# Fine-Grained Complexity of Query Answering

M2 Internship Proposal For 2024

Nofar Carmeli

This research is in the field of database theory, with intersections with fine-grained complexity and enumeration algorithms. The long-term goal of this line of work is, given a user-specified query, to determine what is the most efficient algorithm that can evaluate this query. I would be happy to offer an M2 internship. In particular, I can propose the following two options, according to the student's preferences.

## 1 Version 1: Pure Theory

In terms of time complexity, as the number of answers to the query may be much larger than the size of the input database, we cannot expect to have a linear-time algorithm for query evaluation (only printing the answers may require longer), so it is very natural to consider enumeration measures. In this sense, the ideal time guarantees are linear preprocessing and constant delay between successive answers. As an intermediate goal, we ask which queries can be solved within this time complexity. The goal is to identify a property that distinguishes the tractable queries from the intractable ones. For queries that are a *natural join*, we understand the complete picture, and the tractability depends on the structure of the hypergraph describing the query [3, 1].

As an example, consider the following query and database. The query asks for a list of people who have a hobby that they can practice in the city where they live. You can also see the query answers and the hypergraph describing the query.

Q(person,activity,city,age):-Hobbies(person,activity),Residence(person,city),Availability(city,activity),Age(person,age)

Hobbies

| person | activity |
|---|---|
| Léa | climbing |
| Léa | hiking |
| Thomas | theatre |

Residence

| person | city |
|---|---|
| Léa | Montpellier |
| Thomas | London |

Availability

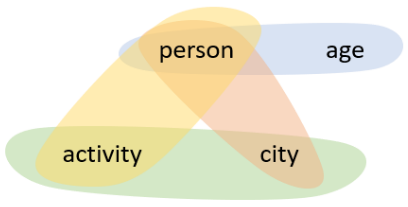
| city | activity |
|---|---|
| Montpellier | climbing |
| Montpellier | swimming |
| London | theatre |

Age

| person | age |
|---|---|
| Léa | 22 |
| Thomas | 24 |

Query Answers

| person | age | activity | city |
|---|---|---|---|
| Léa | 22 | climbing | Montpellier |
| Thomas | 24 | theatre | London |

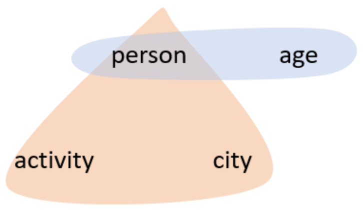


As the hypergraph describing this query is cyclic, we know that there is no algorithm that can produce the answers to this query with linear preprocessing and constant delay, under some complexity assumptions. If the atom Residence(person,city) was removed from the query, it would have been acyclic and solvable efficiently.

In general, when we consider more expressive query classes, the classification is well-understood only in some cases [9, 8, 4, 7], and we can explore during the internship some of the other cases. This type of research usually includes coming up with algorithms, identifying the class of queries for which the algorithms can be applied, coming up with reductions for proving hardness, identifying the class of queries for which the reductions can be applied, identifying queries that are not covered by the algorithms and reductions, trying to extend the algorithms or reductions to cover these queries too, and repeating until we reach a complete classification.

## 2 Version 2: Including an Implementation

When the query is not naturally of the tractable form described in the previous section, how can we still evaluate it? We can resolve the problematic parts of the query during preprocessing using a less efficient algorithm. Since this algorithm is less efficient, we want to make sure that we do not use it to evaluate a larger part of the query than necessary. One way of determining which parts need to be resolved is by using *tree decompositions*. Here is an example of a tree-decomposition for the cyclic query from the previous section. This decomposition means that we should join Hobbies, Residence and Availability during preprocessing, but there is no need to do the same for Age.



As finding the best decomposition is NP-hard in general, practically it sometimes makes sense to compute all relevant decompositions and choose the best we see after a reasonable time. It is worth mentioning that tree decompositions are useful in many domains in addition to database theory. There are several known enumeration algorithms for tree-decompositions [5, 6], some of which have not yet been implemented. These algorithms have generic components, and in some cases, it is not clear what is the best precise complexity these algorithms can achieve using the best data structures. In addition, for more expressive queries, we will need to extend these standard decompositions into *free-connex decompositions* to account for projections in the query [2].

I suggest an internship that investigates the complexity of finding tree decompositions and free-connex decompositions, and then implements a tool that finds such decompositions for a given query.

If you find these topics interesting, I recommend watching one of the recorded talks on my website to get a better sense of the area.

## References

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

[2] Christoph Berkholz, Fabian Gerhardt, and Nicole Schweikardt. Constant delay enumeration for conjunctive queries: a tutorial. *ACM SIGLOG News*, 7(1):4–33, 2020.

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

[4] Karl Bringmann and Nofar Carmeli. Unbalanced triangle detection and enumeration hardness for unions of conjunctive queries. *arXiv preprint arXiv:2210.11996*, 2022.

[5] Caroline Brosse, Vincent Limouzy, and Arnaud Mary. Polynomial delay algorithm for minimal chordal completions. *arXiv preprint arXiv:2107.05972*, 2021.

[6] Nofar Carmeli, Batya Kenig, and Benny Kimelfeld. Efficiently enumerating minimal triangulations. In *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 273–287, 2017.

[7] Nofar Carmeli and Markus Kröll. Enumeration complexity of conjunctive queries with functional dependencies. *Theory of Computing Systems*, 64(5):828–860, 2020.

[8] Nofar Carmeli and Markus Kröll. On the enumeration complexity of unions of conjunctive queries. *ACM Transactions on Database Systems (TODS)*, 46(2):1–41, 2021.

[9] Nofar Carmeli and Luc Segoufin. Conjunctive queries with self-joins, towards a fine-grained enumeration complexity analysis. In *Proceedings of the 42nd ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 277–289, 2023.