

Accessing Answers to Conjunctive Queries with Ideal Time Guarantees

Nofar Carmeli¹

¹*Inria, LIRMM, University of Montpellier, CNRS, France*

Abstract

When can we answer conjunctive queries with ideal time guarantees? We will start this talk by examining different kinds of query-answering tasks and the connections between them. These tasks include enumerating all answers, sampling answers without repetitions, and simulating a sorted array of the answers. From a data complexity point of view, the ideal time guarantees for these tasks are constant time per answer following a linear preprocessing (required to read the database input). Our goal is to avoid the polynomial preprocessing required to produce all answers. We will then have an example-based discussion of the complexity landscape for these tasks. In particular, we will see how self-joins, constraints and unions can play a crucial role in determining the complexity and designing efficient algorithms.

Keywords


conjunctive queries, fine-grained complexity, constant-delay enumeration

1. Introduction: Dichotomy for Conjunctive Queries

We would like to know what is the best time complexity we can achieve for answering a given query over an input database. If the query is not Boolean, it can have many answers. In fact, the number of answers can be orders of magnitude larger than the size of the input database, and so we cannot expect to have a linear-time algorithm (linear in the size of the input) that produces all answers. It is therefore natural to use enumeration measures and bound the delay between successive answers. Since we must read the entire input database before we can decide whether the query has any answers, the best we can hope for is linear preprocessing and constant delay. Which queries can be answered with these ideal time guarantees? This question was considered for Conjunctive Queries (CQs).


Theorem 1 ([1],[2]). *Let Q be a CQ without self-joins. There exists an algorithm enumerating the answers of Q with linear preprocessing and constant delay if and only if Q is free-connex acyclic, assuming the SPARSEBMM and HYPERCLIQUE hypotheses.*


Every query has an associated hypergraph with a vertex for each variable and a hyperedge for each atom. *Acyclic* queries are those that have an α -acyclic hypergraph. If all atoms are binary, this simply means that the graph does not contain cycles. *Free-connex* acyclic queries also

 DL 2023: 36th International Workshop on Description Logics, September 2–4, 2023, Rhodes, Greece

 <https://nofar.carme.li/> (N. Carmeli)

 0000-0003-0673-5510 (N. Carmeli)

 © 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

$$\begin{aligned}
Q_1(\text{student, prof, class}) &\leftarrow \text{Registration}(\text{student, class}), \text{Teaching}(\text{class, prof}), \text{Conflicts}(\text{student, prof}) \\
Q_2(\text{actor, year}) &\leftarrow \text{Cast}(\text{movie, actor}), \text{Release}(\text{movie, year}) \\
Q_3(p_1, p_2, \text{room, grade}) &\leftarrow \text{Seating}(p_1, \text{room}), \text{Seating}(p_2, \text{room}), \text{Grade}(p_1, \text{grade}), \text{Grade}(p_2, \text{grade}) \\
Q_4(\text{post, } p_1, p_2) &\leftarrow \text{Posts}(\text{post, } p_1), \text{Follows}(p_1, p_2) \\
Q_5(\text{post, } p_1, p_2) &\leftarrow \text{Posts}(\text{post, } p_0), \text{Follows}(p_0, p_1), \text{Friends}(p_1, p_2)
\end{aligned}$$

Figure 1: Query Examples.

require that the free variables are connected in a non-standard way. The hardness side of this dichotomy assumes the hardness of Boolean matrix multiplication and hyperclique detection in hypergraphs. For more details about this dichotomy, see [3]. Consider for example the first two queries defined in Figure 1. These two queries are classified as hard according to Theorem 1 as Q_1 is cyclic and Q_2 is acyclic but not free-connex. If the variable year was removed from the head of Q_2 , this query would become acyclic free-connex and classified as easy.

2. Additional Tractable Cases

Can we find CQs that can be answered with ideal guarantees even though they are not free-connex acyclic? As a first impression, we might think that the answer is 'no' due to the negative side of Theorem 1. However, we will see that such cases exist.

As a first example, consider Q_3 . This query is cyclic, but it can be answered with linear preprocessing and constant delay due to the self-joins it contains. Theorem 1 only applies to CQs in which every atom refers to a different relation. If this is not the case, we say that the query contains self-joins, and we can sometimes use these to design more efficient algorithms [4]. The complete classification of CQs with self-joins is not yet known.

Next, let us consider Q_2 again. This query asks for a list of actors and the years in which they were active. If we just consider the structure of the query and assume the input database can be general, this query is classified as hard according to Theorem 1 as it is not free-connex. However, considering the semantics of this query, we can assume that every movie has one release year. This information can be modeled using the functional dependency $\text{Release} : \text{movie} \rightarrow \text{year}$. If we can assume that the input database conforms to this constraint, the problem becomes easier, and it can now be solved with ideal guarantees [5]. The complete classification of CQs with general functional dependencies is not yet known. We do have a classification for unary functional dependencies, where one variable implies another, for CQs without self-joins. This classification is done by extending the query according to the dependencies (in the case of this example, adding the variable year in all places that contain the variable movie), and checking whether the extended query is acyclic free-connex.

Finally, assume the CQ is used as part of a larger query, and consider the union of CQs $Q_4 \cup Q_5$. Even though Q_5 is not free-connex, the entire union can be answered with ideal guarantees since we can intertwine the computation of Q_5 with that of Q_4 . In fact, even a union

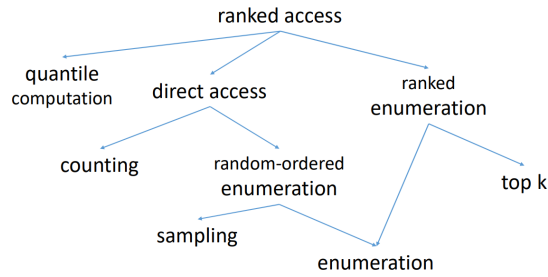


Figure 2: Query answering tasks.

that contains only non-free-connex CQs can become tractable in a similar way. The complete classification of unions of CQs is not yet known, but we have a classification that applies for some subclasses [6, 7].

3. Doing Even Better in the Tractable Cases

Sometimes the final goal of the user is not just to produce all answers to a CQ, but they can be interested in different query answering tasks such as computing the best answer or a median answer according to some ranking function. In other cases, the user may want to compute some statistics over the answers and be willing to compromise for the accuracy of these statistics in favor of speed. In such a case, a random sample of the answers can be useful. Figure 2 shows a summary of some query-answering tasks and the relation of the implication of efficient algorithms between tasks.

The previous section discusses enumeration without order requirements. Clearly, ranked enumeration and random-ordered enumeration are stricter requirements. However, ranked enumeration allows efficiently computing the best answer or in general the top k answers for any k . Random-ordered enumeration is in other words sampling without repetitions, and so this is stronger than the task of sampling with repetitions. Direct access is the task of simulating an array containing the query answers in a way that allows random access to answers in arbitrary indices. It also requires outputting an error message if the index requested is too large, and this can be used together with binary search to determine the number of answers (a task called counting) with only a logarithmic number of access calls. Direct access can also be used to achieve random-ordered enumeration by generating a random permutation of the indices and accessing the corresponding answers on the fly [8]. If the simulated array is required to be ordered, we call this task ranked access, and it can be used to directly access the middle answer for computing the median or to similarly compute any other quantile.

Assume we are interested in ordering the query answers in a lexicographic order. That is, we order the answers by the assignment to one variable, then break ties according to another variable, and so on. The free-connex acyclic CQs, which we know have extremely efficient enumeration algorithms, also have efficient algorithms for quantile computation, direct access and ranked enumeration [9, 10], with linear preprocessing and logarithmic time per outputted answer. Regarding ranked enumeration, this depends on the specific lexicographic order [9].

References

- [1] J. Brault-Baron, De la pertinence de l'énumération: complexité en logiques propositionnelle et du premier ordre, Ph.D. thesis, Université de Caen, 2013.
- [2] G. Bagan, A. Durand, E. Grandjean, On acyclic conjunctive queries and constant delay enumeration, in: *International Workshop on Computer Science Logic*, Springer, 2007, pp. 208–222.
- [3] C. Berkholz, F. Gerhardt, N. Schweikardt, Constant delay enumeration for conjunctive queries: a tutorial, *ACM SIGLOG News* 7 (2020) 4–33.
- [4] N. Carmeli, L. Segoufin, Conjunctive queries with self-joins, towards a fine-grained complexity analysis, in: *PODS'23*, 2023.
- [5] N. Carmeli, M. Kröll, Enumeration complexity of conjunctive queries with functional dependencies, *Theory of Computing Systems* 64 (2020) 828–860.
- [6] N. Carmeli, M. Kröll, On the enumeration complexity of unions of conjunctive queries, *ACM Transactions on Database Systems (TODS)* 46 (2021) 1–41.
- [7] K. Bringmann, N. Carmeli, Unbalanced triangle detection and enumeration hardness for unions of conjunctive queries, *arXiv preprint arXiv:2210.11996* (2022).
- [8] N. Carmeli, S. Zeevi, C. Berkholz, A. Conte, B. Kimelfeld, N. Schweikardt, Answering (unions of) conjunctive queries using random access and random-order enumeration, *ACM Transactions on Database Systems (TODS)* 47 (2022) 1–49.
- [9] N. Carmeli, N. Tziavelis, W. Gatterbauer, B. Kimelfeld, M. Riedewald, Tractable orders for direct access to ranked answers of conjunctive queries, *ACM Transactions on Database Systems* 48 (2023) 1–45.
- [10] N. Tziavelis, W. Gatterbauer, M. Riedewald, Beyond equi-joins: Ranking, enumeration and factorization, in: *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, volume 14, NIH Public Access, 2021, p. 2599.