PhD Proposal: Efficient enumeration via edits

Antoine Amarilli, Mikaël Monet

The field of *enumeration algorithms* [Was16] studies how to efficiently compute the results of problems that have a large number of solutions: for instance, list all source-to-target paths in a graph, list all words of a regular language, list all satisfying assignments for a propositional formula, all answers to a database query, etc. The focus on enumeration algorithms is to ensure a small *delay* between two successive solutions. In many contexts, enumeration algorithms have been produced which ensure the best possible delay, in the form of a *constant delay* [Seg15] bound if the results are assumed to have constant size, or *output-linear* delay [Bag06] when this assumption is not made.

However, when the results to be produced have non-constant size and are large, then even output-linear delay may be too much. However, it looks impossible to achieve a better complexity, because output-linear complexity appears to be necessary just to write down the results. The goal of this PhD topic is to explore a general idea of *enumeration via edits*, and study its applications to specific use cases and connections to neighboring areas. The proposal follows a first article by the PhD supervisors [AM23] which showcased the methodology on a specific task in formal languages.

The idea of enumeration via edits is to avoid paying for the cost of printing solutions by a simple change in the definition of the model: produce each solution by *editing* the previous solution. For instance, when listing paths in a graph, produce each path by editing the previous path; when enumerating words in formal language theory, apply edits on each word to get to the next word. More precisely, the enumeration model can apply changes or issue an "output" instruction which instantaneously produces the current state of the memory as a solution, for free. The challenge then becomes to produce the successive solutions by applying a constant number of edits between any two consecutive solutions, and by computing these edits as efficiently as possible, even while the size of the produced solutions becomes large. Enumeration via edits is similar to the principle in combinatorial enumeration known as "Do not count the output" [Rus03, p8], but it is made more realistic by imposing that each output is produced from the previous output by applying just a small number of changes.

The PhD will study how this approach can apply to problems such as:

- *Enumerating solutions of regular path queries in graphs.* In the setting of graph databases, *regular path queries* (RPQ) are a theoretical language that can be used to look for paths that satisfy a regular language constraint. Formally, fixing an alphabet Σ, a graph database is just a directed graph G with edges labeled with letters of Σ, an RPQ is specified as a regular expression e over Σ, and we are interested in efficiently listing all paths of G that form a word in e. The task can be studied with several semantics, in particular looking for walks, simple paths, or paths by increasing order of length. One first research direction is to investigate for which regular languages and which semantics it is possible to efficiently produce all matching paths with the enumeration via edits methodology.

- *Regular languages.* A natural enumeration task in formal languages is to produce, given a description of a language L, the sequence of words of L, possibly in a specific order. This task is the one studied in [AM23], but many questions remain open after this first foray in the area.
- Other settings, such as the enumeration of satisfying assignments for propositional formulas or *Boolean circuits*, in particular those obeying the restrictions studied in the area of *knowledge compilation* [DM02]; the enumeration of the answers to *database queries*, possibly in specific orders; etc.

More broadly, the PhD can look at connections to the areas of:

- *Factorized representations* [OZ15], because enumeration via edits amounts to identifying common parts of solutions and so may be related to the computation of a factorized set of results.
- *Incremental maintenance* [AJP21], which studies how to efficiently maintain some properties of data (e.g., connectedness of a graph, membership of a word to a language) while the data can be updated by edit operations.

Supervision and environment. This PhD will take place in the LINKS team of the Inria center at University of Lille, in the North of France. The LINKS team focuses on logics, algorithms, formal language theory, and database theory, and offers a dynamic environment for research on these topics. The PhD will be co-supervised by Antoine Amarilli[1] (Advanced Research Position at Inria) and Mikaël Monet[2] (Chargé de recherche Inria).

Applications should be sent by email to: `a3nm@a3nm.net` and `mikael.monet@inria.fr`. The deadline to apply is April 30, 2025.

References

[AJP21] Antoine Amarilli, Louis Jachiet, and Charles Paperman. Dynamic membership for regular languages. In *ICALP*, 2021.

[AM23] Antoine Amarilli and Mikaël Monet. Enumerating regular languages with bounded delay. In *STACS*, 2023.

[Bag06] Guillaume Bagan. MSO queries on tree decomposable structures are computable with linear delay. In *CSL*, 2006.

[DM02] Adnan Darwiche and Pierre Marquis. A knowledge compilation map. *JAIR*, 17, 2002.

[OZ15] Dan Olteanu and Jakub Závodnỳ. Size bounds for factorised representations of query results. *TODS*, 40(1), 2015.

[Rus03] Frank Ruskey. Combinatorial generation. Preliminary working draft, 2003.

[Seg15] Luc Segoufin. Constant delay enumeration for conjunctive queries. *ACM SIGMOD Record*, 44(1), 2015.

[Was16] Kunihiro Wasa. Enumeration of enumeration algorithms, 2016.

[1] `https://a3nm.net/`
[2] `https://mikael-monet.net/`