Enumeration Complexity of Conjunctive Queries with Functional Dependencies

Nofar Carmeli
Technion, Haifa, Israel
snofca@cs.technion.ac.il

Markus Kröll
TU Wien, Vienna, Austria
kroell@dbai.tuwien.ac.at

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 a common type of FDs. Further conclusions of our development include a dichotomy for enumeration with linear delay, and a dichotomy for CQs with disequalities. Finally, we show that all our results apply to the known class of "cardinality dependencies" that generalize FDs (e.g., by stating an upper bound on the number of genres per movies, or friends per person).

2012 ACM Subject Classification Mathematics of computing → Enumeration, Theory of computation → Database query languages (principles)

Keywords and phrases Enumeration, Complexity, CQs

Digital Object Identifier 10.4230/LIPIcs.ICDT.2018.11

Related Version A full version of the paper is available at [8], `https://arxiv.org/abs/1712.07880`.

Funding This work was supported by the German-Israeli Foundation for Scientific Research and Development (GIF), Grant no. I-2436-407.6/2016 and by the Austrian Science Fund (FWF): W1255-N23, P25207-N23, P25518-N23.

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 before printing the first result, as linear time preprocessing is necessary to read the input instance.

21st International Conference on Database Theory (ICDT 2018).
Editors: Benny Kimelfeld and Yael Amsterdamer; Article No. 11; pp. 11:1–11:17
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

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 $\mathsf{DelayC}_{\mathsf{lin}}$, can be regarded as the most efficient class of nontrivial enumeration problems and therefore current work on query enumeration has focused on this class [10, 15, 5]. Bagan et al.[3] show that a subclass of acyclic queries, called *free-connex*, are exactly those that are enumerable in $\mathsf{DelayC}_{\mathsf{lin}}$, 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 [16].

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, we have the query: $Q(\mathit{actor}, \mathit{production}) \leftarrow \mathrm{Cast}(\mathit{movie}, \mathit{actor}), \mathrm{Release}(\mathit{movie}, \mathit{production})$. At first glance, it appears as though this query is not in $\mathsf{DelayC}_{\mathsf{lin}}$, 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 $\mathrm{Release} : \mathit{movie} \rightarrow \mathit{production}$, 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 sorting (in linear time [12]) both relations according to *movie*. ◀

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 some FD. 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 [13]. 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 taking FD-extensions. Hence, the class of all queries Q such that Q^+ is free-connex is an extension of the class of free-connex queries, and this extension is in fact proper. We denote the classes of queries Q such that Q^+ is acyclic or free-connex as FD-acyclic respectively FD-free-connex.

To reach a dichotomy, we now need to answer the following question: Is it possible that Q can be enumerated efficiently even if Q^+ is not 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 [9]. 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 $\mathsf{DelayC_{lin}}$ 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 $\mathsf{W}[1]$-hard [14]. This does not imply, however, that there are no cyclic queries with the corresponding enumeration problems in $\mathsf{DelayC_{lin}}$. The fact that no such queries exist requires an additional proof, which was presented by Brault-Baron [6]. This result holds under a generalization of the triangle finding problem, which is considered not to be solvable within linear time [17]. 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 $\mathsf{DelayC_{lin}}$. 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 $\mathsf{DelayC_{lin}}$ is impossible in that case. This covers all types of CQs and shows a full dichotomy, at least for the case of unary FDs.

The results we present here are not limited to FDs. CDs (Cardinality Dependencies) [7, 2] are a generalization of FDs, denoted $(R_i : A \to B, c)$. 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 and there are at most 200 countries. We show that all results described in this paper also apply to CDs. Moreover, we show how our results can be easily used to yield additional results, such as a dichotomy for CQs with disequalities, and a dichotomy to evaluate CQs with linear delay.

Contributions. Our main contributions are as follows.

- We extend the class of queries that can be evaluated in $\mathsf{DelayC_{lin}}$ by incorporating the 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 under 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.
- We extend our results to CDs.

This work is organized as follows: In Section 2 we provide definitions and results that we will use. Section 3 introduces FD-extended queries and establishes the equivalence between a query and its FD-extension. The generalized version of the dichotomy is shown in Section 4. In Section 5, a lower bound for cyclic queries under unary FDs is shown, and Section 6 shows that all results from the previous sections extend to CDs. Concluding remarks are given in Section 7. All missing proof details can be found in the full version of this article [8].

2 Preliminaries

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

Schemas and Functional Dependencies. A *schema* $\mathcal{S}$ is a pair $(\mathcal{R}, \Delta)$ where $\mathcal{R}$ is a finite set $\{R_1, \dots, R_n\}$ of *relational symbols* and Δ is a set of *Functional Dependencies* (FDs). We denote the *arity* of a relational symbol R_i as $\mathrm{arity}(R_i)$. 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 $A, B \subseteq \{1, \dots, \mathrm{arity}(R_i)\}$.

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 $R_i^I \subseteq dom^{\mathrm{arity}(R_i)}$ 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. Here we assume that all FDs are of the form $R_i \colon A \to b$, where $b \in \{1, \dots, \mathrm{arity}(R_i)\}$, 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.

Conjunctive Queries. Let *var* be a set of variables disjoint from *dom*. A *Conjunctive Query* (CQ) over a schema $\mathcal{S} = (\mathcal{R}, \Delta)$ is an expression of the form $Q(\vec{x}) \leftarrow R_1(\vec{v}_1), \dots, R_m(\vec{v}_m)$, where $R_1, \dots, R_m$ are relational symbols of $\mathcal{R}$, the tuples $\vec{x}, \vec{v}_1, \dots, \vec{v}_m$ hold variables, and every variable in $\vec{x}$ appears in at least one of $\vec{v}_1, \dots, \vec{v}_m$. We often denote this query as $Q(\vec{x})$ or even Q. Define the variables of Q as $var(Q) = \bigcup_{i=1}^{m} \vec{v}_i$, and define the *free variables* of Q as $\mathrm{free}(Q) = \vec{x}$. We call $Q(\vec{x})$ the head of Q, and the atomic formulas $R_i(\vec{v}_i)$ are called *atoms*. We further use $\mathrm{atoms}(Q)$ 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 $\vec{x}$ over a database I, we define $Q(I)$ to be the set of all *mappings* $\mu|_{\vec{x}}$ such that μ is a homomorphism from $R_1(\vec{v}_1), \dots, R_m(\vec{v}_m)$ into I, where $\mu|_{\vec{x}}$ denotes the restriction (or projection) of μ to the variables $\vec{x}$. The problem $\textsc{Decide}_\Delta\langle Q \rangle$ is, given a database instance I, determining whether such a mapping exists.

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 $A = \{a_1, \dots, a_{|A|}\}$, we sometimes denote it by $R_i \colon \{\vec{v}_i[a_1], \dots, \vec{v}_i[a_{|A|}]\} \to \vec{v}_i[b]$, where $\vec{u}[k]$ is the k-th variable of $\vec{u}$. To distinguish between these two representations, we usually denote subsets of integers by $A, B, C, \dots$, integers by $a, b, c, \dots$, and variables by letters from the end of the alphabet.

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, namely: 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 $(x_1, \ldots, x_n)$ 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.

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. Bagan et al. [3] showed that an acyclic CQ has a head-path iff it is not free-connex.

Enumeration Complexity. Given a finite alphabet Σ and binary relation $R \subseteq \Sigma^* \times \Sigma^*$, we denote by $\textsc{Enum}\langle R\rangle$ the *enumeration problem* of given an instance $x \in \Sigma^*$, to output all $y \in \Sigma^*$ such that $(x, y) \in R$. In this paper we adopt the *Random Access Machine* (RAM) model (see [16]). Previous results in the field assume different variations of the RAM model. Here we assume that the length of memory registers is linear in the size of value registers, that is, the accessible memory is polynomial. For a class $\mathcal{C}$ of enumeration problems, we say that $\textsc{Enum}\langle R\rangle \in \mathcal{C}$, 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 $\textsc{Enum}\langle R\rangle \in \mathsf{DelayC_{lin}}$, we have $p(|x|) \in O(|x|)$ and $d(|x|) \in O(1)$.
- For $\textsc{Enum}\langle R\rangle \in \mathsf{DelayLin}$, we have $p(|x|), d(|x|) \in O(|x|)$.

Let $\textsc{Enum}\langle R_1\rangle$ and $\textsc{Enum}\langle R_2\rangle$ be enumeration problems. We say that there is an *exact reduction* from $\textsc{Enum}\langle R_1\rangle$ to $\textsc{Enum}\langle R_2\rangle$, written as $\textsc{Enum}\langle R_1\rangle \leq_e \textsc{Enum}\langle R_2\rangle$, if there are mappings σ and τ such that for every $x \in \Sigma^*$ the mapping $\sigma(x)$ is computable in $O(|x|)$, for every $y \in \Sigma^*$ with $(\sigma(x), y) \in R_2$, $\tau(y)$ is computable in constant time and $\{\tau(y) \mid y \in \Sigma^* \text{ with } (\sigma(x), y) \in R_2\} = \{y' \in \Sigma^* \mid (x, y') \in R_1\}$ in multiset notation. Intuitively, σ is used to map instances of $\textsc{Enum}\langle R_1\rangle$ to instances of $\textsc{Enum}\langle R_2\rangle$, and τ is used to map solutions to $\textsc{Enum}\langle R_2\rangle$ to solutions of $\textsc{Enum}\langle R_1\rangle$. An enumeration class $\mathcal{C}$ is said to be *closed under exact reduction* if for every $\textsc{Enum}\langle R_1\rangle$ and $\textsc{Enum}\langle R_2\rangle$ such that $\textsc{Enum}\langle R_1\rangle \leq_e \textsc{Enum}\langle R_2\rangle$ and $\textsc{Enum}\langle R_2\rangle \in \mathcal{C}$, we have $\textsc{Enum}\langle R_1\rangle \in \mathcal{C}$. Bagan et al. [3] proved that $\mathsf{DelayC_{lin}}$ 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.

Enumerating Answers to CQs. For a CQ Q over a schema $\mathcal{S} = (\mathcal{R}, \Delta)$, we denote by $\textsc{Enum}_\Delta\langle Q\rangle$ the enumeration problem $\textsc{Enum}\langle R\rangle$, 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 $\mathsf{DelayC_{lin}}$ iff it is free-connex:

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

3 FD-Extended CQs

In this section, we formally define the extended query Q^+. We then 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.

We first define Q^+. The *extension* of an atom $R(\vec{v})$ according to an FD $S\colon A \to b$ where $S(\vec{u}) \in \text{atoms}(Q)$ is possible if $\vec{u}[A] \subseteq \vec{v}$ but $\vec{u}[b] \notin \vec{v}$. In that case, $\vec{u}[b]$ 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 dummy variables are added to adjust to that change in case of self-joins. The FDs apply in every relation that contains all relevant variables.

▶ **Definition 3.** [(FD-Extended Query)] Let $Q(\vec{w}) \leftarrow R_1(\vec{v_1}), \dots, R_m(\vec{v_m})$ be a CQ over a schema $\mathcal{S} = (\mathcal{R}, \Delta)$. We define two types of extension steps:

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

The *FD-extension* of Q is the query $Q^+(\vec{y}) \leftarrow R_1^+(\vec{u_m}), \dots, R_m^+(\vec{u_m})$, 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 $\Delta_{Q^+} = \{R_i^+\colon C \to d \mid \exists (R_j\colon A \to b) \in \Delta \text{ s.t. } \vec{u_i}[C] = \vec{v_j}[A] \text{ and } \vec{u_i}[d] = \vec{v_j}[b]\}$.

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 4.** Consider a schema with $\Delta = \{R_1\colon 1 \to 2, R_3\colon 2,3 \to 1\}$, 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 $x \to y$ on the head, and then $x \to y$ and consequently $yz \to w$ on $R_2(x,z)$. These two FDs now appear in the schema also for R_2, and the FDs of the extended schema are $\Delta_{Q^+} = \{R_1^+\colon 1 \to 2, R_2^+\colon 1 \to 3, R_2^+\colon 3,2 \to 4, R_3^+\colon 2,3 \to 1\}$. ◀

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. Therefore, we introduce the notions of acyclic and free-connex queries for FD-extensions:

▶ **Definition 5.** 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 6.** *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.*

Example 1 shows that the converse of the proposition above does not hold. This means that, by Theorem 2, there are queries Q such that we can enumerate the answers to Q^+ in $\mathsf{DelayC_{lin}}$, but we cannot enumerate the answers to Q with the same complexity, if we do not assume the FDs. The following lemma shows that enumerating the answers of Q (when relying on the FDs) is in fact equally hard as enumerating the answers of Q^+.

▶ **Theorem 7.** *Let Q be a CQ over a schema $\mathcal{S} = (\mathcal{R}, \Delta)$, and let Q^+ be its FD-extended query. Then* $\textsc{Enum}_\Delta\langle Q\rangle \leq_e \textsc{Enum}_{\Delta_{Q^+}}\langle Q^+\rangle$ *and* $\textsc{Enum}_{\Delta_{Q^+}}\langle Q^+\rangle \leq_e \textsc{Enum}_\Delta\langle Q\rangle$.

Proof Sketch. We first sketch the reduction $\textsc{Enum}_\Delta\langle Q\rangle \leq_e \textsc{Enum}_{\Delta_{Q^+}}\langle Q^+\rangle$. Given an instance I for the problem $\textsc{Enum}_\Delta\langle Q\rangle$, we set $\sigma(I) = I^+$ as described next. We start by removing tuples that interfere with the extended dependencies. For every dependency $R_j\colon X \to y$ and every atom $R_k(\vec{v_k})$ that contains the variables $X \cup \{y\}$, 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 in each step we extend some R_i^I to $R_i^{I'}$ according to some FD $R_j\colon X \to y$. For each tuple $t \in R_i^I$, if there is no tuple $s \in R_j^I$ that agrees with t over the values of X, then we remove t altogether. Otherwise, we copy t to $R_i^{I'}$ and assign y with the same value that s assigns it. Given an answer $\mu \in Q^+(\sigma(I))$, we set $\tau(\mu)$ to be the projection of μ to $\text{free}(Q)$. To show that $\textsc{Enum}_{\Delta_{Q^+}}\langle Q^+\rangle \leq_e \textsc{Enum}_\Delta\langle Q\rangle$, we describe the construction of an instance $\sigma(I^+)$ by "reversing" the extension steps. If an atom was extended, we simply remove the added attribute. If the head was extended using some $R_j\colon X \to y$, then for each tuple in $R_j^{I^{i+1}}$ that assigns y and X with the values y_0 and $\vec{x}_0$ respectively, we add the value y_0 to a lookup table with pointer $(X, \vec{x}_0, y)$. For every $\mu \in Q(\sigma(I^+))$, $\tau(\mu)$ is defined as μ extended by the values from the lookup table. ◀

The direction $\textsc{Enum}_\Delta\langle Q\rangle \leq_e \textsc{Enum}_{\Delta_{Q^+}}\langle Q^+\rangle$ of Theorem 7 proves that FD-extensions can be used to expand tractable enumeration classes, as the following corollary states.

▶ **Corollary 8.** *Let $\mathcal{C}$ be an enumeration class that is closed under exact reduction. Let Q be a CQ and let Q^+ be its FD-extended query. If* $\textsc{Enum}_{\Delta_{Q^+}}\langle Q^+\rangle \in \mathcal{C}$*, then* $\textsc{Enum}_\Delta\langle Q\rangle \in \mathcal{C}$.

Since free-connex queries are in $\mathsf{DelayC_{lin}}$ and $\mathsf{DelayC_{lin}}$ is closed under exact reduction, if Q is an FD-free-connex query, then the corresponding enumeration problem is in $\mathsf{DelayC_{lin}}$. This follows from Theorem 2 and the fact that $\textsc{Enum}_{\Delta_{Q^+}}\langle Q^+\rangle \leq_e \textsc{Enum}_\emptyset\langle Q^+\rangle$.

▶ **Corollary 9.** *Let Q be a CQ over a schema $\mathcal{S} = (\mathcal{R}, \Delta)$. If Q is FD-free-connex, then* $\textsc{Enum}_\Delta\langle Q\rangle \in \mathsf{DelayC_{lin}}$.

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, disregarding the FDs, according to Theorem 2 it is not in $\mathsf{DelayC_{lin}}$. However, given $R_2\colon z \to y$, the FD-extended query is $Q^+(x, y) \leftarrow R_1^+(z, y, x), R_2^+(z, y)$. As it is free-connex, enumerating Q^+ is in $\mathsf{DelayC_{lin}}$ by Corollary 9.

4 A Dichotomy for Acyclic CQs

In this section, we characterize which self-join-free FD-acyclic queries are in $\mathsf{DelayC_{lin}}$. We use the notion of FD-extended queries defined in the previous section to establish a dichotomy stating that enumerating the answers to an FD-acyclic query is in $\mathsf{DelayC_{lin}}$ iff the query is FD-free-connex. We will prove the following theorem:

▶ **Theorem 10.** *Let Q be an FD-acyclic CQ without self-joins over a schema $\mathcal{S} = (\mathcal{R}, \Delta)$.*

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

The positive case for the dichotomy was described in Corollary 9. Note that the restriction of considering only self-joins-free queries is required only for the negative side. This assumption is standard [3, 6, 13], 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 enumeration $\textsc{Enum}_\emptyset\langle\Pi\rangle$ of the query $\Pi(x, y) \leftarrow A(x, z), B(z, y)$ over the schema $(\{A, B\}, \emptyset)$ where $A, B \subseteq \{1, \dots, n\}^2$. It is strongly conjectured that this problem is not computable in $O(n^2)$ time and currently, the best known algorithms require $O(n^\omega)$ time for some $2.37 < \omega < 2.38$ [11, 1].

The original proof describes an exact reduction $\textsc{Enum}_\emptyset\langle\Pi\rangle \leq_e \textsc{Enum}_\emptyset\langle Q\rangle$. Since Q is acyclic but not free-connex, it contains a head-path $(x, z_1, \dots, z_k, y)$. Given an instance of the matrix multiplication problem, an instance of $\textsc{Enum}_\emptyset\langle Q\rangle$ is constructed, where the variables x,y and $z_1, \dots, z_k$ of the head-path respectively encode the variables x, y and z of Π, while 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} only 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 after linear time preprocessing would result in the computation of the answers of Π in $O(n^2)$ 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 11.** 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 $\mathsf{DelayC_{lin}}$, 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 $\textsc{Enum}_\Delta\langle Q\rangle$. ◀

We now give a detailed sketch of a modification of this construction that shows that $\textsc{Enum}_\emptyset\langle\Pi\rangle \leq_e \textsc{Enum}_{\Delta_{Q^+}}\langle Q^+\rangle$. 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 explanation of the reduction, we start by describing a restricted case, where all FDs are unary. The basic idea in the case of general FDs will remain the same, but it will require 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 be the sets of variables that iteratively imply x, y and some z_i respectively. That is, for $\alpha \in \{x, y, z_1, \ldots, z_k\}$ we first set $V_\alpha := \{\alpha\}$, and then apply $V_\alpha := V_\alpha \cup \{\gamma \in \mathit{var}(Q) \mid \gamma \to \beta \in \Delta_{Q^+} \wedge \beta \in V_\alpha\}$ until a fixpoint is reached. We then define $V_z := V_{z_1} \cup \cdots \cup V_{z_k}$.

The Reduction. Let $I = (A^I, B^I)$ be an instance of $\textsc{Enum}_\emptyset\langle\Pi\rangle$. In order to define $\sigma(I)$, we describe how to construct the relation R^I for every atom $R(\vec{v}) \in \mathrm{atoms}(Q^+)$. If $\mathit{var}(R) \cap V_y = \emptyset$, 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 $R^{\sigma(I)} = \{(f(v_1, a, c), \ldots, f(v_k, a, c)) \mid (a, c) \in A^I\}$, where:

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

Otherwise, $\mathit{var}(R) \cap V_y \neq \emptyset$, and we show that $\mathit{var}(R) \cap V_x = \emptyset$. 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 12.** Consider the FD-extended query $Q^+(x, y, v) \leftarrow R(u, x, z), S(v, y, z)$ with $\Delta_{Q^+} = \{R\colon u \to x, R\colon u \to z, S\colon y \to v\}$. Using the head-path (x, z, y), the reduction will set $V_x = \{x, u\}$, $V_y = \{y\}$, $V_z = \{z, u\}$. Given an instance of the matrix multiplication problem with relations A and B, every tuple $(a, c) \in A$ will result in a tuple $((a, c), a, c) \in R$, and every tuple $(c, b) \in B$ will result in a tuple $(\perp, b, c) \in S$. ◀

We now outline the correctness of this reduction:

Well-defined reduction: For an atom R, either we have $\mathit{var}(R) \cap V_y = \emptyset$ or $\mathit{var}(R) \cap V_x = \emptyset$. 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 $\perp$, 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 would appear in the head of Q^+, then by the definition of Q^+, some z_i will be in the head as well. This cannot happen according to the definition of a head-path. Therefore, the head only encodes the x and y values of the matrix multiplication problem, so two different solutions to $\textsc{Enum}_{\Delta_{Q^+}}\langle Q^+\rangle$ must differ in either x or y, and correspond to different solutions of $\textsc{Enum}_\emptyset\langle\Pi\rangle$. For the other direction, the head necessarily contains the variables x and y. Therefore, two different solutions to $\textsc{Enum}_\emptyset\langle\Pi\rangle$ also correspond to different solutions of $\textsc{Enum}_{\Delta_{Q^+}}\langle Q^+\rangle$.

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

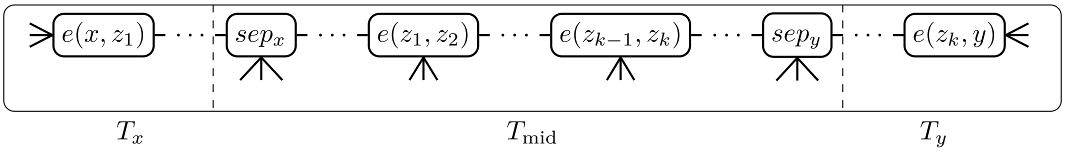

Figure 1 Join tree T of $\mathcal{H}(Q^+)$ for head-paths of length greater than 3. The subtrees T_x, T_y and T_{mid} are disjoint, and are separated by the nodes sep_x and sep_y.

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 13.** 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 $R_2\colon t_1t_2 \to y$. 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 a way of carefully picking the variables to which we assign roles, such that all FDs hold and yet the instance can be constructed in linear time. The idea is that we consider the join-tree of Q^+ and define V_x and V_y to hold variables that appear only in disjoint parts of this tree. This ensures that no atom contains variables of each. The property of a join-tree is used to guarantee that V_x and V_y are inclusive enough to correct all FD violations.

▶ **Lemma 14.** *Let Q be a CQ with no self-joins over a schema $\mathcal{S} = (\mathcal{R}, \Delta)$, such that Q^+ is acyclic but not free-connex. Denote a head-path of Q^+ by $(x, z_1, \ldots, z_k, y)$. Then there exist sets of variables V_x, V_y, V_z such that:*

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

Proof Sketch. We first define a partition of the atoms of Q into three sets: T_x, T_y and T_{mid}, where T_{mid} may be empty. 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. By the running intersection property of T, 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 node 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 node on P that does not contain y. Let T_x be the set of nodes v in T such that the unique path from v to $e(x,z_1)$ does not go through sep_x. Similarly, let T_y be the set of nodes w in T such that the unique path from w to $e(z_k,y)$ does not go through sep_y. Next set $T_{\text{mid}} = V(T) \setminus (T_x \cup T_y)$. Note that the nodes of T are exactly $T_x \cup T_{\text{mid}} \cup T_y$, and we can show that this union is disjoint (see Figure 1). Also note that $e(x,z_1) \in T_x$ and $e(z_k,y) \in T_y$, but T_{mid} may be empty if the head-path is of length three. Therefore, we established a partition of the atoms to two or three sets.

Next we define the sets of variables V_x, V_y and V_z. To do so, for $w \in \mathit{var}(Q)$, denote $\mathsf{Implies}(w) = \{u \in \mathit{var}(Q) \mid u \in U \text{ with } U \to w \in \Delta_{Q^+}\}$. Intuitively, $\mathsf{Implies}(w)$ is the set of all variables on the left-hand side of FDs that have w on the right-hand side. We now define V_x to contain x, and recursively to contain variables that imply those of V_x, but we do not take variables that appear outside of T_x. V_y is defined symmetrically. V_z is defined to contain $z_1, \dots, z_k$, and recursively contain variables that imply those of V_z, but now we do not take variables that appear in the head of the query.

More formally, we recursively define:

- $\boldsymbol{V_x}$: Base $V_x := \{x\}$; Rule $V_x := V_x \cup \{t \in \mathsf{Implies}(w) \mid w \in V_x\} \setminus \mathit{var}(T_y \cup T_{\text{mid}})$
- $\boldsymbol{V_y}$: Base $V_y := \{y\}$; Rule $V_y := V_y \cup \{t \in \mathsf{Implies}(w) \mid w \in V_y\} \setminus \mathit{var}(T_x \cup T_{\text{mid}})$
- $\boldsymbol{V_z}$: Base $V_z := \{z_1, \dots z_k\}$; Rule $V_z := V_z \cup \{t \in \mathsf{Implies}(w) \mid w \in V_z\} \setminus \text{free}(Q^+)$

We now prove that V_x, V_y and V_z meet the requirements of the lemma.

1. The first claim is immediate from the definition of the sets.
2. We first show the claim for $\alpha = x$. Let $\delta = U \to v \in \Delta_{Q^+}$, and let $e(U, v)$ be 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 $u \in \mathit{var}(e_u)$ 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, $u \in \mathit{var}(\mathit{sep}_x)$. Since this is true for all for all $u \in U$, it follows that $v \in \mathit{var}(\mathit{sep}_x)$ by definition of Q^+, contradicting the fact that $v \in V_x$. The case $\alpha = y$ is symmetric.
   Now for the case where $\alpha = z$. If $U \cap V_z = \emptyset$, then $U \subseteq \text{free}(Q^+)$, and by the definition of Q^+, $z_i \in \text{free}(Q^+)$, which is a contradiction to the fact that $v \in \mathit{var}(Q) \setminus \text{free}(Q^+)$.
3. Let $R \in \text{atoms}(Q^+)$. If $R \in T_x$, then by definition of V_y we have that $\mathit{var}(R) \cap V_y = \emptyset$. Otherwise, $R \in T_y \cup T_{\text{mid}}$, and similarly $\mathit{var}(R) \cap V_x = \emptyset$.
4. By definition of V_z, it does not contain any variables of $\text{free}(Q^+)$. ◀

With the sets V_x, V_y, V_z at hand, we can now perform the reduction between the two problems for general FDs. The reduction is based on the case of unary FDs, but with the sets defined according to Lemma 14. Requirements 1 and 4 on the sets guarantee a one-to-one mapping between the results of the two problems, requirement 2 guarantees that all FDs are preserved, and requirement 3 guarantees linear time construction.

▶ **Lemma 15.** *Let Q be a CQ with no self-joins over a schema $\mathcal{S} = (\mathcal{R}, \Delta)$. If Q^+ is acyclic and not free-connex, then* $\textsc{Enum}_{\emptyset}\langle\Pi\rangle \leq_e \textsc{Enum}_{\Delta_{Q^+}}\langle Q^+\rangle$.

This lemma, along with Theorem 7, establishes the hardness result in Theorem 10. This result does not contradict the dichotomy given in Theorem 2: If for a given query Q we have that Q^+ is acyclic but not free-connex, then Q cannot be free-connex by Proposition 6.

Note that Theorem 10, just like the dichotomy presented by Bagan et al. [3], also applies for CQs with disequalities. The extension for such a query is performed as before, ignoring the disequalities. The equivalence described in Theorem 7 still holds, and the proof remains intrinsically the same. The proof of the hardness result presented here also remains similar, with the sole difference that during the construction we take a different and disjoint domain for each variable. This guarantees that all possible disequalities are preserved.

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 $\mathsf{DelayC_{lin}}$ [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:

▶ **Theorem 16.** *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-cyclic, then $\textsc{Decide}_\Delta\langle Q\rangle$ cannot be solved in linear time, assuming that the $\textsc{Tetra}(k)$ problem cannot be solved in linear time for any k.*

As before, the initial hardness proof for cyclic queries no longer holds in the presence of FDs, and we modify the reduction to fix any violations of the FDs. We start by describing the assumption used to obtain the conditional lower bounds. We define $\textsc{Tetra}(k)$ to be the hypergraph with the vertices $\{1, \dots, k\}$ and the edges $\{\{1, \dots, k\} \setminus \{i\} \mid i \in \{1, \dots, k\}\}$. Let $\mathcal{H}$ be a hypergraph. With a slight abuse of notation, we also denote by $\textsc{Tetra}(k)$ the decision problem of whether $\mathcal{H}$ contains a subhypergraph isomorphic to $\textsc{Tetra}(k)$. Note that $\textsc{Tetra}(3)$ is the problem of deciding whether a graph contains a triangle, which is strongly believed to be not solvable within time linear in the size of the graph [17]. The generalization of this assumption is that the $\textsc{Tetra}(k)$ problem cannot be solved in time linear in the size of the graph for any k. This is a stronger assumption than we used in Section 4, as the $\textsc{Tetra}(3)$ can be reduced to the matrix multiplication problem [17]. We will show that if Q^+ is cyclic and only unary FDs are present, the problem $\textsc{Tetra}(k)$ for some k can be reduced to $\textsc{Decide}_{\Delta_{Q^+}}\langle Q^+\rangle$.

▶ **Definition 17.** Let $\mathcal{H}$ be a cyclic hypergraph. We denote by $\mathsf{Tet_{pm}}(\mathcal{H})$ the pseudo-minors of $\mathcal{H}$ isomorphic to $\textsc{Tetra}(k)$ for some k, which are obtained in one of the following ways:

1. Vertex removal steps followed by all possible edge removals.
2. Vertex and edge removal steps that lead to a chordless cycle, followed by edge contraction and edge removal steps that result in a $\textsc{Tetra}(3)$.

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

Brault-Baron [6] showed that if $\mathcal{H}$ is cyclic, then $\mathsf{Tet_{pm}}(\mathcal{H}) \neq \emptyset$. This proof is provided in the full version of this paper. For the reduction we will 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 $X \to y$. Here, we assume that Δ only contains non-trivial FDs, meaning $y \notin X$.

▶ **Lemma 18.** *Let Q be an FD-cyclic CQ with no self-joins over a schema $\mathcal{S} = (\mathcal{R}, \Delta)$. For every $\mathcal{H}_{pm} = (V, E) \in \mathsf{Tet_{pm}}(Q^+)$ and non-trivial $X \to y \in \Delta_{Q^+}$, we have $X \cup \{y\} \nsubseteq V$.*

Proof Sketch. Assume by contradiction that the variables of the FD $\delta = X \to y$ are all part of the pseudo-minor $\mathcal{H}_{pm}$. Note that the variables $X \cup \{y\}$ must appear in a common edge that corresponds to the atom that defines δ. We distinguish between two cases. If $\mathcal{H}_{pm}$ is obtained only by vertex removal and edge removal steps, then by the definition of $\textsc{Tetra}(k)$ it also contains an edge e with $X \subseteq e$ and $y \notin e$. However, this contradicts the fact that Q^+ is an FD-extension, as every edge containing X must also contain y. The other case is that $\mathcal{H}_{pm}$ is a TETRA(3) obtained by edge contraction steps performed on a cycle C. Then $X \cup \{y\}$ is contained in a single edge in C, as none of the vertices $X \cup \{y\}$ have been deleted. Thus, we have that $|X| = 1$ and we can denote $X = \{x\}$. As C is a cycle, it contains an edge e with $x \in e$ and $y \notin e$, which contradicts the fact that Q^+ is an FD-extension. ◀

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 19.** *Let Q be an FD-cyclic CQ with no self-joins over a schema $\mathcal{S} = (\mathcal{R}, \Delta)$, where Δ only contains unary FDs. Let $\mathcal{H}_{pm} \in \mathsf{Tet_{pm}}(Q^+)$ be a pseudo-minor of $\mathcal{H}(Q^+)$ isomorphic to* TETRA(k)*. Then,* TETRA(k) $\leq_m$ DECIDE$_{\Delta_{Q^+}}\langle Q^+\rangle$*, and this reduction can be computed in linear time.*

Proof Sketch. 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 $\mathcal{H}(Q^+) = \mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_t = \mathcal{H}_{pm}$ of pseudo-minors, each one obtained by performing one operation over the previous one. We define the instance I inductively, by first generating relations that correspond to the edges of $\mathcal{H}_{pm}$, and then "reversing" the operations. For every edge e of $\mathcal{H}_{pm}$, we define a relation R_e^t that contains all edges of $\mathcal{G}$ that have the same size as e. We then construct the relations R_e^i of $\mathcal{H}_i$ given the relations R_e^{i+1} of $\mathcal{H}_{i+1}$. We make the following case distinction: If an edge e was removed as some e' contains it, then the relation R_e is added as a projection of $R_{e'}$. If $\mathcal{H}_{i+1}$ is obtained from $\mathcal{H}_i$ by an edge contraction in which a vertex v is replaced by u, then the values corresponding to u in every tuple are copied to the index of v. If a vertex v is removed, then it is assigned with a constant value, and then the following steps are performed on every tuple to correct any FD violations. First, the values of all variables implied by v are concatenated to its value, and then the new value of v is concatenated to all variables implying it. Since Q^+ is an FD-extension, and since only unary FDs are present, we can conclude that whenever a vertex is removed, if x implies y, then y is present in every edge containing x. This fact guarantees that the FD-correction steps can be performed. This construction defines relations that correspond to $\mathcal{H}(Q^+)$, which form I in such a way that $\mathcal{G}$ has a subhypergraph isomorphic to $\mathcal{H}_{pm}$ iff $Q^+(I) \neq \emptyset$. Compliance to any FDs included in $\mathcal{H}_i$ is shown by induction on the sequence, and the induction base holds trivially due to Lemma 18. ◀

Theorem 16 is an immediate consequence of Lemma 19. As in the previous section, by taking a disjoint domain for every variable in the proof of Lemma 19, Theorem 16 also holds for CQs with disequalities. In terms of enumeration complexity, Theorem 16 means that any enumeration algorithm for the answers of such a query cannot output a first solution (or decide that there is none) within linear time, and we get the following corollary.

▶ **Corollary 20.** *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-cyclic, then* ENUM$_\Delta\langle Q\rangle \notin \mathsf{DelayC_{lin}}$*, assuming that the* TETRA(k) *problem cannot be solved 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. A lower bound for this class can be achieved similarly to Corollary 20. Regarding tractability, as acyclic CQs are in DelayLin [3], we conclude from Corollary 8 that FD-acyclic CQs are in this class as well. Thus, we obtain a dichotomy stating that CQs are in DelayLin iff they are FD-acyclic.

▶ **Theorem 21.** *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 $\textsc{Enum}_\Delta\langle Q\rangle \in \mathsf{DelayLin}$.*
- *Otherwise (if Q is FD-cyclic), $\textsc{Enum}_\Delta\langle Q\rangle \notin \mathsf{DelayLin}$, assuming that the $\textsc{Tetra}(k)$ problem cannot be solved in linear time for any k.*

We conclude this section with a short discussion about the extension of our results to general FDs. The following example shows that the proof for Theorem 16 that was provided here cannot be lifted to general FDs. Exploring this extension is left for future work.

▶ **Example 22.** Consider the query $Q() \leftarrow R_1(x, y, u), R_2(x, w, z), R_3(y, v, z), R_4(u, v, w)$, over a schema with all possible two-to-one FDs in the relations R_1, R_2 and R_3. That is, $\Delta = \{xy \to u, yu \to x, ux \to y, zy \to v, yv \to z, vz \to y, xz \to w, zw \to x, wx \to z\}$. Note that $Q^+ = Q$. The hypergraph $\mathcal{H}(Q^+)$ is cyclic, yet it is unclear whether Q can be solved in linear time, and whether $\textsc{Tetra}(3)$ can be reduced to answering Q^+. Using Lemma 18, $\mathcal{H}(Q^+)$ 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 19 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 last section, we show that the results of this paper 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 $(R_i\colon A \to B, c)$, where $R_i\colon A \to B$ 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, 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 $\mathcal{S} = (\mathcal{R}, \Delta^{\mathrm{FD}})$. The schema $\mathcal{S}^+$ is defined with the original c values, and the CDs are $\Delta_{Q^+} = \{(R_i^+\colon A \to b, c) \mid \exists (R_j\colon A \to B, c) \in \Delta, b \in B, A \cup \{b\} \subseteq \mathit{var}(R_i^+)\}$. Note that FD-extensions are indeed a special case of CD-extensions.

The hardness results extend to CDs because FDs are a special case of CDs. Since every instance that preserves the FDs Δ^{FD} also preserves the CDs Δ, we can conclude that $\textsc{Enum}_{\Delta^{\mathrm{FD}}}\langle Q\rangle \leq_e \textsc{Enum}_\Delta\langle Q\rangle$. When only FDs are present we can apply Theorem 7, and get $\textsc{Enum}_{\Delta^{\mathrm{FD}}_{Q^+}}\langle Q^+\rangle \leq_e \textsc{Enum}_{\Delta^{\mathrm{FD}}}\langle Q\rangle$. Combining the two we get the following lemma.

▶ **Lemma 23.** *Let Q be a CQ over a schema $\mathcal{S} = (\mathcal{R}, \Delta)$, where Δ is a set of CDs, and let Q^+ be the corresponding CD-extension. Then $\textsc{Enum}_{\Delta^{FD}_{Q^+}}\langle Q^+\rangle \leq_e \textsc{Enum}_\Delta\langle Q\rangle$.*

Lemma 23 implies that all negative results presented in this paper hold for CDs. 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 $\textsc{Enum}\langle R_1\rangle \leq_{e'} \textsc{Enum}\langle R_2\rangle$, instead of a bijection between the sets of outputs, one output of $\textsc{Enum}\langle R_1\rangle$ corresponds to at most a constant number of outputs of $\textsc{Enum}\langle R_2\rangle$.

▶ **Lemma 24.** *Let Q be a CQ over a schema $\mathcal{S} = (\mathcal{R}, \Delta)$, where Δ is a set of CDs, and let Q^+ be the corresponding CD-extension. Then* $\textsc{Enum}_\Delta\langle Q\rangle \leq_{e'} \textsc{Enum}_{\Delta_{Q^+}}\langle Q^+\rangle$.

Proof Sketch. 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 $\mathcal{S} = (\mathcal{R}, \Delta)$ is also an instance of $\mathcal{S}^1 = (\mathcal{R}, \Delta^1)$, where $\Delta^1 = \{(R_i\colon A \to b, c) \mid (R_i\colon A \to B, c) \in \Delta, b \in B\}$. Therefore, we can conclude that $\textsc{Enum}_\Delta\langle Q\rangle \leq_e \textsc{Enum}_{\Delta^1}\langle Q\rangle$.

We now show that $\textsc{Enum}_{\Delta^1}\langle Q\rangle \leq_{e'} \textsc{Enum}_{\Delta^+}\langle Q^+\rangle$. The proof remains the same as in Theorem 7, 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 $\textsc{Enum}_{\Delta^+}\langle Q^+\rangle$ may correspond to several solutions of $\textsc{Enum}_{\Delta^1}\langle Q\rangle$, 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. ◀

$\mathsf{DelayC_{lin}}$ is closed under this type of reduction. To avoid printing duplicates, we store the printed results. This requires a polynomial amount of memory, where the power of the polynomial is $|\operatorname{free}(Q)|$. Defining the classes of *CD-acyclic* and *CD-free-connex* queries similarly to the case with FDs, we can use Lemma 23 and Lemma 24 with Theorem 10 to generalize the dichotomy presented in Section 4 to accommodate CDs.

▶ **Theorem 25.** *Let Q be a CD-acyclic CQ with no self-joins over a schema $\mathcal{S} = (\mathcal{R}, \Delta)$, where Δ is a set of CDs.*

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

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 23 with Theorem 16, we have that such queries cannot be evaluated in linear time, assuming that the $\textsc{Tetra}(k)$ problem cannot be solved in linear time for any k.

7 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 $\mathsf{DelayC_{lin}}$ if its extension is free-connex, it is not in $\mathsf{DelayC_{lin}}$ 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. In addition to our results on constant delay enumeration of CQs with FDs, the tools provided here have immediate implications in other settings, such as for CQs with disequalities, schemas with CDs, and other enumeration classes such as $\mathsf{DelayLin}$.

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 22, remains open. This result, as well as the original one given by Brault-Baron [6] assumes the hardness of the $\textsc{Tetra}$(k) problem for every k. It will be interesting to see whether we can get the same result based on a weaker assumption. Another possible direction involves CDs. To show that enumerating CD-free-connex CQs can be done in $\mathsf{DelayC}_{\mathsf{lin}}$, we require polynomial space to store all printed results. 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 [6] with negation, to accommodate FDs.

References

1 Amir Abboud and Virginia Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*, pages 434–443. IEEE, 2014.

2 Myrto Arapinis, Diego Figueira, and Marco Gaboardi. Sensitivity of counting queries. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, Rome, Italy*, pages 120:1–120:13, 2016.

3 Guillaume Bagan, Arnaud Durand, and Etienne Grandjean. On acyclic conjunctive queries and constant delay enumeration. In *International Workshop on Computer Science Logic*, pages 208–222. Springer, 2007.

4 Catriel Beeri, Ronald Fagin, David Maier, and Mihalis Yannakakis. On the desirability of acyclic database schemes. *J. ACM*, 30(3):479–513, 1983. doi:10.1145/2402.322389.

5 Christoph Berkholz, Jens Keppeler, and Nicole Schweikardt. Answering conjunctive queries under updates. In *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2017, Chicago, IL, USA, May 14-19, 2017*, pages 303–318, 2017.

6 Johann Brault-Baron. *De la pertinence de l'énumération: complexité en logiques propositionnelle et du premier ordre.* PhD thesis, Université de Caen, 2013.

7 Yang Cao, Wenfei Fan, Tianyu Wo, and Wenyuan Yu. Bounded conjunctive queries. *PVLDB*, 7(12):1231–1242, 2014.

8 Nofar Carmeli and Markus Kröll. Enumeration complexity of conjunctive queries with functional dependencies. *CoRR*, abs/1712.07880, 2017. arXiv:1712.07880.

9 Nadia Creignou, Markus Kröll, Reinhard Pichler, Sebastian Skritek, and Heribert Vollmer. On the complexity of hard enumeration problems. In *Language and Automata Theory and Applications - 11th International Conference, LATA 2017, Umeå, Sweden*, pages 183–195, 2017.

10 Arnaud Durand, Nicole Schweikardt, and Luc Segoufin. 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, pages 121–131, 2014.

11 François Le Gall. Powers of tensors and fast matrix multiplication. In Katsusuke Nabeshima, Kosaku Nagasaka, Franz Winkler, and Ágnes Szántó, editors, *International Symposium on Symbolic and Algebraic Computation, ISSAC '14, Kobe, Japan, July 23-25, 2014*, pages 296–303. ACM, 2014. doi:10.1145/2608628.2608664.

12 Etienne Grandjean. Sorting, linear time and the satisfiability problem. *Ann. Math. Artif. Intell.*, 16:183–236, 1996. doi:10.1007/BF02127798.

13 Benny Kimelfeld. A dichotomy in the complexity of deletion propagation with functional dependencies. In Michael Benedikt, Markus Krötzsch, and Maurizio Lenzerini, editors, *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2012, Scottsdale, AZ, USA, May 20-24, 2012*, pages 191–202. ACM, 2012. doi:10.1145/2213556.2213584.

14 Christos H. Papadimitriou and Mihalis Yannakakis. On the complexity of database queries. *J. Comput. Syst. Sci.*, 58(3):407–427, 1999.

15 Luc Segoufin and Alexandre Vigny. Constant delay enumeration for FO queries over databases with local bounded expansion. In *20th International Conference on Database Theory, ICDT 2017, Venice, Italy*, pages 20:1–20:16, 2017.

16 Yann Strozecki. *Enumeration complexity and matroid decomposition.* PhD thesis, Université Paris Diderot – Paris 7, 2010. Available at `http://www.prism.uvsq.fr/~ystr/these_strozecki`.

17 Virginia Vassilevska Williams and Ryan Williams. Subcubic equivalences between path, matrix and triangle problems. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, Las Vegas, Nevada, USA*, pages 645–654, 2010.