On the Enumeration Complexity of Unions of Conjunctive Queries

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 enumeration complexity of Unions of Conjunctive Queries (UCQs). We aim to identify the UCQs that are tractable in the sense that the answer tuples can be enumerated with a linear preprocessing phase and a constant delay between every successive tuples. It has been established that, in the absence of self joins and under conventional complexity assumptions, the CQs that admit such an evaluation are precisely the free-connex ones. A union of tractable CQs is always tractable. We generalize the notion of free-connexity from CQs to UCQs, thus showing that some unions containing intractable CQs are, in fact, tractable. Interestingly, some unions consisting of only intractable CQs are tractable too. The question of a finding a full characterization of the tractability of UCQs remains open. Nevertheless, we prove that for several classes of queries, free-connexity fully captures the tractable UCQs.

CCS CONCEPTS

• **Theory of computation → Database theory**; *Complexity classes*; *Database query languages (principles)*; *Database query processing and optimization (theory)*.

KEYWORDS

unions of conjunctive queries, enumeration, complexity, constant delay

ACM Reference Format:
Nofar Carmeli and Markus Kröll. 2019. On the Enumeration Complexity of Unions of Conjunctive Queries. In *38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS'19), June 30-July 5, 2019, Amsterdam, Netherlands.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3294052.3319700

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
PODS'19, June 30-July 5, 2019, Amsterdam, Netherlands
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6227-6/19/06…$15.00
https://doi.org/10.1145/3294052.3319700

1 INTRODUCTION

Evaluating a query Q over a database instance D is a fundamental and well-studied problem in database management systems. Most of the complexity results dealing with query evaluation so far aim to solve a decision or counting variant of it. Starting with Durand and Grandjean in 2007 [7], there has been a renewed interest in examining the enumeration problem of all answers to a query, focusing on fine-grained complexity [6, 8, 15, 16]. When evaluating a non-Boolean query over a database, the number of results may be larger than the size of the database itself. Enumeration complexity offers specific measures for the hardness of such problems. In terms of data complexity, the best time guarantee we can hope for is to output all answers with a constant delay between consecutive answers. In the case of query evaluation, this enumeration phase comes after a linear preprocessing phase required to read the database and decide the existence of a first answer. The enumeration class achieving these time bounds is denoted by $\mathsf{DelayC_{lin}}$. Hereafter, we refer to queries in $\mathsf{DelayC_{lin}}$ as tractable, and queries outside of this class as intractable.

Results by Bagan et al. [2] and Brault-Baron [5] form a dichotomy that fully classifies which self-join free Conjunctive Queries (CQs) are in the class $\mathsf{DelayC_{lin}}$ based on the structure of the query: the class of *free-connex* queries is exactly the class that admits tractable enumeration. In the years following this dichotomy, much work has been conducted to achieve similar results for other classes of queries [17]. Unions of CQs (UCQs) are a natural extension of CQs, as they describe the union of the answers to several CQs. UCQs form an important class of queries, as it captures the positive fragment of relational algebra. Previous work which implies results on the enumeration complexity of UCQs imposes strong restrictions on the underlying database [18]. We aim to understand the enumeration complexity of UCQs without such restrictions and based solely on their structure.

Using known methods [19], it can be shown that a union of tractable problems is again tractable. However, what happens if some CQs of a union are tractable while others are not? Intuitively, one might be tempted to expect a union of enumeration problems to be harder than a single problem within the union, making such a UCQ intractable as well. As we will show, this is not necessarily the case.

Example 1. Let $Q = Q_1 \cup Q_2$ with

$$Q_1(x, y) \leftarrow R_1(x, y), R_2(y, z), R_3(z, x) \text{ and}$$
$$Q_2(x, y) \leftarrow R_1(x, y), R_2(y, z).$$

Even though Q_1 is hard while Q_2 is easy, a closer look shows that Q_2 contains Q_1. This means that Q_1 is redundant, and the entire union is equivalent to the easy Q_2.

To avoid cases like these, where the UCQ can be translated to a simpler one, it makes sense to consider non-redundant unions. It was claimed that in all cases of a non-redundant union containing an intractable CQ, the UCQ is intractable too [4]. The following is a counter example which refutes this claim.

Example 2. Let $Q = Q_1 \cup Q_2$ with

$$Q_1(x, y, w) \leftarrow R_1(x, z), R_2(z, y), R_3(y, w) \text{ and}$$
$$Q_2(x, y, w) \leftarrow R_1(x, y), R_2(y, w).$$

According to the dichotomy of Bagan et al. [2], the enumeration problem for Q_2 is in $\mathsf{DelayC_{lin}}$, while Q_1 is intractable. Yet, it turns out that Q is in fact in $\mathsf{DelayC_{lin}}$. The reason is that, since Q_1 and Q_2 are evaluated over the same database, we can use $Q_2(I)$ to find $Q_1(I)$. We can compute $Q_2(I)$ efficiently, and try to extend every such solution to solutions of Q_1 with a constant delay: for every new combination a, c of an output $(a, b, c) \in Q_2(I)$, we find all d values with $(b, d) \in R_3^I$ and then output the solution $(a, c, d) \in Q_1(I)$. Intuitively, the source of intractability for Q_1 is the join of R_1 with R_2 as we need to avoid duplicates that originate in different z values. The union is tractable since Q_2 returns exactly this join. □

As the example illustrates, to compute the answers to a UCQ in an efficient way, it is not enough to view it as a union of isolated instances of CQ enumeration. In fact, this task requires an understanding of the interaction between several queries. Example 2 shows that the presence of an easy query within the union may give us enough time to compute auxiliary data structures, which we can then add to the hard queries in order to enumerate their answers as well. In Example 2, we can assume we have a ternary relation holding the result of Q_2. Then, adding the auxiliary atom $R_{Q_2}(x, z, y)$ to Q_1 results in a tractable structure. We generalize this observation and introduce the concept of *union-extended* queries. We then use union extensions as a central tool for evaluating the enumeration complexity of UCQs, as the structure of such queries has implications on the tractability of the UCQ.

Interestingly, this approach can be taken a step further: We show that the concept of extending the union by auxiliary atoms can even be used to efficiently enumerate the answers of UCQs that *only* contain hard queries. By lifting the concept of free-connex queries from CQs to UCQs via union-extended queries, we show that free-connex UCQs are always tractable. This gives us a sufficient global condition for membership in $\mathsf{DelayC_{lin}}$ beyond any classification of individual CQs.

Finding a full characterization of the tractability of UCQs with respect to $\mathsf{DelayC_{lin}}$ remains an open problem. Nevertheless, we prove that for several classes of queries, free-connexity fully captures the tractable UCQs. A non-free-connex union of two CQs is intractable in the following cases: both CQs are intractable, or they both represent the same CQ up to a different projection. The hardness results presented here use problems with well-established assumptions on the lower bounds, such as Boolean matrix-multiplication [13] or finding a clique or a hyperclique in a graph [14].

Why is establishing lower bounds on UCQ evaluation, even when it contains only two CQs, a fundamentally more challenging problem than the one for CQs? In the case of CQs, hardness results are often shown by reducing a computationally hard problem to the task of answering a query. The reduction encodes the hard problem to the relations of a self-join free CQ, such that the answers of the CQ correspond to an answer of this problem [2, 3, 5, 6]. However, using such an encoding for CQs within a union does not always work. Similarly to the case of CQs with self-joins, relational symbols that appear multiple times within a query can interfere with the reduction. Indeed, when encoding a hard problem to an intractable CQ within a union, a different CQ in the union evaluates over the same relations, and may also produce answers. A large number of such supplementary answers, with constant delay per answer, accumulates to a long delay until we obtain the answers that correspond to the computationally hard problem. If this delay is larger than the lower bound we assume for the hard problem, we cannot conclude that the UCQ is intractable.

The lower bounds presented in this paper are obtained either by identifying classes of UCQs for which we can use similar reductions to the ones used for CQs, or by introducing alternative reductions. As some cases remain unclassified, we spend the last section of this paper inspecting such UCQs, and describing the challenges that will need to be resolved in order to achieve a full classification.

Our main contributions are as follows:

- We show that some non-redundant UCQs containing intractable CQs are tractable, and even that unions containing only intractable CQs may be tractable.
- We extend the notion of free-connexity to UCQs, and show that free-connex UCQs can be evaluated with linear time preprocessing and constant delay.

- We establish lower bounds on UCQs w.r.t. $\mathsf{DelayC_{lin}}$, and prove that free-connexity captures exactly the tractable cases in some classes of UCQs.
- We provide a discussion accompanied by examples, describing the challenges that need to be resolved to achieve a full characterization of the tractable UCQs.

This work is organized as follows: In Section 2 we provide definitions and state results that we use. Section 3 formalizes how CQs within a union can make each other easier, defines free-connex UCQs, and proves that free-connex UCQs are in $\mathsf{DelayC_{lin}}$. In Section 4 we prove conditional lower bounds and conclude a dichotomy for some classes of UCQs. Section 5 discusses the future steps required for a full classification, and demonstrates examples of queries of unknown complexity. Concluding remarks are given in Section 6.

2 PRELIMINARIES

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

Unions of Conjunctive Queries. A *schema* $\mathcal{S}$ is a set of *relational symbols* $\{R_1, \dots, R_n\}$. We denote the *arity* of a relational symbol R_i as $\text{arity}(R_i)$. 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 $R_i \in \mathcal{R}$.

Let *var* be a set of variables disjoint from *dom*. A *Conjunctive Query* (CQ) over schema $\mathcal{S}$ is an expression of the form $Q(\vec{p}) \leftarrow R_1(\vec{v}_1), \dots, R_m(\vec{v}_m)$, where $R_1, \dots, R_m$ are relational symbols of $\mathcal{S}$, the tuples $\vec{p}, \vec{v}_1, \dots, \vec{v}_m$ hold variables, and every variable in $\vec{p}$ appears in at least one of $\vec{v}_1, \dots, \vec{v}_m$. We often denote this query as Q. Define the variables of Q as $var(Q) = \bigcup_{i=1}^m \vec{v}_i$, and define the *free variables* of Q as $\text{free}(Q) = \vec{p}$. We call $Q(\vec{p})$ the head of Q, and the atomic formulas $R_i(\vec{v}_i)$ are called *atoms*. We further use $\text{atoms}(Q)$ to denote the set of atoms of Q. A CQ is said to be *self-join free* if no relational symbol appears in more than one atom.

The *evaluation* $Q(I)$ of a CQ Q over a database I is the set of all *mappings* $\mu|_{\text{free}(Q)}$ such that μ is a homomorphism from $R_1(\vec{v}_1), \dots, R_m(\vec{v}_m)$ into I, and $\mu|_{\text{free}(Q)}$ is the restriction (or projection) of μ to the variables $\text{free}(Q)$. A CQ Q_1 is *contained* in a CQ Q_2, denoted $Q_1 \subseteq Q_2$, if for every instance I, $Q_1(I) \subseteq Q_2(I)$. A *homomorphism* from Q_2 to Q_1 is a mapping $h : var(Q_2) \to var(Q_1)$ such that: (1) for every atom $R(\vec{v})$ of Q_2, $R(h(\vec{v}))$ is an atom in Q_1; (2) $h(\text{free}(Q_2)) = \text{free}(Q_1)$.

A *Union of Conjunctive Queries (UCQ)* Q is a set of CQs, denoted $Q = \bigcup_{i=1}^{\ell} Q_i$, where $\text{free}(Q_{i_1}) = \text{free}(Q_{i_2})$ for all $1 \leq i_1, i_2 \leq \ell$. Semantically, $Q(I) = \bigcup_{i=1}^{\ell} Q_i(I)$. Given a UCQ Q and a database instance I, we denote by $\textsc{Decide}\langle Q \rangle$ the problem of deciding whether $Q(I) \neq \emptyset$.

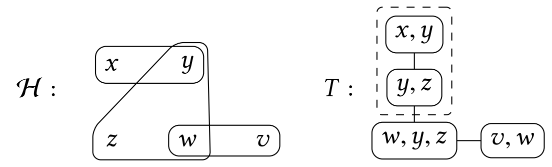

Figure 1: T is an ext-$\{x, y, z\}$-connex tree for hypergraph $\mathcal{H}$.

Hypergraphs. A *hypergraph* $\mathcal{H} = (V, E)$ is 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 *acyclic* if there exists a join tree for $\mathcal{H}$.

Two vertices in a hypergraph are *neighbors* if they appear in the same edge. A *clique* of a hypergraph is a set of vertices, which are pairwise neighbors in $\mathcal{H}$. If every edge in $\mathcal{H}$ has k many vertices, we call $\mathcal{H}$ k-*u*niform. An l-*hyperclique* in a k-uniform hypergraph $\mathcal{H}$ is a set V' of $l > k$ vertices, such that every subset of V' of size k forms a hyperedge.

A hypergraph $\mathcal{H}'$ is an *inclusive extension* of $\mathcal{H}$ if every edge of $\mathcal{H}$ appears in $\mathcal{H}'$, and every edge of $\mathcal{H}'$ is a subset of some edge in $\mathcal{H}$. A tree T is an *ext-S-connex tree* for a hypergraph $\mathcal{H}$ if: (1) T is a join-tree of an inclusive extension of $\mathcal{H}$, and (2) there is a subtree T' of T that contains exactly the variables S [2] (see Figure 1).

Classes of CQs. We associate a hypergraph $\mathcal{H}(Q) = (V, E)$ to a CQ Q where the vertices are the variables of Q, and every hyperedge is a set of variables occurring in a single atom of Q. That is, $E = \{\{v_1, \dots, v_n\} \mid R_i(v_1, \dots, v_n) \in \text{atoms}(Q)\}$. With a slight abuse of notation, we identify atoms of Q with edges of $\mathcal{H}(Q)$. A CQ Q is said to be *acyclic* if $\mathcal{H}(Q)$ is acyclic

A CQ Q is S-*connex* if $\mathcal{H}(Q)$ has an ext-S-connex tree, and it is *free-connex* if it has an ext-free(Q)-connex tree [2]. Equivalently, Q is free-connex if both Q and $(V, E \cup \{\text{free}(Q)\})$ are acyclic [5]. A *free-path* in a CQ Q is a sequence of variables $(x, z_1, \dots, z_k, y)$ with $k \geq 1$, such that: (1) $\{x, y\} \subseteq \text{free}(Q)$ (2) $\{z_1, \dots, z_k\} \subseteq V \setminus \text{free}(Q)$ (3) It is a *chordless path* in $\mathcal{H}(Q)$: that is, every two succeeding variables are neighbors in $\mathcal{H}(Q)$, but no two non-succeeding variables are neighbors. An acyclic CQ has a free-path iff it is not free-connex [2].

Computational Model. The size of the input to most of our problems is measured only by the size of the database instance I. We denote by $||o||$ the size of an object o (i.e., the number of integers required to store it), whereas $|o|$ is its cardinality. 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 word over integers bound by $\max\{|dom|, max_{R \in \mathcal{R}} |R^I|\}$ [9].

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 $O(\log(n))$. 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 within constant time.

We use a variant of the RAM model named DRAM [10], 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.

Enumeration Complexity. Given a finite alphabet Σ and binary relation $R \subseteq \Sigma^* \times \Sigma^*$, the *enumeration problem* $\textsc{Enum}\langle R\rangle$ is: given an instance $x \in \Sigma^*$, output all $y \in \Sigma^*$ such that $(x, y) \in R$. Such y values are often called *solutions* or *answers* to $\textsc{Enum}\langle R\rangle$. An *enumeration algorithm* $\mathcal{A}$ for $\textsc{Enum}\langle R\rangle$ is a RAM that solves $\textsc{Enum}\langle R\rangle$ without repetitions. We say that $\mathcal{A}$ enumerates $\textsc{Enum}\langle R\rangle$ with *delay* $d(|x|)$ if the time before the first output, the time between any two consecutive outputs, and the time between the last output and termination are each bound by $d(|x|)$. Sometimes we wish to relax the requirements of the delay before the first answer, and specify a *preprocessing* time $p(|x|)$. In this case, the time before the first output is only required to be bound by $p(|x|)$. The enumeration class $\mathsf{DelayC_{lin}}$ is defined as the class of all enumeration problems $\textsc{Enum}\langle R\rangle$ which have an enumeration algorithm $\mathcal{A}$ with preprocessing $p(|x|) \in O(|x|)$ and delay $d(|x|) \in O(1)$. Note that we do not impose a restriction on the memory used. In particular, such an algorithm may use additional constant memory for writing between two consecutive answers.

Let $\textsc{Enum}\langle R_1\rangle$ and $\textsc{Enum}\langle R_2\rangle$ be enumeration problems. There is an *exact reduction* from $\textsc{Enum}\langle R_1\rangle$ to $\textsc{Enum}\langle R_2\rangle$, denoted as $\textsc{Enum}\langle R_1\rangle \leq_e \textsc{Enum}\langle R_2\rangle$, if there exist mappings σ and τ such that: (1) for every $x \in \Sigma^*$, $\sigma(x)$ is computable in $O(|x|)$ time; (2) for every y such that $(\sigma(x), y) \in R_2$, $\tau(y)$ is computable in $O(1)$ time; and (3) in multiset notation, $\{\tau(y) \mid (\sigma(x), y) \in R_2\} = \{y' \mid (x, y') \in R_1\}$. Intuitively, σ maps instances of $\textsc{Enum}\langle R_1\rangle$ to instances of $\textsc{Enum}\langle R_2\rangle$, and τ maps solutions of $\textsc{Enum}\langle R_2\rangle$ to solutions of $\textsc{Enum}\langle R_1\rangle$. If $\textsc{Enum}\langle R_1\rangle \leq_e \textsc{Enum}\langle R_2\rangle$ and $\textsc{Enum}\langle R_2\rangle \in \mathsf{DelayC_{lin}}$, then $\textsc{Enum}\langle R_1\rangle \in \mathsf{DelayC_{lin}}$ as well [2].

Computational Hypotheses. In the following, we will use the following well-established hypotheses for lower bounds on certain computational problems:

mat-mul: two Boolean $n \times n$ matrices cannot be multiplied in time $O(n^2)$. This problem is equivalent to the evaluation of the query $\Pi(x, y) \leftarrow A(x, z), B(z, y)$ over the schema $\{A, B\}$ where $A, B \subseteq \{1, \ldots, n\}^2$. It is strongly conjectured that this problem cannot be solved in $O(n^2)$ time, and the best algorithms today require $O(n^\omega)$ time for some $2.37 < \omega < 2.38$ [1, 13].

hyperclique: for all $k \geq 3$, finding a k-hyperclique in a $(k-1)$-uniform graph is not possible in time $O(n^{k-1})$. This is a special case of the (ℓ, k)– Hyperclique Hypothesis [14], which 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. The hyperclique hypothesis is sometimes called Tetra$\langle k\rangle$ [5].

4-clique: it is not possible to determine the existence of a 4-clique in a graph with n nodes in time $O(n^3)$.
This is a special case of the k-Clique Hypothesis [14], which states that detecting a clique in a graph with n nodes requires $n^{\frac{\omega k}{3} - o(1)}$ time, where $\omega < 2.373$ is the matrix multiplication exponent.

Enumerating Answers to UCQs. Given a UCQ Q over some schema $\mathcal{S}$, we denote by $\textsc{Enum}\langle Q\rangle$ 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. In the case of CQs, Bagan et al. [2] showed that a self-join free acyclic CQ is in $\mathsf{DelayC_{lin}}$ iff it is free-connex. In addition, Brault-Baron [5] showed that self-join free cyclic queries are not in $\mathsf{DelayC_{lin}}$. In fact, the existence of a single answer to a cyclic CQ cannot be determined in linear time.

Theorem 3 ([2, 5])**.** *Let Q be a self-join free CQ.*

(1) If Q is free-connex, then $\textsc{Enum}\langle Q\rangle \in \mathsf{DelayC_{lin}}$.
(2) If Q is acyclic and not free-connex, then $\textsc{Enum}\langle Q\rangle$ *is not in* $\mathsf{DelayC_{lin}}$, *assuming mat-mul.*
(3) If Q is cyclic, then $\textsc{Enum}\langle Q\rangle \notin \mathsf{DelayC_{lin}}$, *as* $\textsc{Decide}\langle Q\rangle$ *cannot be solved in linear time, assuming hyperclique.*

The positive case of this dichotomy can be shown using the *Constant Delay Yannakakis* (CDY) algorithm [11]. It uses an ext-free(Q)-connex tree T for Q. First, it performs the classical Yannakakis preprocessing [20] over T to obtain a relation for each node in T, where all tuples can be used for some answer in $Q(I)$. Then, it considers only the subtree of T containing free(Q), and joins the relations corresponding to this subtree with constant delay.

3 UPPER BOUNDS VIA UNION EXTENSIONS

In this section we generalize the notion of free-connexity to UCQs and show that such queries are in $\mathsf{DelayC_{lin}}$. We do so by introducing the concepts of *union extensions* and variable sets that a single CQ can *provide* to another CQ in the union in order to help evaluation. First, note that using known techniques [19, Proposition 2.38] a union of tractable CQs is also tractable.

Theorem 4. *Let Q be a UCQ. If all CQs in Q are free-connex, then* $\textsc{Enum}\langle Q\rangle \in \mathsf{DelayC_{lin}}$.

Proof. The following is an algorithm to evaluate a union of two CQs. In case of a union $Q = \bigcup_{i=1}^{\ell} Q_i$ of more CQs, we can use this recursively by treating the second query as $Q_2 \cup \ldots \cup Q_\ell$.

Algorithm 1 Answering a union of two tractable CQs

```
while $a \leftarrow Q_1(I).next()$ do
    if $a \notin Q_2(I)$ then
        print $a$
    else
        print $Q_2(I).next()$
while $a \leftarrow Q_2(I).next()$ do
    print $a$
```

By the end of the run, the algorithm prints $Q_1(I) \setminus Q_2(I)$ over all iterations of line 3, and it prints $Q_2(I)$ in lines 5 and 7. Line 5 is called $Q_1(I) \cap Q_2(I)$ times, so the command $Q_2(I).next()$ always succeeds there. Since free-connex CQs can be enumerated in constant delay and tested in constant time after a linear time preprocessing phase, this algorithm runs within the required time bounds. □

The technique presented in the proof of 4 has the advantage that it does not require more than constant memory available for writing in the enumeration phase. Alternatively, this theorem is a consequence of the following lemma, which gives us a general approach to compile several enumeration algorithms into one. This lemma is useful to show upper bounds for UCQs even in cases not covered by Theorem 4.

Lemma 5. *Let $R \subseteq \Sigma^* \times \Sigma^*$, and let $\mathcal{A}$ be an algorithm that outputs the solutions to* $\textsc{Enum}\langle R\rangle$ *such that:*

- *the delay of $\mathcal{A}$ is bound by $p(x)$ at most n times and bound by $d(x)$ otherwise;*
- *every result is produced at most m times.*

Then, there exists an enumeration algorithm $\mathcal{A}'$ for $\textsc{Enum}\langle R\rangle$*, with $np(x) + md(x)$ preprocessing time and $md(x)$ delay.*

Proof. $\mathcal{A}'$ simulates $\mathcal{A}$ and maintains a lookup table and a queue that are initialized as empty. When $\mathcal{A}$ returns a result, $\mathcal{A}'$ checks the lookup table to determine whether it was found before. If it was not, the result is added to both the lookup table and the queue. $\mathcal{A}'$ first performs $np(x)$ computation steps, and then after every $md(x)$ computation steps, it outputs a result from the queue. $\mathcal{A}'$ returns its ith result after $np(x) + imd(x)$ computation steps. At this time, $\mathcal{A}$ produced at least mi results, which form at least i unique results, so the queue is never empty when accessed. When it is done simulating $\mathcal{A}'$, $\mathcal{A}$ outputs all remaining results in the queue. $\mathcal{A}'$ outputs all results of $\mathcal{A}$ with no duplicates since every result enters the queue exactly once. □

A direct consequence of Lemma 5 is that to show that a problem is in $\mathsf{DelayC_{lin}}$, it suffices to find an algorithm for this problem where the delay is usually constant, but it may be linear a constant number of times, and the number of times every result is produced is bound by a constant.

As Example 2 shows, Theorem 4 does not cover all tractable UCQs. We now address the other cases and start with some definitions. We define *body-homomorphisms* between CQs to have the standard meaning of homomorphism, but without the restriction on the heads of the queries.

Definition 6. Let Q_1, Q_2 be CQs.

- A *body-homomorphism* from Q_2 to Q_1 is a mapping $h : var(Q_2) \to var(Q_1)$ such that for every atom $R(\vec{v})$ of Q_2, $R(h(\vec{v})) \in Q_1$.
- If Q_1, Q_2 are self-join free and there exists a body-homomorphism h from Q_2 to Q_1 and vice versa, we say that Q_2 and Q_1 are *body-isomorphic*, and h is called a *body-isomorphism.*

We now formalize the way that one CQ can help another CQ in the union during evaluation by providing variables.

Definition 7. Let Q_1, Q_2 be CQs. We say that Q_2 *provides* a set of variables $V_1 \subseteq var(Q_1)$ to Q_1 if:

(1) There is a body-homomorphism h from Q_2 to Q_1.
(2) There is $V_2 \subseteq \text{free}(Q_2)$ such that $h(V_2) = V_1$.
(3) There is $V_2 \subseteq S \subseteq \text{free}(Q_2)$ such that Q_2 is S-connex.

Intuitively, the first condition requires that Q_2 provides complete information regarding the possible assignments to some variables of Q_1, the second condition ensures that these include the provided variables, and the third condition guarantees that these assignments can easily be computed as part of Q_2. The following lemma shows why provided variables play an important role for UCQ enumeration: If Q_2 provides a set of variables to Q_1, then we can produce an auxiliary relation for Q_1 containing all possible value combinations of these variables. This can be done efficiently while producing some answers to Q_2.

Lemma 8. *Let Q_1, Q_2 be CQs such that Q_2 provides V_1 to Q_1. Given an instance I, one can compute with linear time preprocessing and constant delay a set of mappings $M \subseteq Q_2(I)$, which can be translated to $Q_1(I)|_{V_1}$ in time $O(|M|)$.*

Proof. Let h, V_2 and S be a body-homomorphism and two sets of variables meeting the conditions of Definition 7. Take an ext-S-connex tree T for Q_2, and perform the CDY algorithm on Q_2 while treating S as the free-variables. This results in a set N of mappings from the variables of S to the domain such that $N = Q_2(I)|_S$.

For every mapping $\mu \in N$, extend it once to obtain a mapping from all variables of Q_2 as follows. Go over all vertices of T starting from the connected part containing S and treating a neighbor of an already treated vertex at every step. Consider a step where in its beginning μ is a homomorphism from a set S_1, and we are treating an atom $R_i(\vec{v}_i, \vec{u}_i)$ where $\vec{v}_i \subseteq S_1$ and $\vec{u}_i \cap S_1 = \emptyset$. We take some tuple in R_i of the form $(\mu(\vec{v}_i), t_i)$ and extend μ to also map $\mu(\vec{u}_i) = t_i$. Such a tuple exists since the CDY algorithm has a preprocessing step that removes dangling tuples. This extension takes constant time, and in its end we have that $\mu|_{free(Q)} \in Q_2(I)$. These extensions form $M \subseteq Q_2(I)$. When computing M, the delay for the first element may be linear due to the preprocessing phase of the CDY algorithm, but the delay after that is constant.

We now describe how M can be translated to $Q_1(I)|_{V_1}$. As $M|_{V_2} = Q_2(I)|_{V_2}$, we need to use the body-homomorphism in the opposite direction. For every variable $v_1 \in V_1$, define $h^{-1}(v_1) = \{v_2 \in V_2 \mid h(v_2) = v_1\}$. Given a mapping μ in $Q_2(I)$, if $\mu(v_2)$ is the same for all $v_2 \in h^{-1}(v_1)$, denote it by $\mu(h^{-1}(v_1))$. Otherwise, $\mu(h^{-1}(v_1))$ is undefined, and in the following μ is skipped. As h is a body-homomorphism, we have that $M|_{V_2} \circ h^{-1} = Q(I)|_{V_1}$. Given $\mu \in M$, we can compute $\mu|_{V_2} \circ h^{-1}$ (or determine it is undefined) in constant time. Doing this for every $\mu \in M$ computes $Q_1(I)|_{V_1}$ in time $O(|M|)$. □

Note that without the first condition of Definition 7, the lemma above does not hold in general. Here is an example.

Example 9. Consider $Q = Q_1 \cup Q_2$, which is a slight modification of the UCQ in from Example 2, with

$$Q_1(x, y, w) \leftarrow R_1(x, z), R_2(z, y), R_3(y, w) \text{ and}$$
$$Q_2(x, y, w) \leftarrow R_1(x, y), R_2(y, w), R_4(y).$$

Since R_4 is not a relational symbol in Q_1, there is no body-homomorphism from Q_2 to Q_1. If, $R_4^I = dom$, then we can take the same approach as in Example 2, as the answers of Q_2 form $Q_1(I)|_{\{x,z,y\}}$. However, if R_4^I is smaller, this extra atom may filter the answers to Q_2, and we do not obtain all of $Q_1(I)|_{\{x,z,y\}}$ in general. □

During evaluation, a set of variables provided to a CQ can form an auxiliary relation accessible by an auxiliary atom. The CQ with its auxiliary atoms is called a *union extension.*

Definition 10. Let $Q = Q_1 \cup \ldots \cup Q_n$ be a UCQ.

- A *union extension* Q_1^+ of $Q_1(\vec{v}) \leftarrow R_1(\vec{v}_1), \ldots, R_s(\vec{v}_s)$ is

$$Q_1^+(\vec{v}) \leftarrow R_1(\vec{v}_1), \ldots, R_s(\vec{v}_s), P_1(\vec{u}_1), \ldots, P_k(\vec{u}_k)$$

  where $k \geq 0$, each $\vec{u}_i$ is provided by some $Q_j \in Q$, and $P_1, \ldots, P_k$ are fresh relational symbols. By way of recursion, the variables $\vec{u}_i$ may alternatively be provided by a union extension of some $Q_j \in Q$.
- *Virtual atoms* are atoms appearing in Q_1^+ but not in Q_1.

Union extension can transform an intractable query to a free-connex one.

Definition 11. Let $Q = Q_1 \cup \ldots \cup Q_n$ be a UCQ.

- Q_1 is said to be *union-free-connex* with respect to Q if it has a free-connex union extension.
- Q is *free-connex* if all CQs in Q are union-free-connex.

Note that the term free-connex for UCQs is a generalization of that for CQs: If a UCQ Q contains only one CQ, then Q is free-connex iff the CQ it contains is free-connex. We next show that tractability of free-connex queries also carries over to UCQs.

Theorem 12. *Let Q be a UCQ. If Q is free-connex, then* $\textsc{Enum}\langle Q \rangle \in \mathsf{DelayC_{lin}}$.

Proof. For each query Q_1 in the union (in an order imposed by the recursive definition of union extensions), we first instantiate its free-connex union extension Q_1^+, and then evaluate the resulting free-connex CQ using the CDY algorithm: For every virtual atom containing some variables V_1, use Lemma 8 to generate a subset of $Q_2(I)$ while obtaining a relation $Q_1(I)|_{V_1}$ assigned to this atom. After instantiating all virtual relations, we have an instance I^+ for Q_1^+, and we can evaluate it as usual using the CDY algorithm. We have that $Q_1(I) = Q_1^+(I^+)$ since all virtual atoms in Q_1^+ are assigned relations that contain merely a projection of the results.

Overall, there is a constant number of times where the delay is linear: once per query and once per virtual atom. Similarly, every result is produced at most a constant number of times: once per query and once per virtual atom. According to Lemma 5 this means that $\textsc{Enum}\langle Q \rangle \in \mathsf{DelayC_{lin}}$. □

We can now revisit Example 2 and explain its tractability using the terminology and results introduced in this section. There is a body-homomorphism $h : Q_2 \to Q_1$ with $h((x, y, w)) = (x, z, y)$. The query Q_2 provides $\{x, z, y\}$ to Q_1, as $\{x, y, w\} \subseteq \text{free}(Q_2)$, and Q_2 is $\{x, y, w\}$-connex. Adding $R'(x, z, y)$ to Q_1 results in a free-connex union extension $Q_1^+(x, y, w) \leftarrow R_1(x, z), R_2(z, y), R_3(y, w), R'(x, z, y)$, as illustrated in Figure 2. Since every query in Q is union-free-connex, we have that $\textsc{Enum}\langle Q \rangle \in \mathsf{DelayC_{lin}}$ by Theorem 12.

Remark 1. Example 2 is a counter example to a past made claim [4, Theorem 4.2b]. The claim is that if a UCQ contains an intractable CQ and does not contain redundant CQs (a CQ contained in another CQ in the union), then the union is intractable. In contrast, none of the CQs in Example 2 is redundant, Q_1 is intractable, and yet the UCQ is tractable.

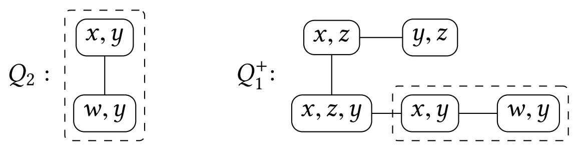

Figure 2: A $\{x, y, w\}$-connex tree for query Q_2 and a $\{x, y, w\}$-connex tree for query Q_1^+.

The intuition behind the proof of the past claim is reducing the hard CQ Q_1 to Q. This can be done by assigning each variable of Q_1 with a different and disjoint domain (e.g., by concatenating the variable names to the values in the relations corresponding to the atoms), and leaving the relations that do not appear in the atoms of Q_1 empty. It is well known that $Q_1 \subseteq Q_2$ iff there exists a homomorphism from Q_2 to Q_1. The claim is that since there is no homomorphism from another CQ in the union to Q_1, then there are no answers to the other CQs with this reduction. However, it is possible that there is a body-homomorphism from another CQ to Q_1 even if it is not a full homomorphism (the free variables do not map to each other). Therefore, in cases of a body-homomorphism, the reduction from the Q_1 to Q does not work. In such cases, the union may be tractable, as we show in Theorem 12. In Lemma 14, we use the same proof described here, but restrict it to UCQs where there is no body-homomorphism from other CQs to Q_1. ◻

The tractability result in Theorem 12 is based on the structure of the union-extended queries. This means that the intractability of any query within a UCQ can be resolved as long as another query can provide the right variables. The following example shows that this can even be the case for a UCQ only consisting of non-free-connex CQs. Moreover, the example illustrates why we need the definition of union extensions to be recursive.

Example 13. Let $Q = Q_1 \cup Q_2 \cup Q_3$ with

$$\begin{aligned}
Q_1(x, y, v, u) \leftarrow & R_1(x, z_1), R_2(z_1, z_2), R_3(z_2, z_3), \\
& R_4(z_3, y), R_5(y, v, u), \\
Q_2(x, y, v, u) \leftarrow & R_1(x, y), R_2(y, v), R_3(v, z_1), \\
& R_4(z_1, u), R_5(u, t_1, t_2), \\
Q_3(x, y, v, u) \leftarrow & R_1(x, z_1), R_2(z_1, y), R_3(y, v), \\
& R_4(v, u), R_5(u, t_1, t_2).
\end{aligned}$$

Each of three CQs is intractable on its own: Q_1 has the free-path (x, z_1, z_2, z_3, y), while Q_2 has the free-path (v, z_1, u), and Q_3 has the free-path (x, z_1, y). The CQ Q_2 provides the variables $\{x, z_1, y\}$ to Q_3, as Q_2 is $\{x, y, v\}$-connex, $\{x, y, v\}$ are free in Q_2, and there is a body-homomorphism h from Q_2 to Q_3 with $h((x, y, v)) = (x, z_1, y)$. Extending the body of Q_3 by the virtual atom $R'(x, z_1, y)$ yields the free-connex union extension Q_3^+. Similarly, we have that Q_3 provides $\{v, z_1, u\}$ to Q_2, and extending Q_2 by $R''(v, z_1, u)$ yields the free-connex union extension Q_2^+. Since Q_2^+ and Q_3^+ provide $\{x, z_1, z_2, y\}$ and respectively $\{x, z_2, z_3, y\}$ to Q_1, we obtain a free-connex union extension Q_1^+ by adding virtual atoms with the variables (x, z_1, z_2, y) and (x, z_2, z_3, y) to Q_1. Thus, Q is free-connex and $\text{Enum}\langle Q \rangle \in \text{DelayC}_{\text{lin}}$ by Theorem 12. ◻

Remark 2. The approach presented here can also be used when there are functional dependencies in the schema. By taking functional dependencies into account, we can find even more tractable cases. If there are functional dependencies, some intractable CQs have a tractable FD-extension that can be computed efficiently [6]. Given a UCQ over a schema with functional dependencies, we can first take the FD-extensions of all CQs in the union, and then take the union extensions of those and evaluate the union. ◻

4 LOWER BOUNDS

In this section, we prove lower bounds for evaluating UCQs within the time bounds of $\text{DelayC}_{\text{lin}}$. We begin with some general observations regarding cases where a single CQ is not harder than a union containing it, and then continue to handle other cases. In Section 4.1 we discuss unions containing only intractable CQs, and in Section 4.2 we discuss unions containing two body-isomorphic CQs. In both cases such UCQs may be tractable, and in case of such a union of size two, we show that our results from Section 3 capture all tractable unions.

In order to provide some intuition for the choices we make throughout this section, we first explain where the approach used for proving the hardness of single CQs fails. Consider Example 2. The original proof that shows that Q_1 is hard describes a reduction from Boolean matrix multiplication [2, Lemma 26]. Let A and B be binary representations of Boolean $n \times n$ matrices, i.e. $(a, b) \in A$ corresponds to a 1 in the first matrix at index (a, b). Define a database instance I as $R_1^I = A$, $R_2^I = B$, and $R_3^I = \{1, \ldots, n\} \times \{\bot\}$. One can show that $Q_1(I)$ corresponds to the answers of AB. If $\text{Enum}\langle Q_1 \rangle \in \text{DelayC}_{\text{lin}}$, we can solve matrix multiplication in time $O(n^2)$, in contradiction to MAT-MUL. Since Q_2 evaluates over the same relations, Q_2 also produces answers over this construction. Since the number of results for Q_2 might reach up to n^3, evaluating Q in constant delay does not necessarily compute the answers to Q_1 in $O(n^2)$ time, and does not contradict the complexity assumption.

So in general, whenever we show a lower bound to a UCQ by computing a hard problem through answering one CQ in the union, we need to ensure that the other CQs cannot have too many answers over this construction. As a first step to resolve this issue, we describe cases where there is a way to encode any arbitrary instance of Q_1 to an instance of Q, such that no other CQ in the union returns results.

Lemma 14. *Let Q be a UCQ of self-join free CQs, and let $Q_1 \in Q$ such that for all $Q_i \in Q \setminus \{Q_1\}$ there is no body homomorphism from Q_i to Q_1. Then* $\textsc{Enum}\langle Q_1\rangle \leq_e \textsc{Enum}\langle Q\rangle$.

Proof sketch. Given an instance of $\textsc{Enum}\langle Q_1\rangle$, we assign each variable of Q_1 with a different and disjoint domain by concatenating the variable names to the values in their corresponding relations. We leave the relations that do not appear in the atoms of Q_1 empty. Since there is no body-homomorphism from Q_i to Q_1, then there are now no answers to Q_i over this construction, and the answers to Q are exactly those of Q_1. □

The lemma above implies that if there is an intractable CQ in a union where no other CQ maps to it via a body-homomorphism, then the entire union is intractable. This also captures cases such as a union of CQs where one of them is hard, and the others contain a relation that does not appear in the first.

Using the same reduction, a similar statement with relaxed requirements can be made in case it is sufficient to consider the decision problem.

Lemma 15. *Let Q be a UCQ of self-join free CQs, and let $Q_1 \in Q$ such that for all $Q_i \in Q$, either there exists no body-homomorphism from Q_i to Q_1, or Q_1 and Q_i are body-isomorphic. Then $\textsc{Decide}\langle Q_1\rangle \leq \textsc{Decide}\langle Q\rangle$ via a linear-time many-one reduction.*

Proof sketch. We use the same encoding as in Lemma 14. A CQ with no body-homomorphism to Q_1 has no answers. A CQ which is body-isomorphic to Q_1 has an answer iff Q_1 has an answer. Therefore $Q(I) \neq \emptyset$ iff $Q_1(I) \neq \emptyset$. □

Theorem 3 states that deciding whether a cyclic CQ has any answers cannot be done in linear time (assuming hyperclique). Following Lemma 15, if a UCQ Q containing a cyclic Q_1 where the conditions of Lemma 15 are satisfied, the entire union cannot be decided in linear time, and thus $\textsc{Enum}\langle Q\rangle \notin \mathsf{DelayC_{lin}}$.

4.1 Unions of Intractable CQs

We now discuss unions containing only CQs classified as hard according to Theorem 3. In the following, *intractable CQs* refers to self-join-free CQs that are not free-connex. The following lemma can be used to identify a CQ on which we can apply Lemma 14 or Lemma 15.

Lemma 16. *Let Q be a UCQ. There exists a query $Q_1 \in Q$ such that for all $Q_i \in Q$ either there is no body-homomorphism from Q_i to Q_1 or Q_1 and Q_i are body-isomorphic.*

Proof sketch. Consider a longest sequence $(Q^1, \dots, Q^m)$ such that for every $2 \leq j \leq m$ there is a body-homomorphism from Q^j to Q^{j-1}, but no body-homomorphism in the opposite direction. The CQ Q^m satisfies the conditions of the lemma: For every Q_i not on the sequence, if there is a body homomorphism from Q_i to Q^m, then there is also one in the opposite direction due to the maximality of the sequence; For every Q_i on the sequence, there is a body-homomorphism from Q^m to Q_i, so either there is no body-homomorphism in the opposite direction, or the CQs are body-isomorphic. □

Using the results obtained so far, we deduce a characterization of all cases of a union of intractable CQs, except those that contain a pair of body-isomorphic acyclic CQs.

Theorem 17. *Let Q be a UCQ of intractable CQs that does not contain two body-isomorphic acyclic CQs. Then, $Q \notin \mathsf{DelayC_{lin}}$, assuming mat-mul and hyperclique.*

Proof. Let Q_1 be a CQ in Q given by Lemma 16. We treat the two possible cases of the structure of Q_1. In case Q_1 is acyclic, since we know that Q does not contain body-isomorphic acyclic CQs, then for all $Q_i \in Q \setminus \{Q_1\}$ there is no body-homomorphism from Q_i to Q_1. According to Lemma 14, $\textsc{Enum}\langle Q_1\rangle \leq_e \textsc{Enum}\langle Q\rangle$. Since Q_1 is self-join free acyclic non-free-connex, we have that $\textsc{Enum}\langle Q_1\rangle \notin \mathsf{DelayC_{lin}}$ assuming mat-mul. Therefore $\textsc{Enum}\langle Q\rangle$ is not in $\mathsf{DelayC_{lin}}$ either. In case Q_1 is cyclic, we use Lemma 15 to conclude that $\textsc{Decide}\langle Q_1\rangle \leq \textsc{Decide}\langle Q\rangle$. According to Theorem 3, since Q_1 is self-join free cyclic, $\textsc{Decide}\langle Q_1\rangle$ cannot be solved in linear time assuming hyperclique. Therefore $\textsc{Decide}\langle Q\rangle$ cannot be solved in linear time, thus $\textsc{Enum}\langle Q\rangle \notin \mathsf{DelayC_{lin}}$. □

In the next example, we demonstrate how the reductions from Lemma 15 and Theorem 3 combine in Theorem 17.

Example 18. Consider the UCQ $Q = Q_1 \cup Q_2 \cup Q_3$ with

$$Q_1(x,y) \leftarrow R_1(x,y), R_2(y,u), R_3(x,u),$$
$$Q_2(x,y) \leftarrow R_1(y,v), R_2(v,x), R_3(y,x),$$
$$Q_3(x,y) \leftarrow R_1(x,z), R_2(y,z).$$

The queries Q_1 and Q_2 are cyclic, and Q_3 is acyclic but not free-connex. This union is intractable according to Theorem 17. Note that Q_1 and Q_2 are body-isomorphic, but there is no body-homomorphism from Q_3 to Q_1. The proof of Theorem 3 states the following: If $\textsc{Enum}\langle Q_1\rangle \in \mathsf{DelayC_{lin}}$, then given an input graph G, we can use Q_1 to decide the existence of triangles in G in time $O(n^2)$, in contradiction to hyperclique. The same holds true for the $\textsc{Enum}\langle Q\rangle$. For every edge (u,v) in G with $u < v$ we add $((u,x),(v,y))$ to R_1^I, $((u,y),(v,z))$ to R_2^I and $((u,x),(v,z))$ to R_3^I. The query detects triangles: for every triangle a,b,c in G with $a < b < c$, the query Q_1 returns $((a,x),(b,y))$. The union only returns answers corresponding to triangles:

- For every answer $((d,x),(e,y))$ to Q_1, there exists some f such that d,e,f is a triangle in G with $d<e<f$.
- For every answer $((g,z),(h,x))$ to Q_2, there exists some i such that g,h,i is a triangle in G with $h<i<g$.
- The query Q_3 returns no answers over I. □

Theorem 17 does not cover the case of a UCQ containing acyclic non-free-connex queries with isomorphic bodies. Since this requires a more intricate analysis, we first restrict ourselves to such unions of size two. In the next section we discuss unions of two body-isomorphic CQs in general, and show in Theorem 28 that such a UCQ is tractable iff the UCQ is free-connex. By combining this with Theorem 17, we have the following dichotomy for the case of unions containing exactly two intractable CQs.

Theorem 19. *Let $Q = Q_1 \cup Q_2$ be a union of intractable CQs.*

- *If Q is free-connex, then* $\textsc{Enum}\langle Q\rangle \in \mathsf{DelayC_{lin}}$.
- *If Q is not free-connex, then* $\textsc{Enum}\langle Q\rangle \notin \mathsf{DelayC_{lin}}$, *assuming* MAT-MUL, HYPERCLIQUE *and* 4-CLIQUE.

4.2 Unions of Two Body-Isomorphic CQs

Consider a set of body-isomorphic CQs. As all of them have the same structure, either every CQ in this set is cyclic, or every CQ is acyclic. In the case of a union of two cyclic CQs, the UCQ is intractable according to Theorem 17. So in this section, we discuss the union of body-isomorphic acyclic CQs. Note that, unlike the previous section, we allow a CQ in the union to be free-connex. We first introduce a new notation for body-isomorphic UCQs that we use hereafter.

Consider a UCQ of the form $Q_1 \cup Q_2$, where there exists a body-isomorphism h from Q_2 to Q_1. That is, the CQs have the structure:

$$Q_1(\vec{v}_1) \leftarrow R_1(h(\vec{w}_1)), \dots, R_n(h(\vec{w}_n)),$$
$$Q_2(\vec{v}_2) \leftarrow R_1(\vec{w}_1), \dots, R_n(\vec{w}_n).$$

Applying h^{-1} to the variables of Q_1 does not affect evaluation, so we can rewrite Q_1 as $Q_1(h^{-1}(\vec{v}_1)) \leftarrow R_1(\vec{w}_1), \dots, R_n(\vec{w}_n)$. Since now the two CQs have exactly the same body, we can treat the UCQ as a query with one body and two heads:

$$Q_1(h^{-1}(\vec{v}_1)), Q_2(\vec{v}_2) \leftarrow R_1(\vec{w}_1), \dots, R_n(\vec{w}_n)$$

We use this notation from now on for UCQs containing only body-isomorphic CQs. Note that when treating a UCQ as one CQ with several heads, we can use the notation atoms(Q), as the atoms are the same for all CQs in the union, and the notation free(Q_i), as the free variables may differ between different queries Q_i in the union. With this notation at hand, we now inspect some examples of two body-isomorphic acyclic CQs.

Example 20. Consider $Q = Q_1 \cup Q_2$ with

$$Q_1(x,y,v) \leftarrow R_1(x,z), R_2(z,y), R_3(y,v), R_4(v,w) \text{ and}$$
$$Q_2(x,y,v) \leftarrow R_1(w,v), R_2(v,y), R_3(y,z), R_4(z,x).$$

Since Q_1 and Q_2 are body-isomorphic, Q can be rewritten as

$$Q_1(w,y,z), Q_2(x,y,v) \leftarrow R_1(w,v), R_2(v,y), R_3(y,z), R_4(z,x).$$

In this case we can use the same approach used for single CQs in Theorem 3, and show that this UCQ is not in $\mathsf{DelayC_{lin}}$ assuming MAT-MUL. Let A and B be binary representations of Boolean $n \times n$ matrices as explained in the beginning of this section. Define a database instance I with $R_1^I = A$, $R_2^I = B$, $R_3^I = \{1,\dots,n\} \times \{\perp\}$ and $R_4^I = \{(\perp,\perp)\}$. Since $Q_1(I)$ corresponds to the answers of AB, and $|Q_2(I)| = O(n^2)$, we cannot enumerate Q within the time bounds of $\mathsf{DelayC_{lin}}$ unless we can solve matrix multiplication in time $O(n^2)$. □

A union of two intractable body-isomorphic acyclic CQs may also be tractable. In fact, by adding a single variable to the heads in Example 20, we obtain a tractable UCQ.

Example 21. Let Q be the UCQ

$$Q_1(w,y,x,z), Q_2(x,y,w,v) \leftarrow R_1(w,v), R_2(v,y), R_3(y,z), R_4(z,x).$$

Both CQs are acyclic non-free-connex. As Q_2 provides the variables $\{v,w,y\}$ and Q_1 provides $\{x,y,z\}$, both CQs have free-connex union extension:

$$Q_1^+(w,y,x,z) \leftarrow R_1(w,v), R_2(v,y), R_3(y,z), R_4(z,x), P_1(v,w,y),$$
$$Q_2^+(x,y,w,v) \leftarrow R_1(w,v), R_2(v,y), R_3(y,z), R_4(z,x), P_2(x,y,z).$$

By Theorem 12 it follows that $\textsc{Enum}\langle Q\rangle \in \mathsf{DelayC_{lin}}$. □

Intuitively, the reason why the reduction of Example 20 fails in Example 21 is the fact that all the variables of the free-paths in one CQ, which are used to encode matrix multiplication, are free in the other CQ. Indeed, if we encode matrices A and B to the relations of the free path w,v,y in Q_1, there can be n^3 answers to Q_3. The answer set in this case is too large to contradict the assumed lower bound for matrix multiplication. As it turns out, there are cases where we cannot reduce matrix multiplication to a union in this manner, and yet we can show that it is intractable using an alternative problem.

Example 22. Let Q be the UCQ

$$Q_1(x,y,t), Q_2(x,y,w) \leftarrow R_1(x,w,t), R_2(y,w,t).$$

This union is intractable under the 4-CLIQUE assumption. For a given graph $G = (V,E)$ with $|V| = n$, we compute

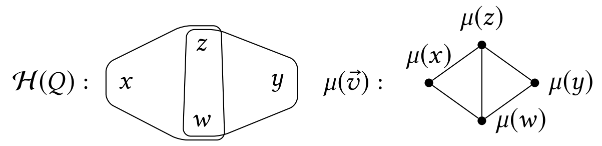

Figure 3: If $\mu|_{free(Q_i)} \in Q(I)$, the induced subgraph of $\mu(w,x,y,z)$ forms a clique where one edge might be missing.

the set T of all triangles in G in time n^3. Define a database instance I as $R_1^I = R_2^I = T$. For every output $\mu|_{free(Q_i)}$ in $Q(I)$ with $i \in \{1,2\}$, we know that $(\mu(x),\mu(z),\mu(w))$ and $(\mu(y),\mu(w),\mu(z))$ are triangles. If $\mu(x) \neq \mu(y)$, this means that $\mu((x,y)) \in E$ if and only if $(\mu(x),\mu(y),\mu(w),\mu(z))$ forms a 4-clique (see Figure 3). Since there are $O(n^3)$ answers to Q, if ENUM$\langle Q\rangle \in$ DelayC$_{\text{lin}}$, we can check whether $\mu((x,y)) \in E$ for every answer in $Q(I)$, and determine whether a 4-clique appears in G in time $O(n^3)$. □

Note that we can use the 4-CLIQUE assumption in Example 22, since, in addition to the free-path variables, there is another variable in both free-path relations. We now generalize the observations from the examples.

Definition 23. Let $Q = Q_1 \cup Q_2$ be a UCQ where Q_1 and Q_2 are body-isomorphic.

- Q_1 is said to be *free-path guarded* if for every free-path P in Q_1, we have that $var(P) \subseteq$ free(Q_2).
- Consider a path $P = (u_1,\dots u_k)$ in Q_1. Two atoms $R_1(\vec{v}_1)$ and $R_2(\vec{v}_2)$ of Q_1 are called *subsequent P-atoms* if $\{u_{i-1},u_i\} \subseteq \vec{v}_1$ and $\{u_i,u_{i+1}\} \subseteq \vec{v}_2$ for some $1 < i < k$.
- Q_1 is said to be *bypass guarded* if for every free-path P in Q_1 and variable u that appears in two subsequent P-atom, we have that $u \in$ free(Q_2).

Note that every free-connex CQ is trivially free-path guarded and bypass guarded. In the following two lemmas, we show that if some CQ in a union is either not free-path guarded or bypass-guarded, then the UCQ is intractable.

In Example 20, Q_1 is not free-path guarded, so we can use the reduction from matrix multiplication.

Lemma 24. *Let $Q = Q_1 \cup Q_2$ be a UCQ of self-join free body-isomorphic acyclic CQs. If Q_1 is not free-path guarded, then* ENUM$\langle Q\rangle$ *is not in* DelayC$_{\text{lin}}$, *assuming MAT-MUL.*

PROOF SKETCH. Let A and B be Boolean matrices and let $P = (z_0,\dots,z_{k+1})$ be a free-path in Q_1. In the proof of part (2) in Theorem 3, the matrix multiplication AB is reduced to ENUM$\langle Q_1\rangle$ by encoding the matrices to the relations of Q_1 [2, Lemma 26]. They define $V_x = \{z_0\}$, $V_z = \{z_1,\dots,z_k\}$ and $V_y = \{z_{k+1}\}$. Since P is chordless, no atom contains both a variables of V_x and a variable of V_y. For every atom $R_i(\vec{v}_i)$ that has no variables in V_y and for every 1 in the matrix A in indices (a,b), we add a tuple to R_i where every variable in V_x corresponds to the value a, every variable in V_z corresponds to the value b, and all other variables are assigned the constant $\perp$. Similarly, we encode the matrix B to the other atoms. Since $V_x \cup V_y$ are free, we get the answers to matrix multiplication in $Q_1(I)$, and as no variable in V_z is free, we do not have duplicates. We extend this encoding as follows. If z_0 or z_{k+1} are not free in Q_2, we use the exact same reduction. In this case, $V_x \cap$ free$(Q_2) = \emptyset$ or $V_y \cap$ free$(Q_2) = \emptyset$. Otherwise, let z_i be the first variable in P that is not free in Q_2. We define $V_x = \{z_0,\dots,z_{i-1}\}$, $V_z = \{z_i\}$ and $V_y = \{z_{i+1}\dots,z_{k+1}\}$, and we have that $V_z \cap$ free$(Q_2) = \emptyset$. With this encoding Q_1 still computes matrix multiplication, and as the CQs both have the same body, we have that Q_2 has at most n^2 answers. To distinguish the answers of Q_1 from those of Q_2, we can concatenate the variable names to the values, as we did in Lemma 14. Thus, enumerating ENUM$\langle Q\rangle$ in DelayC$_{\text{lin}}$ solves matrix multiplication in $O(n^2)$, which contradicts MAT-MUL. □

In Example 22, we encounter a UCQ where both CQs are free-path guarded, but Q_1 is not bypass guarded. In every UCQ with this property we can encode 4-CLIQUE.

Lemma 25. *Let $Q = Q_1 \cup Q_2$ be a UCQ of self-join free body-isomorphic acyclic CQs. If Q_1 and Q_2 are free-path guarded and Q_1 is not bypass guarded, then* ENUM$\langle Q\rangle \notin$ DelayC$_{\text{lin}}$, *assuming 4-CLIQUE.*

PROOF. Let $G = (V,E)$ be a graph with $|V| = n$. We show how to solve the 4-CLIQUE problem on G in time $O(n^3)$ if ENUM$\langle Q\rangle$ is in DelayC$_{\text{lin}}$. Let P be a free-path in Q_1 and let $u \notin$ free(Q_2) such that u appears in two subsequent P-atoms. It can be shown that, under the conditions of this lemma, P is of the form (z_0,z_1,z_2). Let R_1 and R_2 be atoms with $\{z_0,z_1,u\} \subseteq var(R_1)$ and $\{z_1,z_2,u\} \subseteq var(R_1)$. Further let (a,b,c) be a triangle in G. We define a mapping $\tau_{(a,b,c)}$ on variables of Q in order to encode this triangle to tuples $r \in R^I$ of a database over Q, such that $var(R)$ either contains $\{z_0,z_1,u\}$ or $\{z_1,z_2,u\}$. That is, given $v \in var(Q)$, we define

$$\tau_{a,b,c}(v) = \begin{cases} a & \text{if } v = z_0 \text{ or } v = z_2, \\ b & \text{if } v = z_1, \\ c & \text{if } v = u, \\ \perp & \text{otherwise.} \end{cases}$$

For every atom $R(v_1,\dots,v_s) \in$ atoms(Q), we define

$$R^I = \{(\tau_{a,b,c}(v_1),\dots,\tau_{a,b,c}(v_s)) \mid (a,b,c) \text{ a triangle in } G\}.$$

Note that $|R^I| \in O(n^3)$, as there are at most n^3 triangles in G, and that we can construct I within $O(n^3)$ steps. Now consider a homomorphism $\mu : var(Q) \to V$ mapping Q into the database. Since $(\mu(z_0),\mu(z_1),\mu(u))$ and $(\mu(z_1),\mu(z_2),\mu(u))$ form triangles in G, we have that G contains a 4-clique iff $(\mu(z_0), \mu(z_2)) \in E(Q)$. As $z_0, z_1 \in \text{free}(Q_1)$ it suffices to check every $\mu|_{free(Q_1)} \in Q(I)$ for this property. We have that $\{z_0, z_1, z_2, u\}$ is neither contained in free(Q_1) nor in free(Q_2). Thus, $|Q(I)| \in O(n^3)$. If $\text{Enum}\langle Q \rangle$ is in $\text{DelayC}_{\text{lin}}$, we can construct I, output $Q(I)$ and check every output for an edge of the form $(\mu(z_0), \mu(z_2))$ in time $O(n^3)$, which contradicts 4-clique. □

We show that any UCQ that is not covered by Lemma 24 and Lemma 25 is in fact union-free-connex. To prove this, we need a structural property given as follows.

Lemma 26. *Let $Q = Q_1 \cup Q_2$ be a UCQ of body-isomorphic acyclic CQs, where Q_1 and Q_2 are free-path guarded, and Q_1 is bypass guarded, and let P be a free-path in Q_1. There exists a join-tree T for Q with a subtree T_P such that $var(P) \subseteq var(T_P)$, and every variable that appears in two different atoms of T_P is in* free(Q_2).

Proof. Consider a path $A_1, \ldots, A_s$ between two atoms on a join tree. We define a *contraction step* for a path of length 2 or more: if there is A_j such that $A_j \cap A_{j+1} \subseteq A_1 \cap A_s$, then remove the edge (A_j, A_{j+1}) and add the edge (A_1, A_s). The new graph is still a join-tree since all of the atoms on the path between A_1 and A_s contain $A_1 \cap A_s$, and in particular they contain $A_j \cap A_{j+1}$. The unique path on the join-tree between the atoms A_1 and A_s is now of length one. A path on a join-tree is said to be *fully-contracted* if none of its subpaths can be contracted.

Now let T be a join-tree of Q, and let $P = (z_0, \ldots, z_{k+1})$. We consider some path in T between an atom containing $\{z_0, z_1\}$ and an atom containing $\{z_k, z_{k+1}\}$. Take the unique subpath T_P of it containing only one atom with $\{z_0, z_1\}$ and one atom with $\{z_k, z_{k+1}\}$, and fully contract it. Note that T_P contains $var(P)$ due to the running intersection property.

First, we claim that every variable u that appears in two or more atoms of T_P is part of a chordless path from z_0 to z_{k+1}. We first show a chordless path from u to z_{k+1}. Denote the last atom on T_P containing u by A_i. If A_i contains z_{k+1}, we are done. Otherwise, consider the subpath A_{i-1}, A_i, A_{i+1}. Since it is fully contracted, $A_i \cap A_{i+1} \nsubseteq A_{i-1} \cap A_{i+1}$. This means that there is a variable v in A_i and in A_{i+1} that does not appear in A_{i-1}. Now consider the last atom containing v, and continue with the same process iteratively until reaching z_{k+1}. Do the same symmetrically to find a chordless path from u to z_0. Note that the concatenation of the two paths is chordless by construction and since z_0 and z_{k+1} are not neighbors.

Assume by contradiction that some variable $u \notin \text{free}(Q_2)$ appears in two distinct atoms of T_P. There is a chordless path from z_0 to z_{k+1} that contains u. Take a subpath of it starting with the first variable before u which is in free(Q_2), and ending with the first variable after u which is in free(Q_2). This is a free-path in Q_2, and since Q_2 is free-path guarded, $u \in \text{free}(Q_1)$. Next consider two neighboring atoms on T_P that contain u. There exists some z_i that appears in both atoms. Note that $i > 0$ and $i < k+1$ since the path only contains one atom with z_0 and one atom with z_{k+1}. Since Q_1 is bypass guarded, u is not a neighbor of both z_{i-1} and z_{i+1}. Without loss of generality, assume it is not a neighbor of z_{i+1}. Then there is a chordless path $(u, z_i, z_{i+1}, \ldots, z_{k+1})$. Since $u \in \text{free}(Q_1)$, it is a free-path. This contradicts the fact that Q_1 is free-path guarded since $u \notin \text{free}(Q_2)$. □

We are now ready to show that the properties free-path guardedness and bypass guardedness imply free-connexity.

Lemma 27. *Let $Q = Q_1 \cup Q_2$ be a UCQ of body-isomorphic acyclic CQs. If Q_1 and Q_2 are both free-path guarded and bypass guarded, then Q is free-connex.*

Proof. We describe how to iteratively build a union extension for each CQ. In every step we take one free-path among the queries in Q and add a virtual atom in order to eliminate this free-path. We show that this eventually leads to free-connex union extensions.

Let $P = (z_0, \ldots, z_{k+1})$ be a free-path in Q_1. Take T_P according to Lemma 26, and denote by V_P the variables $var(P)$ and all variables that appear in more than one atom of T_P. First we claim that Q_2 provides V_P. It is guaranteed that $V_P \subseteq \text{free}(Q_2)$, so we only need to show that Q_2 is acyclic V_P-connex. Take the join-tree T from Lemma 26. For every vertex A_i in T_P, add another vertex with $A'_i = var(A_i) \cap V_P$ and an edge (A_i, A'_i). Then, for every edge (A_i, A_j) in T_P, add the edge (A'_i, A'_j) and remove (A_i, A_j). The running intersection property is maintained since for every edge (A_i, A_j) removed, $var(A_i) \cap var(A_j) \subseteq V_P$, meaning that all vertices on the path A_i, A'_i, A'_j, A_j contain $var(A_i) \cap var(A_j)$. The subtree containing the new vertices contains exactly V_P.

We add the atom $R(V_P)$ to both Q_1 and Q_2 and obtain Q_1^+ and Q_2^+ respectively. After this extension there are no free-paths that start in z_0 and end in z_{k+1} since they are now neighbors. If both of the CQs are now free-connex, then we are done. Otherwise, we use the extension iteratively, as one can show that for the UCQ $Q_1^+ \cup Q_2^+$, both Q_1^+ and Q_2^+ are free-path guarded and bypass guarded. Note that after a free-path from z_0 to z_{k+1} is treated, and even after future extension, there will never be a free-path from z_0 to z_{k+1} since they are now neighbors. Since there is a finite number of variable pairs, at some point all pairs that have a free-path between them are resolved, this process stops, and there are no free-paths. □

Since Lemma 24, Lemma 25 and Lemma 27 cover all cases of a union of two self-join-free body-isomorphic acyclic CQs, we have a dichotomy that characterizes the UCQs discussed in this section.

Theorem 28. *Let $Q = Q_1 \cup Q_2$ be a UCQ of self-join free body-isomorphic CQs.*

- *If Q_1 and Q_2 are free-path guarded and bypass guarded, then Q is free-connex and* $\textsc{Enum}\langle Q\rangle \in \mathsf{DelayC}_{\mathsf{lin}}$.
- *Otherwise, Q is not free-connex and* $\textsc{Enum}\langle Q\rangle$ *is not in* $\mathsf{DelayC}_{\mathsf{lin}}$*, under the assumptions* HYPERCLIQUE*,* MAT-MUL *and* 4-CLIQUE*.*

Proof. By Theorem 17, if the CQs are cyclic, $\textsc{Enum}\langle Q\rangle$ is not in $\mathsf{DelayC}_{\mathsf{lin}}$ assuming HYPERCLIQUE. Now assume that the CQs are acyclic. By Lemma 27, if Q_1 and Q_2 are both free-path guarded and bypass guarded, then Q is free-connex, and $\textsc{Enum}\langle Q\rangle \in \mathsf{DelayC}_{\mathsf{lin}}$ by Theorem 12. By Lemma 24 and Lemma 25, if one of Q_1 and Q_2 are not free-path guarded or not bypass guarded, then $\textsc{Enum}\langle Q\rangle$ is not in $\mathsf{DelayC}_{\mathsf{lin}}$ (assuming MAT-MUL and 4-CLIQUE), and then Q is not free-connex by Theorem 12. □

5 TOWARDS A DICHOTOMY

In this section we examine the next steps that are required to fully characterize which UCQs are in $\mathsf{DelayC}_{\mathsf{lin}}$. We pinpoint some of the difficulties that must be tackled when formulating such a dichotomy, accompanied by examples. In Section 5.1 we discuss unions containing only acyclic CQs, and in Section 5.2 we discuss those that contain at least one cyclic CQ.

5.1 Unions of Acyclic CQs

We inspect two ways of extending the results of Section 4.2. The first such extension is to a union of two CQs that are not body-isomorphic. If there is an intractable CQ Q_1 in the union where for every other CQ Q_i in the union there is no body-homomorphism from Q_i to Q_1, we can reduce Q_1 to the union as described in Lemma 14. Since Q_1 is intractable, the union is intractable too. In case there is a body-homomorphism to the hard queries, one might think that it is sufficient for intractability to have unguarded intractable structures similarly to the previous section. This is incorrect.

Example 29. Let $Q = Q_1 \cup Q_2$ with

$$Q_1(x,y,w) \leftarrow R_1(x,z), R_2(z,y), R_3(y,w) \text{ and}$$
$$Q_2(x,y,w) \leftarrow R_1(x,t_1), R_2(t_2,y), R_3(w,t_3).$$

The query Q_1 is acyclic non-free-connex, while Q_2 is free-connex. Further Q_1 is not contained in Q_2 but there is a body-homomorphism from Q_2 to Q_1. The variable z is part of the free-path (x,z,y) in Q_1, but the variables t_1 and t_2 that map to it via the body-homomorphism are not free in Q_2. If we extend the notion of guarding to non-body-isomorphic CQs in the natural way, the free-path (x,z,y) is not guarded. Nevertheless, we cannot compute matrix multiplication in $O(n^2)$ time by encoding it to the free-path (x,z,y) like before. Over such a construction, there can be n^3 many results to Q_2 as w and y are not connected in Q_2, and they can have distinct values. This is not an issue when discussing body-isomorphic CQs. We do not know whether this example is in $\mathsf{DelayC}_{\mathsf{lin}}$. □

A future characterization of the union of CQs that are not body-isomorphic would need an even more careful approach than the one used in Section 4.2 in order to handle the case that variables mapping to the free-path are not connected via other variables mapping to the free-path.

A second way of extending Section 4.2 is to consider more than two body-isomorphic acyclic CQs. This case may be tractable or intractable, and for some queries of this form, we do not have a classification yet. Example 13 shows a tractable union of such form, while the following example shows that free-paths that share edges can be especially problematic within a union.

Example 30. Let $k \geq 4$ and consider the UCQ Q containing k body-isomorphic CQs, with an atom $R_i(x_i,z)$ for every $1 \leq i \leq k-1$. The heads are all possible combinations of $k-1$ out of the k variables of the query, $\{z, x_1, \ldots, x_{k-1}\}$. In the case of $k = 4$ we have the following UCQ:

$$Q_1(x_1,x_2,x_3), Q_2(x_1,x_2,z), Q_3(x_1,x_3,z), Q_4(x_2,x_3,z)$$
$$\leftarrow R_1(x_1,z), R_2(x_2,z), R_3(x_3,z).$$

The query Q_1 has free-paths (x_i,z,x_j) between all possible pairs of i and j. In this case, enumerating the solutions to the UCQ is not in $\mathsf{DelayC}_{\mathsf{lin}}$, assuming 4-CLIQUE: Encode each relation with all edges in the input graph, and concatenate the variable names. That is, for every edge (u,v) in the graph, add $((u,x_1),(v,z))$ to R_1. By concatenating variable names, we can identify which solutions come from which CQ, and ignore the answers to all CQs other than Q_1. The answers to Q_1 gives us 3 vertices that have a common neighbor. We can check in constant time if every pair of the 3 vertices are neighbors, and if so, we found a 4-clique. As there can be $O(n^3)$ solutions the the UCQ, we solve 4-clique in $O(n^3)$ time. This proves that the UCQ is intractable assuming 4-CLIQUE.

With the same strategy, we can solve k-clique in time $O(n^{k-1})$, but this does not result in a lower bound for large k values: It is assumed that one can not find a k-clique in time $n^{\frac{\omega k}{3}-o(1)}$, which does not contradict an $O(n^{k-1})$ algorithm. However, this reduction does not seem to fully capture the hardness of this query, as it encodes all relations with the same set of edges. We do not know if queries of the structure given here are hard in general, or if they become easy for larger k values, as any current approach that we know of for constant delay enumeration fails for them. □

We briefly describe some classification we can achieve when we exclude cases like Example 30. First note that when generalizing the notion of guarding free-paths to unions of several CQs, a free-path does not have to be guarded by a single CQ.

Definition 31. Let $Q = Q_1 \cup \ldots \cup Q_n$ be a union of body-isomorphic CQs, and let $P = (z_0, \ldots, z_{k+1})$ be a free-path in Q_1. We say that a set $\mathcal{U} \subseteq 2^{var(P)}$ is a *union guard* for P if:

- $\{z_0, z_{k+1}\} \in \mathcal{U}$.
- For every $\{z_a, z_c\} \subseteq u \in \mathcal{U}$ with $a + 1 < c$, we have that $\{z_a, z_b, z_c\} \in \mathcal{U}$ for some $a < b < c$.
- For every $u \in \mathcal{U}$, we have $u \subseteq \text{free}(Q_i)$ for some $1 \leq i \leq n$.

We now show that if a UCQ contains a free-path with no union guard, then the entire union is intractable.

Theorem 32. *Let $Q = Q_1 \cup \ldots \cup Q_n$ be a UCQ of body-isomorphic acyclic CQs. If there exists a free-path in Q_1 that is not union guarded, then* Enum$\langle Q \rangle \notin$ DelayC$_{\text{lin}}$*, assuming* MAT-MUL.

Proof sketch. If some path $P = (z_0, \ldots, z_{k+1})$ is not union guarded, then there exist some a and b such that $0 \leq a + 1 < b \leq k + 1$, and $\{z_a, z_b\} \subseteq \text{free}(Q_r)$ for some $Q_r \in Q$, but for all $Q_s \in Q$ and for all $a < b < c$ we have $\{z_a, z_b, z_c\} \notin \text{free}(Q_s)$. Then, $P' = (z_a, z_{a+1}, \ldots, z_c)$ is a free-path of Q_r. We can use a construction similar to Lemma 24 on the path P' to compute matrix multiplication in $O(n^2)$ by evaluating the union in linear time preprocessing and constant delay. □

As we do not know of a classification for Example 30, we restrict the UCQs we consider to cases where the free-paths within each CQ do not share variables. Then, we manage to obtain a similar characterization to that of Section 4.2.

Definition 33. Let $Q = Q_1 \cup \ldots \cup Q_n$ be a union of body-isomorphic CQs, and let P be a free-path of Q_1. We say that P is *isolated* if the following two conditions hold:

- Q is $var(P)$-connex and
- $var(P') \cap var(P) = \emptyset$ for all free-paths $P' \neq P$ in Q_1 .

Note that isolated is a stronger property than bypass-guarded. Given a free-path P, A bypass-guarded query can have a variable in two subsequent P-atoms as long as this variable is free in another CQ. An isolated free-path cannot have such a variable at all. If all free-paths in a union of body-isomorphic acyclic CQs are union guarded and isolated, we can show that the union is tractable.

Theorem 34. *Let $Q = Q_1 \cup \ldots \cup Q_n$ be a union of body-isomorphic acyclic CQs. If every free-path in Q is union guarded and isolated, then* Enum$\langle Q \rangle \in$ DelayC$_{\text{lin}}$.

Proof sketch. If every free-path in Q is union guarded and isolated, then we can apply an iterative process that eliminates free-paths without introducing new ones. Imagine a union guard as the nodes of a tree, where the first condition of Definition 31 defines the root and the second condition defines the children of each node. The leaves of such a tree are of the form $\{z_i, z_{i+1}, z_{i+2}\}$. Using the third condition of Definition 31 and the first condition of Definition 33, it is possible to show via a bottom-up induction on this tree that every vertex is provided by some union extension of a CQ in the union. Adding these variables sets as virtual atoms results in a union extension without the treated free-path. Due to the second condition of Definition 33, this extension does not introduce new free-paths. As we can apply this process to all free-paths, we obtain a free-connex union extension for every CQ in the union, and conclude that Q is free-connex. According to Theorem 12, Enum$\langle Q \rangle \in$ DelayC$_{\text{lin}}$. □

Following Theorem 32 and Theorem 34, it is left to handle cases like Example 30, of unions containing body-isomorphic acyclic CQs where some free-path is union guarded but not isolated. After solving this case, and regarding unions of intractable CQs, we should also explore unions containing body-isomorphic acyclic CQs but also other intractable CQs.

5.2 Unions Containing Cyclic CQs

We now discuss UCQs containing at least one cyclic query. We first mention that many of the observations we have regarding acyclic CQs also apply here. In the following examples, Q_1 is cyclic while Q_2 is free-connex. Example 35 shows that unions containing cyclic CQs may be tractable and covered by Theorem 12. Example 36 demonstrates that it is not enough to resolve the cyclic structures in CQs, but that we should also handle free-paths. Finally, Example 37 shows that, much like Example 29 in the acyclic case, even if all intractable structures are unguarded, the original reductions showing the intractability of single CQs may not work.

Example 35. Let $Q = Q_1 \cup Q_2$ with

$$\begin{aligned} Q_1(x, y, z, w) \leftarrow & R_1(y, z, w, x), R_2(t, y, w), \\ & R_3(t, z, w), R_4(t, y, z), \\ Q_2(x, y, z, w) \leftarrow & R_1(x, z, w, v), R_2(y, x, w). \end{aligned}$$

The query Q_2 provides $\{t, y, z, w\}$ to Q_1. Adding the virtual atom $R'(t, y, z, w)$ to Q_1 results in a free-connex union-extension, so the union $Q_1 \cup Q_2$ is tractable. □

Example 36. Let $Q = Q_1 \cup Q_2$ with

$$Q_1(x, y, v) \leftarrow R_1(v, z, x), R_2(y, v), R_3(z, y) \text{ and}$$
$$Q_2(x, y, v) \leftarrow R_1(y, v, z), R_2(x, y).$$

The union $Q_1 \cup Q_2$ is intractable despite the fact that Q_2 guards and provides the cycle variables $\{v, y, z\}$. This is due to the unguarded free-path (x, z, y) in Q_1. Similarly to Example 20, we can encode matrix multiplication to x, z, y. □

Example 37. Let $Q = Q_1 \cup Q_2$ with

$$Q_1(x,z,y,v) \leftarrow R_1(x,z,v), R_2(z,y,v), R_3(y,x,v) \text{ and}$$
$$Q_2(x,z,y,v) \leftarrow R_1(x,z,v), R_2(y,t_1,v), R_3(t_2,x,v).$$

We do not know the complexity of this example. As Q_2 does not have a free variable that maps to y via a homomorphism, Q_1 is not union-free-connex, so we cannot conclude using Theorem 12 that $Q_1 \cup Q_2$ is tractable. The only intractable structure in Q_1 is the cycle x, y, z, but encoding the triangle finding problem to this cycle in Q_1, like we did in Example 18, could result in n^3 answers to Q_2. This means that if the input graph has triangles, we are not guaranteed to find one in $O(n^2)$ time by evaluating the union efficiently. □

In addition to these issues, in the cyclic case, even if the original intractable structures are resolved, the extension may introduce new ones. Resolving the following example in general is left for future work.

Example 38. We start with a single case of a general example. Let $Q = Q_1 \cup Q_2$ with

$$Q_1(x_2,x_3,x_4) \leftarrow R_1(x_2,x_3,x_4), R_2(x_1,x_3,x_4), R_3(x_1,x_2,x_4),$$
$$Q_2(x_2,x_3,x_4) \leftarrow R_1(x_2,x_3,x_1), R_2(x_4,x_3,v).$$

There is a body-homomorphism from Q_2 to Q_1, but Q_1 is not contained in Q_2. Q_1 is cyclic, as it has the cycle (x_1, x_2, x_3). This is the only intractable structure in Q_1, i.e. $\mathcal{H}(Q_1)$ does not contain a hyperclique or a free-path. The query Q_2 is both free-connex and $\{x_2, x_3, x_4\}$-connex, and it provides $\{x_1, x_2, x_3\}$ to Q_1. Nevertheless, extending Q_1 with a virtual atoms $R(x_1, x_2, x_3)$ does not result in a free-connex CQ. Even though the extension "removes" all intractable structures from Q_1, it is intractable as it introduces a new intractable structure, namely the hyperclique $\{w, x, y, z\}$.

In this case, we have $\text{E}\textsc{num}\langle Q_1 \cup Q_2\rangle \notin \mathsf{DelayC_{lin}}$ assuming 4-CLIQUE, with a reduction similar to that of Example 22: Given an input graph, compute all triangles and encode them to the three relations. For every triangle $\{a, b, c\}$, add the tuple $((a, x_2), (b, x_3), (c, x_4))$ to R_1, $((a, x_1), (b, x_3), (c, x_4))$ to R_2 and $((a, x_1), (b, x_2), (c, x_4))$ to R_3. By concatenating variable names, we can identify which solutions correspond to which CQ, and thus are able to ignore the answers to Q_2. The answers to Q_1 represent 3 vertices that appear in a 4-clique: For every answer $((b, x_2), (c, x_3), (d, x_4))$ to Q_1, we know that $((b, x_2), (c, x_3), (d, x_4)) \in R_1$, and there exists some a with $((a, x_1), (c, x_3), (d, x_4)) \in R_2$ and $((a, x_1), (b, x_2), (d, x_4)) \in R_3$. This means that $\{a, b, c, d\}$ is a 4-clique. In the opposite direction, for every 4-clique $\{a, b, c, d\}$ we have that $\{b, c, d\}$, $\{a, c, d\}$ and $\{a, b, d\}$ are triangles. By construction, the tuple $((b, x_2), (c, x_3), (d, x_4))$ is an answer to Q_1. As there are $O(n^3)$ triangles and there can be at most $O(n^3)$ solutions to Q_2, we solve 4-clique in $O(n^3)$ time. This proves that the UCQ is intractable assuming 4-CLIQUE.

This example can be generalized to higher orders. There, we do not have a similar lower bound. Consider the union of the following:

$$Q_1(x_2, \ldots, x_k) \leftarrow \{R_i(\{1, \ldots, k\} \setminus \{i\}) \mid 1 \le i \le k-1\}$$
$$Q_2(x_2, \ldots, x_k) \leftarrow R_1(x_2, \ldots, x_{k-1}, x_1),$$
$$R_2(x_k, x_3, \ldots, x_{k-1}, v).$$

Again, the query Q_1 is cyclic and Q_2 is free-connex. Even though Q_2 provides $\{x_1, \ldots, x_{k-1}\}$, adding a virtual atom with these variables does not result in a free-connex extension, as this extension is again cyclic. Just like in Example 30, we can encode k-clique to this example in general, but this does not imply a lower bound for large k values. □

6 CONCLUSIONS

In this paper we study the enumeration complexity of UCQs with respect to $\mathsf{DelayC_{lin}}$. We formalized how CQs within a union can make each other easier by providing variables, and we introduced union extensions. Then, we defined free-connex UCQs, and showed that these are tractable. In particular, we demonstrated that UCQs containing only intractable CQs may be tractable.

We showed that in case of a union of two intractable CQs or two acyclic body-isomorphic CQs, free-connexity fully captures the tractable cases. Nevertheless, achieving a full classification of UCQs remains an open problem. In Section 5 we described the next steps we plan to tackle in this vein, and provided examples with unknown complexity. Resolving these examples is a necessary step on the way to a future dichotomy.

In this paper, we only considered time bounds. The class CD∘Lin describes the problems that can be solved with the same time bounds, but with the additional restriction that the available space for writing during the enumeration phase is constant. Evaluating free-connex CQs is in this class, and Kazana offers a comparison between $\mathsf{DelayC_{lin}}$ and CD∘Lin [12, Section 8.1.2]. The lower bounds we showed naturally hold for CD∘Lin. The tractability of unions containing only free-connex CQs also holds for this class. However, the memory we used in our techniques for the tractable UCQs that do not contain only tractable CQs may increase in size by a constant with every new answer. An interesting question is whether we can achieve the same time bounds when restricting the memory according to CD∘Lin.

ACKNOWLEDGMENTS

The work of Nofar Carmeli was supported by the Irwin and Joan Jacobs Fellowship. The work of Markus Kröll was supported by the Austrian Science Fund (FWF): P30930-N35, W1255-N23.

REFERENCES

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

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

[3] Christoph Berkholz, Jens Keppeler, and Nicole Schweikardt. 2017. 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*. 303–318.

[4] Christoph Berkholz, Jens Keppeler, and Nicole Schweikardt. 2018. Answering UCQs under Updates and in the Presence of Integrity Constraints. In *21st International Conference on Database Theory, ICDT 2018, March 26-29, 2018, Vienna, Austria*. 8:1–8:19.

[5] Johann Brault-Baron. 2013. *De la pertinence de lâĂŹénumération: complexité en logiques propositionnelle et du premier ordre.* Ph.D. Dissertation. Université de Caen.

[6] Nofar Carmeli and Markus Kröll. 2018. Enumeration Complexity of Conjunctive Queries with Functional Dependencies. In *ICDT 2018, March 26-29, 2018, Vienna, Austria (LIPIcs)*, Benny Kimelfeld and Yael Amsterdamer (Eds.), Vol. 98. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 11:1–11:17.

[7] Arnaud Durand and Etienne Grandjean. 2007. First-order Queries on Structures of Bounded Degree Are Computable with Constant Delay. *ACM Trans. Comput. Logic* 8, 4, Article 21 (Aug. 2007). https://doi.org/10.1145/1276920.1276923

[8] Fernando Florenzano, Cristian Riveros, Martín Ugarte, Stijn Vansummeren, and Domagoj Vrgoc. 2018. Constant Delay Algorithms for Regular Document Spanners. In *PODS*. ACM, 165–177.

[9] Jörg Flum, Markus Frick, and Martin Grohe. 2002. Query evaluation via tree-decompositions. *J. ACM* 49, 6 (2002), 716–752.

[10] Etienne Grandjean. 1996. Sorting, Linear Time and the Satisfiability Problem. *Ann. Math. Artif. Intell.* 16 (1996), 183–236. https://doi.org/10.1007/BF02127798

[11] Muhammad Idris, Martin Ugarte, and Stijn Vansummeren. 2017. The Dynamic Yannakakis Algorithm: Compact and Efficient Query Processing Under Updates. In *Proceedings of the 2017 ACM International Conference on Management of Data (SIGMOD ’17)*. ACM, New York, NY, USA, 1259–1274. https://doi.org/10.1145/3035918.3064027

[12] Wojciech Kazana. 2013. *Query evaluation with constant delay.* Theses. École normale supérieure de Cachan - ENS Cachan. https://tel.archives-ouvertes.fr/tel-00919786

[13] François Le Gall. 2014. Powers of Tensors and Fast Matrix Multiplication. In *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation (ISSAC ’14)*. ACM, New York, NY, USA, 296–303. https://doi.org/10.1145/2608628.2608664

[14] Andrea Lincoln, Virginia Vassilevska Williams, and R. Ryan Williams. 2018. 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, New Orleans, LA, USA, January 7-10, 2018*. 1236–1252.

[15] Matthias Niewerth and Luc Segoufin. 2018. Enumeration of MSO Queries on Strings with Constant Delay and Logarithmic Updates. In *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, Houston, TX, USA, June 10-15, 2018*, Jan Van den Bussche and Marcelo Arenas (Eds.). ACM, 179–191. https://doi.org/10.1145/3196959.3196961

[16] Nicole Schweikardt, Luc Segoufin, and Alexandre Vigny. 2018. Enumeration for FO Queries over Nowhere Dense Graphs. In *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, Houston, TX, USA, June 10-15, 2018*, Jan Van den Bussche and Marcelo Arenas (Eds.). ACM, 151–163. https://doi.org/10.1145/3196959.3196971

[17] Luc Segoufin. 2015. Constant Delay Enumeration for Conjunctive Queries. *SIGMOD Rec.* 44, 1 (May 2015), 10–17. https://doi.org/10.1145/2783888.2783894

[18] Luc Segoufin and Alexandre Vigny. 2017. Constant Delay Enumeration for FO Queries over Databases with Local Bounded Expansion. In *ICDT 2017 (Leibniz International Proceedings in Informatics (LIPIcs))*, Michael Benedikt and Giorgio Orsi (Eds.), Vol. 68. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 20:1–20:16.

[19] Y. Strozecki. 2010. *Enumeration complexity and matroid decomposition.* Ph.D. Dissertation. Université Paris Diderot - Paris 7.

[20] Mihalis Yannakakis. 1981. Algorithms for Acyclic Database Schemes. In *Proceedings of the Seventh International Conference on Very Large Data Bases - Volume 7 (VLDB ’81)*. VLDB Endowment, 82–94. http://dl.acm.org/citation.cfm?id=1286831.1286840