ^0 that satisfies $Q_i(I_i^0)$ there is a mapping μ that satisfies $Q_i(I_i)$, and vice versa. We show this by induction, considering one FD-correction step involving one variable at a time. Consider the first FD-correction step on a vertex v implying w. In any atom $R(\mathbf{u})$ such that $\mathbf{u}[k] = v$, we showed that there exists

an index j such that $\mathbf{u}[j] = w$. For every tuple $t_0 \in R^0$, there is a similar tuple $t \in R$ with the only difference being $t[k] = (t_0[k], t_0[j])$. Therefore, by defining $\mu(v) = (\mu_0(v), \mu_0(w))$ and $\mu(u) = \mu_0(u)$ for all other variables $u \neq v$, every tuple that is used in the evaluation of μ^0 in I_i^0 results in a tuple that can be used in the evaluation of μ in I_i. Indeed, μ is a valid evaluation of $Q_i(I_i)$. A similar argument holds similarly for the opposite direction and for the second FD-correction step. For the opposite direction for example, if $\mu(v) = (a_v, a_w)$, we define $\mu^0(v) = a_v$. □

Theorem 4 is an immediate consequence of Lemma 5:

Proof of Theorem 4 For the sake of a contradiction assume that Q is FD-cyclic, and $\text{DECIDE}_\Delta \langle Q \rangle$ is solvable in linear time. Theorem 2 implies a linear time reduction $\text{DECIDE}_{\Delta_{Q^+}} \langle Q^+ \rangle \leq_m \text{DECIDE}_\Delta \langle Q \rangle$. Therefore, it is possible to solve $\text{DECIDE}_{\Delta_{Q^+}} \langle Q^+ \rangle$ in linear time as well. As Q^+ is cyclic, there exists a pseudo-minor $\mathcal{H}_{pm} \in \text{Tet}_{pm}(Q^+)$ isomorphic to $\text{TETRA}(k)$ for some $k \geq 3$. According to Lemma 5, this $\text{TETRA}(k)$ problem is also solvable in linear time. □

In terms of enumeration complexity, Theorem 4 means that any enumeration algorithm for such a query cannot output a first solution (or decide that there is none) within linear time, and we get the following corollary.

Corollary 3 *Let Q be a self-join-free CQ over a schema $\mathcal{S} = (\mathcal{R}, \Delta)$, where Δ only contains unary FDs. If Q is FD-cyclic, then $\text{ENUM}_\Delta \langle Q \rangle \notin DelayC_{lin}$, assuming that the $\text{TETRA}(k)$ problem is not solvable in linear time for any k.*

Less restrictive than constant delay enumeration, the class $DelayLin$ consists of enumeration problems that can be solved with a linear delay between solutions. Acyclic CQs are known to be in $DelayLin$ [3], and Corollary 1 shows that FD-acyclic CQs are in this class as well. Theorem 4 implies a lower bound for $DelayLin$ as well. Thus, we obtain a dichotomy stating that CQs are in $DelayLin$ iff they are FD-acyclic.

Theorem 5 *Let Q be a CQ with no self-joins over a schema $\mathcal{S} = (\mathcal{R}, \Delta)$, where Δ only contains unary FDs.*

- *If Q is FD-acyclic, then $\text{ENUM}_\Delta \langle Q \rangle \in DelayLin$.*
- *Otherwise (if Q is FD-cyclic), $\text{ENUM}_\Delta \langle Q \rangle \notin DelayLin$, assuming that the $\text{TETRA}(k)$ problem cannot be solved in linear time for any k.*

Proof If Q^+ is acyclic, then $\text{ENUM}_\emptyset \langle Q^+ \rangle \in DelayLin$ [3]. According to Theorem 2, $\text{ENUM}_\Delta \langle Q \rangle \leq_e \text{ENUM}_{\Delta_{Q^+}} \langle Q^+ \rangle$. Since every instance that satisfies Δ_{Q^+} also satisfies $\emptyset$, we conclude that $\text{ENUM}_{\Delta_{Q^+}} \langle Q^+ \rangle \leq_e \text{ENUM}_\emptyset \langle Q^+ \rangle$ using the identity mapping. Therefore, $\text{ENUM}_\Delta \langle Q \rangle \in DelayLin$, since $DelayLin$ is closed under exact reductions.

If Q^+ is cyclic, assume by contradiction that $\text{ENUM}_\Delta \langle Q \rangle \in DelayLin$. According to Corollary 3, $\text{ENUM}_{\Delta_{Q^+}} \langle Q^+ \rangle \in DelayLin$ as well. This means that finding

a first answer to Q^+ or deciding that there is none can be done in linear time, in contradiction to Theorem 4. □

We conclude this section with a short discussion about its extension to general FDs. The following example shows that the proof for Theorem 4 that was provided here cannot be lifted to general FDs. Exploring this extension is left for future work.

Example 6 Consider $Q() \leftarrow R_1(x, y, u), R_2(x, w, z), R_3(y, v, z), R_4(u, v, w)$ over a schema with all possible two-to-one FDs in R_1, R_2 and R_3. That is,

$$\Delta = \{xy \rightarrow u, yu \rightarrow x, ux \rightarrow y, zy \rightarrow v,$$
$$yv \rightarrow z, vz \rightarrow y, xz \rightarrow w, zw \rightarrow x, wx \rightarrow z\}.$$

Note that $Q^+ = Q$. The hypergraph $\mathcal{H}(Q^+)$ is cyclic, see Fig. 2, yet it is unclear whether Q can be solved in linear time, and whether TETRA(3) can be reduced to answering Q^+. Using Lemma 4, $\mathcal{H}(Q^+)$ has triangle pseudo-minors that do not contain all variables of any FD. Consider for example the one obtained by removing all vertices other than x, y, z. A construction similar to that of Lemma 5 would assign u with the values of x and y, assign v with the values of y and z, and assign w with the values of x and z. This results in the edge $\{u, v, w\}$ containing all three values of any possible triangle, meaning that this edge cannot be constructed in linear time. Other choices of triangle pseudo-minors lead to similar encoding problems.

6 Cardinality Dependencies

In this section, we show that our results also apply to CQs over schemas with cardinality dependencies. *Cardinality Dependencies* (CDs) [2, 7] are a generalization of FDs, where the left-hand side does not uniquely determine the right-hand side, but rather provides a bound on the number of distinct values it can have. Formally, Δ is the set of *CDs* of a schema $\mathcal{S} = (\mathcal{R}, \Delta)$. Every $\delta \in \Delta$ has the form $(R_i : A \rightarrow B, c)$, where $R_i : A \rightarrow B$ is an FD and c is a positive integer. A CD δ is *satisfied* by an instance I over $\mathcal{S}$, if every set of tuples $S \subseteq (R_i)^I$ that agrees on the indices of A,
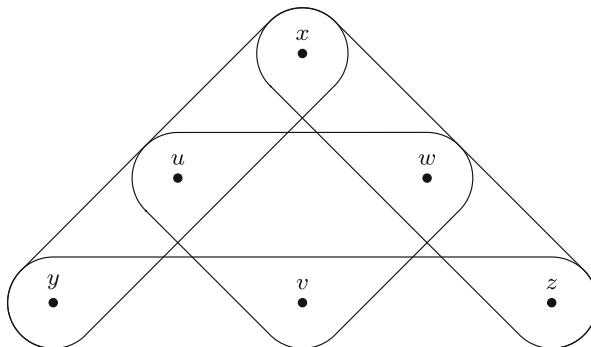

Fig. 2 The hypergraph $\mathcal{H}(Q^+)$ for Example 6

but no pair of them agrees on all indices of B, contains at most c tuples. It follows from the definition that δ is an FD if $c = 1$.

Denote by Δ^{FD} the FDs obtained from a set of CDs Δ by setting all c values to one. Given a query Q over $\mathcal{S} = (\mathcal{R}, \Delta)$, we define the *CD-extended query Q^+* of Q to be the FD-extended query of Q over $\mathcal{S} = (\mathcal{R}, \Delta^{FD})$. The schema $\mathcal{S}^+$ is defined with the original c values, and the extended CDs are

$$\Delta_{Q^+} = \{(R_i^+ : A \to b, c) \mid \exists (R_j : A \to B, c) \in \Delta, b \in B, A \cup \{b\} \subseteq var(R_i^+)\}.$$

Note that FD-extensions are indeed a special case of CD-extensions.

The hardness results extend to CDs because they are not more restrictive than FDs: Since every instance that preserves the FDs Δ^{FD} also preserves the CDs Δ, we conclude that $\text{ENUM}_{\Delta^{FD}}\langle Q \rangle \leq_e \text{ENUM}_{\Delta}\langle Q \rangle$. When only FDs are present we can apply Theorem 2, and get $\text{ENUM}_{\Delta_{Q^+}^{FD}}\langle Q^+ \rangle \leq_e \text{ENUM}_{\Delta^{FD}}\langle Q \rangle$. Combining the two we get the following lemma.

Lemma 6 *Let Q be a CQ over a schema $\mathcal{S} = (\mathcal{R}, \Delta)$, where Δ is a set of CDs, and let Q^+ be its CD-extension. Then $\text{ENUM}_{\Delta_{Q^+}^{FD}}\langle Q^+ \rangle \leq_e \text{ENUM}_{\Delta}\langle Q \rangle$.*

Defining the classes of *CD-acyclic* and *CD-free-connex* queries similarly to the case with FDs, Lemma 6 implies that all lower bounds presented in this paper hold for CDs. If Q is self-join-free and CD-acyclic but not CD-free-connex and $\text{ENUM}_{\Delta}\langle Q \rangle \in \text{Delay}C_{lin}$, then by Lemma 6 we have that $\text{ENUM}_{\Delta_{Q^+}^{FD}}\langle Q^+ \rangle$ is in $\text{Delay}C_{lin}$ as well. According to Lemma 2 this means that $\text{ENUM}_{\emptyset}\langle \Pi \rangle \in \text{Delay}C_{lin}$, and the matrix multiplication problem can be solved in quadratic time. So $\text{ENUM}_{\Delta}\langle Q \rangle \notin \text{Delay}C_{lin}$, assuming the boolean matrix multiplication assumption. Similarly, we conclude the hardness of self-join-free CD-cyclic CQs over schemas that contain only unary CDs, of the form $(A \to B, c)$ with $|A| = 1$. Combining Lemma 6 with Theorem 4, we have that such queries cannot be evaluated in linear time, assuming that the $\text{TETRA}(k)$ problem cannot be solved in linear time for any k.

In order to extend the positive results, we need to show that the CD-extension is at least as hard as the original query w.r.t. enumeration. We use a slight relaxation of exact reductions: For $\text{ENUM}\langle R_1 \rangle \leq_{e'} \text{ENUM}\langle R_2 \rangle$, instead of a bijection between the sets of outputs, one output of $\text{ENUM}\langle R_1 \rangle$ corresponds to at most a constant number of outputs of $\text{ENUM}\langle R_2 \rangle$.

Lemma 7 *Let Q be a CQ over a schema $\mathcal{S} = (\mathcal{R}, \Delta)$, where Δ is a set of CDs, and let Q^+ be its CD-extension. Then $\text{ENUM}_{\Delta}\langle Q \rangle \leq_{e'} \text{ENUM}_{\Delta_{Q^+}}\langle Q^+ \rangle$.*

Proof When dealing with FDs, we assume that the right-hand side has only one variable, as we can use such FDs to describe all possible ones. With CDs this no longer holds. Nonetheless, every instance of the schema $\mathcal{S} = (\mathcal{R}, \Delta)$ satisfies $\Delta^1 = \{(R_i : A \to b, c) \mid (R_i : A \to B, c) \in \Delta, b \in B\}$, so is also an instance of $\mathcal{S}^1 = (\mathcal{R}, \Delta^1)$. Therefore, $\text{ENUM}_{\Delta}\langle Q \rangle \leq_e \text{ENUM}_{\Delta^1}\langle Q \rangle$ using the identity mapping. It is left to show that $\text{ENUM}_{\Delta^1}\langle Q \rangle \leq_{e'} \text{ENUM}_{\Delta_{Q^+}}\langle Q^+ \rangle$. The proof idea is

the same as in Theorem 2, except now, for each tuple extended from R_i^I to $R_i^{I^+}$ we can have at most c new tuples. Since this process is only done a constant number of times, the construction still only requires linear time, and the rest of the proof holds. Note that now one solution of $\text{ENUM}_{\Delta^+}\langle Q^+\rangle$ may correspond to several solutions of $\text{ENUM}_{\Delta^1}\langle Q\rangle$, as some variables were possibly added to the head. However, as the possible values of the added head variables are bounded by CDs, the number of solutions of Q^+ that correspond to one solution of Q is bounded by a constant.

We now formally prove that $\text{ENUM}_{\Delta^1}\langle Q\rangle \leq_{e'} \text{ENUM}_{\Delta_{Q^+}}\langle Q^+\rangle$. Denote Q by $Q(\mathbf{p}) \leftarrow R_1(\mathbf{v}_1), \ldots, R_m(\mathbf{v}_m)$. Given an instance I of $\text{ENUM}_\Delta\langle Q\rangle$, we define $\sigma(I)$. We start by removing tuples that interfere with the extended dependencies. For every dependency $\delta = (R_j\colon X \to y, c) \in \Delta^1$ and every atom $R_k(\mathbf{v}_k)$ that contains $X \cup \{y\}$, we correct R_k according to δ: we only keep tuples of R_k^I that agree with some tuple of R_j^I over the values of $X \cup \{y\}$. Next, we follow the extension of the schema and extend the instance accordingly. This phase results in a sequence of instances $I_0, I_1, \ldots, I_n = \sigma(I)$ that correspond to a sequence of queries $Q = Q_0, Q_1, \ldots, Q_n = Q^+$ such that each query is the result of extending an atom or the head of the previous query according to an FD. If in step i the head was extended, we set $I_{i+1} = I_i$. Now assume some relation R_k is extended according to some CD $(R_j\colon X \to y, c)$. For each tuple $t \in R_k^{I_i}$, if there is no tuple $s \in R_j^{I_i}$ that agrees with t over the values of X, then we remove t altogether. Otherwise, we consider all values such tuples assign y. Denote those values by $a_1, \ldots, a_m$, and note that due to the CD, $m \leq c$. We copy t to $R_k^{I_{i+1}}$ m times, each time assigning y with a different value of $a_1, \ldots, a_m$. Given an answer $\mu \in Q^+(\sigma(I))$, we define $\tau(\mu)$ to be the projection of μ to free(Q).

We need to show that $Q(I) = \{\mu|_{\text{free}(Q)} : \mu|_{\text{free}(Q^+)} \in Q^+(\sigma(I))\}$, and that an element of the left-hand side may only appear a constant amount of times on the right-hand side. First, if $\mu|_{\text{free}(Q^+)} \in Q^+(\sigma(I))$, since all tuples of $\sigma(I)$ appear (perhaps projected) in I, then $\mu|_{\text{free}(Q)} \in Q(I)$. It is left to show the opposite direction: if $\mu|_{\text{free}(Q)} \in Q(I)$ then $\mu|_{\text{free}(Q^+)} \in Q^+(I^+)$. We show by induction on $Q = Q_0, Q_1, \ldots, Q_n = Q^+$ that $\mu|_{\text{free}(Q_i)} \in Q_i(I_i)$. The induction base holds since in the cleaning phase we did not remove "useful" tuples. Since $\mu|_{\text{free}(Q)} \in Q(I)$, there exist tuples, one of each relation of the query, that agree on the values of $X \cup \{y\}$ (they all assign them with the values μ assigns them). These tuples were not removed in the cleaning phase, and therefore $\mu|_{\text{free}(Q)} \in Q(I_0)$. Next assume that $\mu|_{\text{free}(Q_i)} \in Q_i(I_i)$, and we want to show that $\mu|_{\text{free}(Q_{i+1})} \in Q_{i+1}(I_{i+1})$. This claim is trivial in case the head was extended. Note that there can be at most $c - 1$ different answers $\mu'|_{\text{free}(Q_{i+1})}$ in $Q_{i+1}(I_{i+1})$ such that $\mu|_{\text{free}(Q_{i+1})} \neq \mu'|_{\text{free}(Q_{i+1})}$ but $\mu|_{\text{free}(Q_i)} = \mu'|_{\text{free}(Q_i)}$, as the added variable y is bound by the CD to have at most c possible values. Now consider the case where an atom $R_k(\mathbf{v}_k)$ was extended according to a CD $(R_j\colon X \to y, c)$. The tuple $\mu(\mathbf{v}_k) \in R_k^{I_i}$ was extended with the value $\mu(y)$ due to the tuple $\mu(\mathbf{v}_j) \in R_j^{I_i}$ that agrees with it on the values of X, and so $\mu(\mathbf{v}_k, y) \in R_k^{I_{i+1}}$. In case of self-joins, other atoms with the relation R_k are extended with a new and distinct variable. Such variables will be mapped to this value $\mu(y)$ as well. Overall, we have that μ (extended by mappings of the fresh variables) is also a homomorphism in $Q_{i+1}(I_{i+1})$. □

We can now extend our positive results to accommodate CDs. Let Q be a CD-free-connex CQ over a schema $\mathcal{S} = (\mathcal{R}, \Delta)$, where Δ contains CDs. According to Lemma 7, $\text{ENUM}_\Delta \langle Q \rangle \leq_{e'} \text{ENUM}_{\Delta_{Q+}} \langle Q^+ \rangle \leq_e \text{ENUM}_\emptyset \langle Q^+ \rangle$, and due to Theorem 1, $\text{ENUM}_\emptyset \langle Q^+ \rangle \in \text{D}elayC_{lin}$. The class $\text{D}elayC_{lin}$ is closed under this type of reduction. To avoid printing duplicates, we need to store previous results in a lookup table, and verify that a generated result is new before printing it. This alone is not enough, as we can have a long sequence of generating known results, and then the delay between generating new results can be larger than constant. For this reason, we use the known technique of delaying the results, also used by Capelli et. al [8, Proposition 12]. If every answer to $\text{ENUM}_\Delta \langle Q \rangle$ corresponds to at most c answers to $\text{ENUM}_\emptyset \langle Q^+ \rangle$, we save the newly generated results in a queue, and after generating c results we pop and print a result from the queue. This guarantees that the queue is never empty when accessed, and the results are printed with constant delay. Therefore, $\text{ENUM}_\Delta \langle Q \rangle \in \text{D}elayC_{lin}$, and we deduce the following:

Theorem 6 *Let Q be a CD-acyclic CQ with no self-joins over the schema $\mathcal{S} = (\mathcal{R}, \Delta)$, where Δ is a set of CDs.*

- *If Q is CD-free-connex, then $\text{ENUM}_\Delta \langle Q \rangle \in \text{D}elayC_{lin}$.*
- *If Q is not CD-free-connex, then $\text{ENUM}_\Delta \langle Q \rangle \notin \text{D}elayC_{lin}$, assuming that the product of two $n \times n$ boolean matrices cannot be computed in time $\mathcal{O}(n^2)$.*

7 Conjunctive Queries with Disequalities

In this last section, we show that all results in this article apply to CQs with disequalities. To keep the proof ideas clear, we begin by assuming that the schema only contains FDs. We incorporate our results with CDs at the end of this section. A *CQ with disequalities* over a schema $\mathcal{S} = (\mathcal{R}, \Delta)$ is an expression of the form

$$Q(\mathbf{p}) \leftarrow R_1(\mathbf{v}_1), \dots, R_m(\mathbf{v}_m), w_1 \neq w_2, \dots, w_{k-1} \neq w_k$$

where $Q(\mathbf{p}) \leftarrow R_1(\mathbf{v}_1), \dots, R_m(\mathbf{v}_m)$ is a CQ denoted by Q_{base}, and $w_1, \dots, w_k$ are variables in $var(Q)$. We denote $\text{dis}(Q) = \{w_1 \neq w_2, \dots, w_{k-1} \neq w_k\}$. The evaluation is $Q(I) = \{\mu|_{\mathbf{p}} \in Q_{\text{base}}(I) | \forall w_i \neq w_j \in \text{dis}(Q) : \mu(w_i) \neq \mu(w_j)\}$.

The structural definitions of CQs with disequalities depend only on their bases. That is, Q is said to be *acyclic* if Q_{base} is acyclic, and we similarly define *acyclic* and *free-connex* CQs with disequalities. The FD-extension of a CQ with disequalities is denoted by Q^+. The base extends as before, and the disequalities remain the same. That is, $(Q^+)_{\text{base}} = Q^+_{\text{base}}$ is the FD-extension of Q_{base}, and $\text{dis}(Q^+) = \text{dis}(Q)$. We say that Q is *FD-acyclic* if Q_{base} is FD-acyclic, and similarly define *FD-free-connex* and *FD-cyclic* CQs with disequalities.

We first show that our positive results apply also to CQs with disequalities. The following lemma is the equivalent of the positive case of Theorem 2. In the proof, the reduction from the query to its extension works as before for the base query, and the disequalities are satisfied as they remain the same during the extension.

Lemma 8 *Let Q be a CQ with disequalities over a schema $\mathcal{S} = (\mathcal{R}, \Delta)$, and let Q^+ be its FD-extension. Then* $\text{ENUM}_\Delta \langle Q \rangle \leq_e \text{ENUM}_{\Delta_{Q^+}} \langle Q^+ \rangle$.

Proof Recall that Q^+_{base} is the FD-extension of Q_{base}. According to the proof of Theorem 2, we can show that $\text{ENUM}_\Delta \langle Q_{\text{base}} \rangle \leq_e \text{ENUM}_{\Delta_{Q^+}} \langle Q^+_{\text{base}} \rangle$ by using a construction σ such that $\mu|_{\text{free}(Q)} \in Q_{\text{base}}(I)$ iff $\mu|_{\text{free}(Q^+)} \in Q^+_{\text{base}}(\sigma(I))$. We use this same construction to show that $\text{ENUM}_\Delta \langle Q \rangle \leq_e \text{ENUM}_{\Delta_{Q^+}} \langle Q^+ \rangle$. By definition we have that $\mu|_{\text{free}(Q)} \in Q(I)$ if and only if $\mu|_{\text{free}(Q)} \in Q_{\text{base}}(I)$ and in addition $\forall u \neq w \in \text{dis}(Q) : \mu(u) \neq \mu(w)$. As in the proof of Theorem 2, and since $\text{dis}(Q) = \text{dis}(Q^+)$, this is true iff $\mu|_{\text{free}(Q^+)} \in Q^+_{\text{base}}(\sigma(I))$ and $\forall u \neq w \in \text{dis}(Q^+) : \mu(u) \neq \mu(w)$. This is equivalent to $\mu|_{\text{free}(Q^+)} \in Q^+(\sigma(I))$. We conclude that $\mu|_{\text{free}(Q)} \in Q(I)$ if and only if $\mu|_{\text{free}(Q^+)} \in Q^+(\sigma(I))$. $\square$

Combining Lemma 8 with the full version of Theorem 1 that holds for CQs with disequalities [3], we have that if Q is an FD-free-connex CQ with disequalities over a schema $\mathcal{S} = (\mathcal{R}, \Delta)$, then $\text{ENUM}_\Delta \langle Q \rangle \in \mathsf{D}elayC_{lin}$.

We now discuss the lower bound. We formalize an idea proposed by Bagan et al. to extend their negative results to CQs with disequalities, and show that this idea can be applied to reductions in general, and to our case in particular.

Lemma 9 *Let Q be a CQ with disequalities over a schema $\mathcal{S} = (\mathcal{R}, \Delta)$. Then* $\text{ENUM}_\Delta \langle Q_{\text{base}} \rangle \leq_e \text{ENUM}_\Delta \langle Q \rangle$.

Proof Given an instance I of $\text{ENUM}_\Delta \langle Q_{\text{base}} \rangle$, we construct $\sigma(I)$ by assigning every variable a disjoint domain. Formally, for every atom $R(\mathbf{v})$ of Q_{base} and every tuple $(c_1, \ldots, c_t) \in R^I$, we have the tuple $((c_1, \mathbf{v}[1]), \ldots, (c_t, \mathbf{v}[t]))$ in $R^{\sigma(I)}$. We claim that the answers of Q_{base} over I are exactly those of Q over $\sigma(I)$ if we omit the variable names. That is, we define $\tau : dom \times var(Q) \to dom$ as $\tau((c, v)) = c$, and show $Q_{\text{base}}(I) = \tau(Q(\sigma(I)))$.

Let $\mu|_{\text{free}(Q)} \in Q_{\text{base}}(I)$. Then we know that for every atom $R(\mathbf{v})$ of Q_{base}, there exists a tuple $(\mu(\mathbf{v}[1]), \ldots, \mu(\mathbf{v}[t])) \in R_I$. By definition of σ, it results in $((\mu(\mathbf{v}[1]), \mathbf{v}[1]), \ldots, (\mu(\mathbf{v}[t]), \mathbf{v}[t])) \in R^{\sigma(I)}$. Therefore, as we define the mapping $f_\mu : var(Q) \to dom \times var(Q)$ to be $f_\mu(u) = (\mu(u), u)$, we have that $f_\mu|_{\text{free}(Q)} \in Q_{\text{base}}(\sigma(I))$. Note that for all $u, w \in var(Q)$ with $u \neq w$ we have $f_\mu(u) \neq f_\mu(w)$. Therefore, $f_\mu \in Q(\sigma(I))$ as all disequalities in $\text{dis}(Q)$ are satisfied. Since $\tau \circ f_\mu = \mu$, we have that $\mu|_{\text{free}(Q)} \in \tau(Q(\sigma(I)))$, and this concludes that $Q_{\text{base}}(I) \subseteq \tau(Q(\sigma(I)))$. The opposite direction holds as well. If $\nu|_{\text{free}(Q)}$ is in $Q(\sigma(I))$, then for every atom $R(\mathbf{v})$ in Q we have that $\nu(\mathbf{v}) \in R^{\sigma(I)}$. By construction, $\tau(\nu(\mathbf{v})) \in R^I$, and therefore $\tau \circ \nu|_{\text{free}(Q)} \in Q_{\text{base}}(I)$. $\square$

By combining Theorem 2 with Lemma 9, we get the following reduction.

Lemma 10 *Let Q be a CQ with disequalities over a schema $\mathcal{S} = (\mathcal{R}, \Delta)$. Then* $\text{ENUM}_{\Delta_{Q^+}} \langle Q^+_{\text{base}} \rangle \leq_e \text{ENUM}_\Delta \langle Q \rangle$.

Lemma 10 means that the hardness results of this paper extend to CQs with disequalities. In particular, if a self-join-free Q is FD-acyclic but not FD-free-connex, then $\text{ENUM}_\emptyset\langle\Pi\rangle \leq_e \text{ENUM}_{\Delta_{Q^+}}\langle Q_{\text{base}}^+\rangle \leq_e \text{ENUM}_\Delta\langle Q\rangle$ by Lemma 2 and Lemma 10. We get that then $\text{ENUM}_\Delta\langle Q\rangle \notin DelayC_{lin}$, assuming that the product of two $n \times n$ boolean matrices cannot be computed in time $\mathcal{O}(n^2)$. Similarly, we can use Lemma 5 and Lemma 10 to show that if a self-join-free Q is FD-cyclic, and the schema only contains unary FDs, then $\text{DECIDE}_\Delta\langle Q\rangle$ cannot be solved in linear time, assuming that the $\text{TETRA}(k)$ problem is not solvable in linear time for any k.

We next consider CQs with disequalities in the presence of CDs. The ideas presented in Sections 6 and 7 can be combined to show the following lemma.

Lemma 11 *Let Q be a CQ with disequalities over a schema $\mathcal{S} = (\mathcal{R}, \Delta)$, where Δ is a set of CDs. We have:*

– $\text{ENUM}_\Delta\langle Q\rangle \leq_{e'} \text{ENUM}_{\Delta_{Q^+}}\langle Q^+\rangle$
– $\text{ENUM}_{\Delta_{Q^+}^{FD}}\langle Q_{\text{base}}^+\rangle \leq_e \text{ENUM}_\Delta\langle Q\rangle$

The proof for $\text{ENUM}_\Delta\langle Q\rangle \leq_{e'} \text{ENUM}_{\Delta_{Q^+}}\langle Q^+\rangle$ works similarly to that of Lemma 8, except it builds upon Lemma 7 instead of Theorem 2. We use the same reduction as in Lemma 7, and the disequalities remain unchanged. By combining Lemma 9 with Lemma 6, he have that by transitivity $\text{ENUM}_{\Delta_{Q^+}^{FD}}\langle Q_{\text{base}}^+\rangle \leq_e \text{ENUM}_\Delta\langle Q_{\text{base}}\rangle \leq_e \text{ENUM}_\Delta\langle Q\rangle$.

Lemma 11 along with Lemma 2, Lemma 5 and the positive results by Bagan et al. [3, Theorem 13, Theorem 17] prove that all results presented in this article apply to CQs with disequalities over schemas with cardinality dependencies. The following theorem summarizes all classification results presented here, as FDs are a special case of CDs, and CQs are a special case of CQs with disequalities.

Theorem 7 *Let Q be a CQ with disequalities over a schema $\mathcal{S} = (\mathcal{R}, \Delta)$, where Δ is a set of CDs.*

–   *If Q is CD-free-connex, then $\text{ENUM}_\Delta\langle Q\rangle \in DelayC_{lin}$.*
–   *If Q is CD-acyclic, then $\text{ENUM}_\Delta\langle Q\rangle \in DelayLin$.*
–   *If Q is self-join-free, CD-acyclic but not CD-free-connex, then $\text{ENUM}_\Delta\langle Q\rangle \notin DelayC_{lin}$, assuming that the product of two $n \times n$ boolean matrices cannot be computed in time $\mathcal{O}(n^2)$.*
–   *If Q is self-join-free, CD-cyclic and Δ contains only unary CDs, then $\text{DECIDE}_\Delta\langle Q\rangle$ cannot be solved in linear time, assuming that the $\text{TETRA}(k)$ problem is not solvable in linear time for any k. In particular, $\text{ENUM}_\Delta\langle Q\rangle \notin DelayC_{lin}$ and $\text{ENUM}_\Delta\langle Q\rangle \notin DelayLin$.*

8 Concluding Remarks

Previous hardness results regarding the enumeration complexity of CQs no longer hold in the presence of dependencies. In this paper, we have shown that some of the

queries which where previously classified as hard are in fact tractable in the presence of FDs, and that the others remain intractable. We have classified the enumeration complexity of self-join-free CQs according to their FD-extension. Under previously used complexity assumptions: a query is in $DelayC_{lin}$ if its extension is free-connex, it is not in $DelayC_{lin}$ if its extension is acyclic but not free-connex, and it is not even decidable in linear time if the schema has only unary FDs and its extension is cyclic. We also show that these results apply for CQs with disequalities and in the presence of CDs. In addition to our results on constant delay enumeration, we show that this work has consequences in other enumeration classes such as $DelayLin$.

This work opens up quite a few directions for future work. Our proof for the hardness of FD-cyclic CQs assumes that all FDs are unary. The question of whether this result holds for general FDs, along with the classification of Example 6, remains open. In addition, to show that enumerating CD-free-connex CQs can be done in $DelayC_{lin}$, we store all printed results. The required space has the size of the output, which may be polynomial in that of the input in data complexity and exponential in combined complexity. It is unclear whether there exists a solution that requires less space. Finally, we wish to explore how the tools provided here can be used to extend other known results on query enumeration, such as a dichotomy for enumerating CQs with negation [6], to accommodate FDs.

References

1. Abboud, A., Williams, V.V.: Popular conjectures imply strong lower bounds for dynamic problems. In: 2014 IEEE 55Th Annual Symposium on Foundations of Computer Science (FOCS), pp. 434–443. IEEE (2014)
2. Arapinis, M., Figueira, D., Gaboardi, M.: Sensitivity of counting queries. In: 43Rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, pp. 120:1–120:13. Rome (2016)
3. Bagan, G., Durand, A., Grandjean, E.: On acyclic conjunctive queries and constant delay enumeration. In: International Workshop on Computer Science Logic, pp. 208–222. Springer (2007)
4. Beeri, C., Fagin, R., Maier, D., Yannakakis, M.: On the desirability of acyclic database schemes. J. ACM **30**(3), 479–513 (1983). https://doi.org/10.1145/2402.322389
5. Berkholz, C., Keppeler, J., Schweikardt, N.: Answering conjunctive queries under updates. In: Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2017, pp. 303–318. Chicago, May 14-19, 2017 (2017)
6. Brault-Baron, J.: De La Pertinence De Lénumération: Complexitéen Logiques Propositionnelle Et Du Premier Ordre. Ph.D. thesis, Université de Caen (2013)
7. Cao, Y., Fan, W., Wo, T., Yu, W.: Bounded conjunctive queries. PVLDB **7**(12), 1231–1242 (2014)
8. Capelli, F., Strozecki, Y.: Incremental delay enumeration: Space and time. Discrete Applied Mathematics. https://doi.org/10.1016/j.dam.2018.06.038 (2018)
9. Carmeli, N., Kröll, M.: Enumeration complexity of conjunctive queries with functional dependencies. In: Kimelfeld, B., Amsterdamer, Y. (eds.) 21st International Conference on Database Theory (ICDT 2018), Leibniz International Proceedings in Informatics (LIPIcs), vol. 98, pp. 11:1–11:17. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (2018). https://doi.org/10.4230/LIPIcs.ICDT.2018.11. http://drops.dagstuhl.de/opus/volltexte/2018/8598
10. Creignou, N., Kröll, M., Pichler, R., Skritek, S., Vollmer, H.: On the complexity of hard enumeration problems. In: Language and Automata Theory and Applications - 11Th International Conference, LATA 2017, pp. 183–195. Umeå (2017)
11. Durand, A., Schweikardt, N., Segoufin, L.: Enumerating answers to first-order queries over databases of low degree. In: Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS '14, pp. 121–131 (2014)

12. Flum, J., Frick, M., Grohe, M.: Query evaluation via tree-decompositions. J. ACM **49**(6), 716–752 (2002)
13. Grandjean, E.: Sorting, linear time and the satisfiability problem. Ann. Math. Artif. Intell. **16**, 183–236 (1996). https://doi.org/10.1007/BF02127798
14. Kimelfeld, B.: A dichotomy in the complexity of deletion propagation with functional dependencies. In: Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS '12, pp. 191–202. ACM, New York (2012). https://doi.org/10.1145/2213556.2213584
15. Le Gall, F.: Powers of tensors and fast matrix multiplication. In: Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation, ISSAC '14, pp. 296–303. ACM, New York (2014). https://doi.org/10.1145/2608628.2608664
16. Lincoln, A., Williams, V.V., Williams, R.R.: Tight hardness for shortest cycles and paths in sparse graphs. In: Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, pp. 1236–1252. New Orleans, January 7-10, 2018 (2018)
17. Papadimitriou, C.H., Yannakakis, M.: On the complexity of database queries. J. Comput. Syst. Sci. **58**(3), 407–427 (1999)
18. Segoufin, L., Vigny, A.: Constant delay enumeration for FO queries over databases with local bounded expansion. In: 20Th International Conference on Database Theory, ICDT 2017, pp. 20:1–20:16. Venice (2017)
19. Williams, V.V., Williams, R.: Subcubic Equivalences between Path, Matrix and Triangle Problems. In: 51Th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, pp. 645–654. Las Vegas (2010)