Enumeration Classes Defined by Circuits

Nadia Creignou, Arnaud Durand, Heribert Vollmer

NormaSTIC 2023

Enumeration ?

What are the dance schools in Metz or Nancy?

$\boxed{20}$

- Académie Arz-Klehr
- Art k Danse
- Les Feux de la Rampe
- DancerShow57
- Nickel-School
- L'Estudio
- Aude Ecole de Danse
- La Fabrique
- L'Espace Lafayette
- Acadanse

⋮

What ingredients are needed to cook a tiramisu?

$\boxed{6}$

- mascarpone cream
- sugar
- eggs
- coffee
- cocoa
- lady finger cookies

What are the plant species on earth?

$\boxed{298\ 000}$

- Adonis pyrenaica
- Androsace alpina
- Lavatera maritima
- Matthiola tricuspidata

⋮

What are the maximal independent sets in a graph with n vertices?

$\boxed{3^{n/3}}$

Introduction

- Topic: algorithms and complexity for enumeration problems.
- Enumeration problem: generate all solutions of a problem one by one and without repetition.

> **Examples:** generate all models of a propositional formula, all triangles in a graph, etc

- Subject has deserved a lot of attention in graphs algorithms and combinatorics but also in data management.

Input sensitive versus output sensitive complexity

- Input-sensitive approach: Measure of the time complexity of an enumeration algorithm in the **size of the input**
- Output-sensitive approach: Measure of the time complexity of an enumeration algorithm in the **size of the input and the ouput**.

How to measure the complexity of such problems?

Focus is put on the dynamic of the generation process: the delay

For combinatorial problems, three main types of tractability:

- OutputP: polynomial time in the size of $|x|$ and the solutions set.
- IncP: computing the $(i+1)$th solution is polynomial in $|x|$ and i
- DelayP: computing all solutions with a delay polynomial in $|x|$ between two solutions. First attempt to capture *efficient* enumeration.

Very few lower bounds. All under complexity hypothesis.

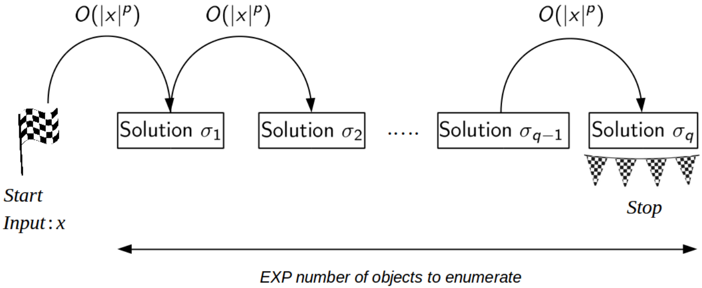
Very few lower bounds

How to prove that an enumeration problem is not in DelayP?

- Show hardness of some related decision problem: For instance one cannot enumerate the models of any Boolean formula in DelayP, unless $\mathbf{P} = \mathbf{NP}$.
- Show that it is as hard as enumerating the minimal transversals of a hypergraph.

Capturing efficient enumeration

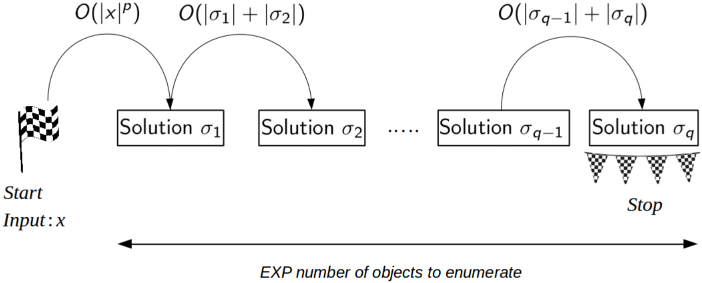
- Polynomial delay: An algorithm runs with polynomial delay if the pre-processing step and the delay between two consecutive solutions are polynomially bounded in the size of the input.



Classical algorithm : flashlight binary search for satisfiability problems

Capturing efficient enumeration

- Linear delay: An algorithm runs with linear delay if the pre-processing is polynomially bounded in the size of the input and the delay between two consecutive solutions σ_i and σ_{i+1} is linearly bounded in $(|\sigma_i| + |\sigma_{i+1}|)$.



Classical algorithm : enumeration of S-T paths in a DAG

Towards more efficiency

In the context of data management and query answering:

- The measures above, even linear delay, are not considered as really tractable.
- Consider $\mathbf{CD}\circ\mathbf{lin}$: problems that can be enumerated on RAMs with constant delay after linear time preprocessing (Durand and Grandjean'07)
- Contains a lot of natural query problems
- Conditional lower bounds have been proved (e.g. acyclic conjunctive queries based on hypothesis on the complexity of Boolean Matrix Computation (BMM)).

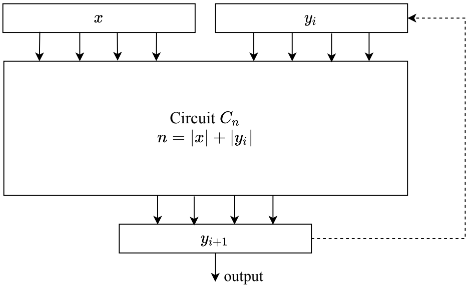
In this work

- Focus on enumeration below DelayP
- Propose enumeration algorithms based on "small" Boolean circuits
- Low classes but of a different nature than imposing constant delay
- Contributions and objectives:
  - Propose a hierarchy of small enumeration classes
  - Show they are populated by natural problems
  - Prove lower bounds! some of them non conditionally.

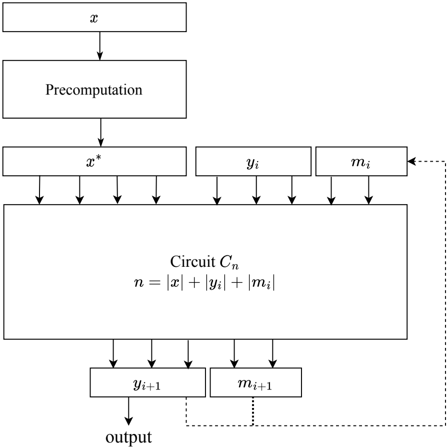
Circuit enumeration - simplistic view

- Starting point: a family of circuits (C_n) of a given kind.
- x is the main input
- successive outputs $y_1, y_2, ..., y_i, ...$ serve as auxiliary input

Compute next solution from the previous one by a circuit



Circuit enumeration: using precomputation and memory



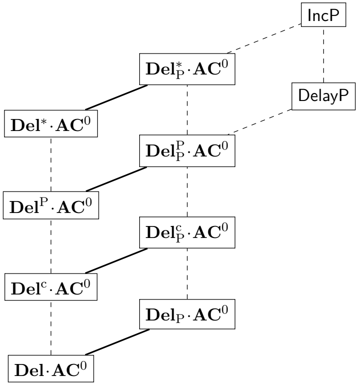
A hierarchy of classes

$\mathbf{AC}^0$ uniform circuit families of polynomial size and constant depth, using unbounded fan-in AND/OR gates plus negation gates.

A catalog of classes: $\mathbf{Del}^{\text{memory}}_{\text{precomp}} \cdot \mathbf{AC}^0$

- $\mathbf{Del} \cdot \mathbf{AC}^0$: no precomputation, no memory
- $\mathbf{Del}^{\mathrm{c}} \cdot \mathbf{AC}^0$: no precomp, constant size memory
- $\mathbf{Del}^{\mathrm{P}} \cdot \mathbf{AC}^0$: no precomp, polynomial size memory
- $\mathbf{Del}_{\mathbf{P}} \cdot \mathbf{AC}^0$: polytime precomp, no memory
- $\mathbf{Del}^{\mathrm{c}}_{\mathbf{P}} \cdot \mathbf{AC}^0$: polytime precomp, constant size memory
- $\mathbf{Del}^{\mathrm{P}}_{\mathbf{P}} \cdot \mathbf{AC}^0$: polytime precomp, polynomial size memory
- Unbounded memory: $\mathbf{Del}^{*} \cdot \mathbf{AC}^0$, $\mathbf{Del}^{*}_{\mathbf{P}} \cdot \mathbf{AC}^0$

A first view of the hierarchy



- Inclusions between classes
- Bold lines denote (obvious) strict inclusions

Comparison with constant delay

- $\mathbf{CD}\circ\mathbf{lin}$: problems that can be enumerated on RAMs with constant delay after linear time preprocessing
- Classes $\mathbf{Del}\cdot\mathbf{AC}^0$ and $\mathbf{CD}\circ\mathbf{lin}$ are incomparable
  - Output the parity of the number of 1 is in $\mathbf{CD}\circ\mathbf{lin}$ (and not in $\mathbf{Del}\cdot\mathbf{AC}^0$)
  - Enumerate the 1 entries of $A \times B$ with A, B Boolean matrices is in $\mathbf{Del}\cdot\mathbf{AC}^0$
- $\mathbf{CD}\circ\mathbf{lin} \subsetneq \mathbf{Del}^{\mathrm{P}}_{lin}\cdot\mathbf{AC}^0$.

An example: Gray codes

Problem: Enumerate $\{0,1\}$-words of a given length n by changing one bit between two consecutive outputs.

> One method: Binary reflected Gray code G_n.
> The 2^n words w and their range r for $n = 4$.

r :	w	r :	w
0 :	0000	8 :	1100
1 :	0001	9 :	1101
2 :	0011	10 :	1111
3 :	0010	11 :	1110
4 :	0110	12 :	1010
5 :	0111	13 :	1011
6 :	0101	14 :	1001
7 :	0100	15 :	1000

Gray codes enumeration

- Given n and $r < 2^n$
- Let $r = b_{n-1} \cdots b_1 b_0$ (in binary)
- Let $G_r^n = a_{n-1} \cdots a_1 a_0 \in \Sigma^n$ be the rth word.

Well known that, for all $j = 0, ..., n-1$,

$$b_j = \sum_{i=j}^{n-1} a_i \bmod 2 \text{ and } a_j = (b_j + b_{j+1}) \bmod 2.$$

Given a word, computing its rank is computing parity.

Result: Given 1^n, enumerating all words of length n in a Gray code order is in $\mathbf{Del}^{\mathrm{c}}{\cdot}\mathbf{AC}^0 \backslash \mathbf{Del}_{\mathbf{P}}{\cdot}\mathbf{AC}^0$

Gray codes enumeration

In $\mathbf{Del^c}\cdot\mathbf{AC}^0$. Classical method:

- Step 0 : output the word $0\cdots0$ of length n.
- Step $2k+1$: switch the bit at position 0.
- Step $2k+2$: find minimal position i where there is a 1 and switch bit at position $i+1$.

$3:\quad 0010$
$4:\quad 0110$
$5:\quad 0111$
$6:\quad 0101$

Not in $\mathbf{Del_P}\cdot\mathbf{AC}^0$. Suppose it is:

- Consider arbitrary $w = w_{n-1}\ldots w_0$
- There exists $r < 2^n$ s.t. $G_r^n = w$. Let $w' = G_{r+1}^n$
- Compare w and w': decide which step has been applied
- Hence decide is r is odd or even and the parity of the number of 1s in w.

Contradict classical result that $\textsc{Parity}$ is not in $\mathbf{AC}^0$.

Satisfiability problems

Question: Is $\mathbf{AC}^0$ powerful enough to serve as a main enumerator engine for interesting SAT fragments?

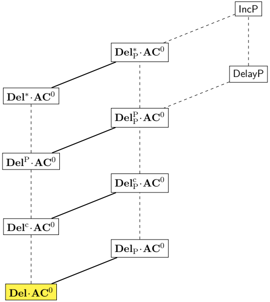
ENUM-SAT

Input:	A set of clauses Γ over a set of variables V
Output:	an enumeration of all assignments that satisfy Γ

- ENUM-MONOTONE-SAT : positive (resp. negative) clauses
- ENUM-KROM-SAT : clauses of length at most 2
- ENUM-XOR-SAT : clauses with xor disjunctions
- ENUM-HORN-SAT : clauses with at most one positive literal

Enumerating SAT using AC^0 circuits

$\textsc{Enum-Monotone-Sat} \in \mathbf{Del}{\cdot}\mathbf{AC}^0$



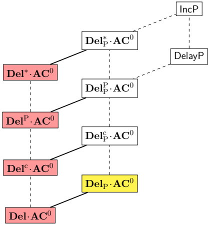
Enumerating SAT using AC^0 circuits

$\textsc{Enum-Krom-Sat} \in \mathbf{Del_P} \cdot \mathbf{AC}^0 \setminus \mathbf{Del}^* \cdot \mathbf{AC}^0$.

Lower bound: reduction from $\textsc{st-connectivity}$.

Upper bound:

- no memory is needed
- Do not use the flashlight algorithm
- Build on the Apsvall, Plass, Tarjan 79's algorithm for 2-$\textsc{Sat}$ as a preprocessing.

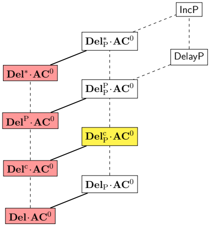


Enumerating SAT using AC^0 circuits

$\textsc{Enum-XOR-Sat} \in \mathbf{Del}^{c}_{\mathbf{P}}\cdot\mathbf{AC}^0 \setminus \mathbf{Del}^{*}\cdot\mathbf{AC}^0$.

Upper bound: Gaussian elimination + Gray code enumeration.

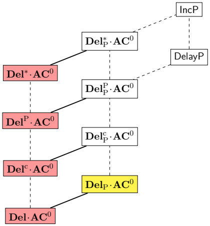
Lower bound: Express $\textsc{Parity}$ with XOR-constraints.



Enumerating SAT using AC^0 circuits

New: in fact, $\textsc{Enum-XOR-Sat} \in \mathbf{Del_P}\cdot\mathbf{AC}^0$

$\textsc{Enum-XOR-Sat} \in \mathbf{Del_P}\cdot\mathbf{AC}^0 \setminus \mathbf{Del}^*\cdot\mathbf{AC}^0$.

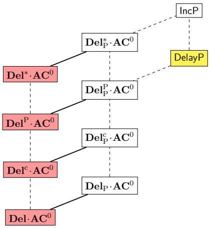


Enumerating SAT using AC^0 circuits

Where is Enum-Horn-Sat in this hierarchy of circuits?

Open: Known to be in DelayP, but $\mathbf{AC}^0$ might not be powerful enough even with memory and precomputation.

New: Enum-Horn-Sat is $\mathbf{Del}{\cdot}\mathbf{P}$-complete via parsimonious like reductions.



Separating classes without precomputation

Proposition: $\mathbf{Del}\cdot\mathbf{AC}^0 \subsetneq \mathbf{Del}^{\mathrm{c}}\cdot\mathbf{AC}^0 \subsetneq \mathbf{Del}^{\mathrm{P}}\cdot\mathbf{AC}^0 \subsetneq$ DelayP.

- All results are unconditional
- Each proof exhibits a concrete problem in the upper class which is not in the lower one.
- They build on existing lower bounds, mainly on the fact that $\textsc{Parity}$ is not in $\mathbf{AC}^0$ (even non-uniform).

Focus on $\mathbf{Del}{\cdot}\mathbf{AC}^0 \subsetneq \mathbf{Del}^{\mathrm{c}}{\cdot}\mathbf{AC}^0$

Let $x \in \{0,1\}^*$, $x = x_1 \dots x_n$ and $m = \lceil \log n \rceil + 1$. Consider $R_L = A \cup B$ with:

- $A = \{\, y \in \{0,1\}^* \,||y| = m, y \neq 0^m, y \neq 1^m \,\}$
- $B = \{\, 1^m \,\}$ if x has an even number of ones, else $B = \{\, 0^m \,\}$.

Why is R_L in $\mathbf{Del}^{\mathrm{c}}{\cdot}\mathbf{AC}^0$? Simply because $2^m - 2 \approx n$ dummy solutions to enumerate (A) let enough "time" to know if x has an even number of 1 by patiently transmitting one bit of memory from one step to the other...

Focus $\mathbf{Del}\cdot\mathbf{AC}^0 \subsetneq \mathbf{Del}^{\mathrm{c}}\cdot\mathbf{AC}^0$

Let $x \in \{0,1\}^*$, $x = x_1 \dots x_n$ and $m = \lceil \log n \rceil + 1$. Consider $R_L = A \cup B$ with:

- $A = \{\, y \in \{0,1\}^* \mid |y| = m, y \neq 0^m, y \neq 1^m \}$
- $B = \{\, 1^m \}$ if x has an even number of ones, else $B = \{\, 0^m \}$

Why is R_L not in $\mathbf{Del}\cdot\mathbf{AC}^0$?
Suppose it is and let (C_n) be the family doing it. Let $z_1, \dots, z_t$ be an enumeration of A. We construct a circuit family as follows:

- compute in parallel all $C_{|\cdot|}(x)$ and $C_{|\cdot|}(x, z_i)$ for $1 \leq i \leq t$
- check which of 0^m or 1^m appear

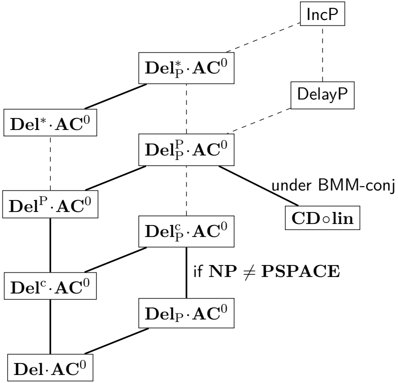
One conditional result

Proposition:
$\mathbf{NP} \neq \mathbf{PSPACE}$ implies $\mathbf{Del}^c{\cdot}\mathbf{AC}^0 \setminus \mathbf{Del}_{\mathbf{P}}{\cdot}\mathbf{AC}^0 \neq \emptyset$.

(Indirect) consequence of a result by Hertrampf, Lautemann, Schwentick, Vollmer and Wagner (93) which shows that $\mathbf{PSPACE}$ is serializable in $\mathbf{AC}^0$.

Serializable : Computations in $\mathbf{PSPACE}$ can be cut into an exponentially long sequence of $\mathbf{AC}^0$ computations that pass a constant number of bits to the next one.

Summary of separation results



- Inclusions between classes
- Bold lines denote strict inclusions
- BMM-conj: Boolean Matrix Multiplication can not be done in $O(m)$, where m is the number of non-zero-entries of the two matrices

Conclusion and open problems

- We introduced enumeration algorithms based on circuits
- Definitions are modular (one can vary the kind of circuits, the memory usage and precomputation)
- Even small classes of circuits lead to powerful enumeration engine (inside DelayP)
- Some lower bounds can be proven using two ingredients
  - Classical lower bounds on circuit
  - Enumeration algorithms are forced to output regularly
- Lot's of open problems and room to extend methods to find "natural" lower bounds for well-known algorithmic problems