Enumeration Complexity of logical query problems with second order variables

Arnaud Durand[1] and Yann Strozecki[2]

1,2 IMJ, CNRS UMR 7586
Université Paris Diderot, France
durand@logique.jussieu.fr, strozecki@logique.jussieu.fr

Abstract

We consider query problems defined by first order formulas of the form $\Phi(\mathbf{x}, \mathbf{T})$ with free first order and second order variables and study the data complexity of enumerating results of such queries. By considering the number of alternations in the quantifier prefixes of formulas, we show that such query problems either admit a constant delay or a polynomial delay enumeration algorithm or are hard to enumerate. We also exhibit syntactically defined fragments inside the hard cases that still admit good enumeration algorithms and discuss the case of some restricted classes of database structures as inputs.

Keywords and phrases Descriptive Complexity, Enumeration, Query Problem

Digital Object Identifier 10.4230/LIPIcs.xxx.yyy.p

Introduction

Query answering for logical formalisms is a fundamental problem in database theory. There are two natural ways to consider the answering process and consequently to evaluate the complexity of such problems. Given a query φ on a database structure $\mathcal{S}$, one may consider computing the result $\varphi(\mathcal{S})$ as a global process and measure its data complexity in terms of the database and the output sizes. Alternatively, one can see this task as a dynamical process in which one computes the tuples of the solution set one after the other. In this case, the main measure is the delay spent between two successive output tuples. In recent years, this approach has deserved some attention in the context of logical query problems: see, for example, [3] for a study on conjunctive queries, or [4, 1] for monadic second order lorgic on bounded tree-width structures or [6, 11] for first order queries on structures of bounded degree. However, having only free first order variables in formulas is not enough to capture complex objects of non constant size. This is the case when one wants to obtain, for example, cliques or hypergraph transversals of arbitrary size (see Example 2) or classical NP properties.

It is known since Fagin's theorem [7] that NP corresponds exactly to problems definable in existential second order logic. That is, the language L is in NP, if and only if there exists an existential second order formula $\Phi(\mathbf{T})$ over a signature $\sigma \cup \{\mathbf{T}\}$, such that, for all σ-structure $\mathcal{S}$:

$$\mathcal{S} \in L \iff \mathcal{S} \models \exists \mathbf{T}\ \Phi(\mathbf{T}).$$

In this paper, we consider first order query with possibly free second order variables and study their enumeration complexity. Since in full generality such formulas may be very

Conference title on which this volume is based on.
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

expressive, we consider fragments defined by the quantifier alternation of formulas[1]. The hardness of counting the number of solutions of a problem may sometimes be seen as a first approach of enumeration complexity. In [12], a descriptive complexity point of view of counting problems is proposed. They show that for the $\mathbf{\Sigma}_0$ (quantifier free) fragment, counting the number of solutions can be done in polynomial time and that already at the first universal level $\mathbf{\Pi}_1$ (only one block of universal quantifiers) ♯P-complete problems can be defined. They also show that the $\mathbf{\Sigma}_1$ level (one block of existential quantifiers) and some other syntactically defined fragment admits a fully polynomial randomized approximation scheme to count the number of solutions. In this paper, we show that the situation for enumeration is more complex. Our contributions are as follows.

- For any fixed formula $\Phi(\mathbf{x}, \mathbf{T}) \in \mathbf{\Sigma}_0$, there exists an algorithm that, given a structure $\mathcal{S}$, enumerates $\Phi(\mathcal{S})$ with polynomial time precomputation and constant delay. Under a parameterized complexity assumption, the degree of the polynomial in the precomputation step depends on the formula size. To show constant delay enumeration we prove that one can pass from one solution (of the form $(\mathbf{x}, \mathbf{T})$) to another by only a constant number of local changes.
- We also prove that, for any k, if the structure $\mathcal{S}$ is of degree bounded by k, then there is an enumeration algorithm for the $\mathbf{\Sigma}_0$ fragment with linear precomputation and constant delay.
- For any fixed formula $\Phi(\mathbf{x}, \mathbf{T}) \in \mathbf{\Sigma}_1$, there exists an algorithm that, given a structure $\mathcal{S}$, enumerates $\Phi(\mathcal{S})$ with polynomial time precomputation and polynomial time delay. To this aim, we study the closure under union problem in the context of enumeration and prove that some closure result holds even for the union of two problems which are efficiently enumerable but relatively to different orderings (of their respective solution space).
- The class $\mathbf{\Pi}_1$ already contains problems that are hard to enumerate and $\mathbf{\Pi}_2$ is enough to capture all **FO** definable problems on ordered structures up to parsimonious reductions.
- Finally, we exhibit natural fragments above $\mathbf{\Pi}_1$ that admit efficient enumeration procedures.

Basic definitions about logical query problems, enumeration problems and main enumeration complexity measures are given in Section 1. Results about the enumeration complexity of $\mathbf{\Sigma}_0$ query problems are given in Section 2 and about the $\mathbf{\Sigma}_1$ query problems in Section 3. In this latter section, we also discuss the relationship with the enumeration of models of propositional formulas in disjunctive normal form and shows that the two problems are intimately related. The $\mathbf{\Pi}_1$ fragment is studied in Section 4 where both the hardness results are given and some tractable fragments are exhibited.

1 Preliminaries

Enumeration problem and complexity

Let $\mathcal{I}, \mathcal{O}$ be two sets and R be a polynomially balanced binary predicate $R \subseteq \mathcal{I} \times \mathcal{O}$ decidable in polynomial time. In particular, given $x \in \mathcal{I}$ and $y \in \mathcal{O}$, checking whether $R(x, y)$ can be

[1] Note that this is the approach to define the classes of the **W** hierarchy in parameterized complexity

done in time polynomial in $|x|$. One defines the enumeration function associated to R as follows.

ENUM·R
Input: $x \in \mathcal{I}$
Output: an enumeration of elements in $R(x) = \{y : R(x, y)\}$

In this paper, we consider the *random access machine* model (RAM) with addition and subtraction as its basic arithmetic operations. It has read-only *input registers* $I_1, I_2, \ldots$ (containing the input x), read-write *work registers* $R_1, R_2, \ldots$ and *output registers* $O_1, O_2, \ldots$. Our model is equipped with an additional instruction Output which, when executed, indicates that the non empty output registers contain a partial output $y \in R(x)$. Time complexity is used under the *uniform cost* model. A RAM is of space complexity $O(h(n))$ if, for all inputs of size n, it uses working registers T_i of addresses $i = O(h(n))$ and content $O(\max(n, h(n)))$.

A scheme $\mathcal{A} = (\mathcal{A}_p, \mathcal{A}_e)$ (see [2] for a similar definition) computes the enumeration problem ENUM·R if, for any input x:

- $\mathcal{A}_p$ computes from x an extended input $ext(x)$. This is called the precomputation phase.
- Given $ext(x)$, $\mathcal{A}_e$ computes one after the other and without repetition the elements of $R(x)$ and stops immediately after writing the last one.

We denote by $\text{time}_j(x)$ the moment when $\mathcal{A}$ has completed the writing of the j^{th} solution i.e. after the j^{th} Output instruction is executed (by convention, $\text{time}_0(x) = 0$). Let $\text{delay}_j(x) = \text{time}_j(x) - \text{time}_{j-1}(x)$.

▶ **Definition 1.** Let $g : \mathbb{N} \to \mathbb{N}$, $f : \mathbb{N} \to \mathbb{N}$ be two functions. The problem ENUM·R belongs to the class DELAY(g, f) if there exists an enumeration scheme $\mathcal{A} = (\mathcal{A}_p, \mathcal{A}_e)$ that computes ENUM·R such that, for all input x:

- Precomputation uses time and space $O(g(|x|))$,
- Solutions $y \in R(x)$ are computed successively from $ext(x)$ using delay $O(f(|x|))$ and space $O(\max_{y \in R(x)}(f(|x|), |y|))$

The two enumeration classes below are classical:

$$\text{DELAYP} = \bigcup_{k,h} \text{DELAY}(n^k, n^h), \quad \text{CONSTANT-DELAY} = \bigcup_{k} \text{DELAY}(n^k, O(1)).$$

Logical definitions

We suppose the reader familiar with the basics of finite model theory and first order logic [10]. A signature $\sigma = \{R_1, ..., R_k\}$ is a set of relational symbols (constant symbols will also be authorized). The arity of a predicate R_i is denoted by $\text{ar}(R_i)$. A σ-structure $\mathcal{S} = \langle D, R_1^{\mathcal{S}}, \ldots, R_k^{\mathcal{S}} \rangle$ is composed of a domain D, together with an interpretation $R_i^{\mathcal{S}} \subseteq D^{\text{ar}(R_i)}$ for symbols R_i of σ. When the context is clear, the interpretation $R_i^{\mathcal{S}}$ of R_i is denoted by R_i^*. The size of $\mathcal{S}$ is equal to the cardinality $|D|$ of its domain plus the sum of the number of tuples times the arity for all relations. It is denoted by $|\mathcal{S}|$. If $n \in \mathbb{N}$ such that $|D| = n$ then, D will often be identified with the initial segment of the integers $[n]$.

Let σ be a signature and $\mathbf{T} = (T_1, \ldots, T_h)$ be a tuple of predicate symbols not in σ, let $\mathbf{z} = (z_1, \ldots, z_l)$ be a tuple of variables. We consider first order formulas $\Phi(\mathbf{z}, \mathbf{T})$ with free first order and second order variables. Such formulas, of signature $\sigma \cup \mathbf{T}$ have atomic

formulas (atoms) built over relations of $\sigma \cup \mathbf{T}$ and equality symbol $=$. We denote by $\mathbf{\Sigma}_0$ (or $\mathbf{\Pi}_0$) the set of quantifier free first order formulas. A formula $\Phi(\mathbf{z}, \mathbf{T})$ is in $\mathbf{\Sigma}_{i+1}$ (resp. $\mathbf{\Pi}_{i+1}$), for $i \geq 0$, if it is of the form: $\exists \mathbf{x} \psi$ (resp. $\forall \mathbf{x} \psi$) where ψ is in $\mathbf{\Pi}_i$ (resp. $\mathbf{\Sigma}_i$).

Enumeration Query problems and data complexity

Let $\mathscr{F}$ be a subclass of first order formulas and $\Phi(\mathbf{z}, \mathbf{T}) \in \mathscr{F}$, we consider the following variant of the classical query problem.

ENUM·Φ
Input: A σ-structure $\mathcal{S}$
Output: an enumeration of elements in $\Phi(\mathcal{S}) = \{(\mathbf{z}^*, \mathbf{T}^*) : (S, \mathbf{z}^*, \mathbf{T}^*) \models \Phi(\mathbf{z}, \mathbf{T})\}$

We denote by ENUM·$\mathscr{F}$ the collection of problems ENUM·Φ for $\Phi \in \mathscr{F}$. Note that it can be supposed without loss of generality that the tuple $\mathbf{T}$ contains only one relation T of arity $r = \max_{i \leq h} \mathrm{ar}(T_i) + 1$. To do this, one simply represents each predicate T_i by T and a new constant symbol a_i and replace in formulas each $T_i(\mathbf{x})$ by $T(\mathbf{a}_i, \mathbf{x})$ where the length of $\mathbf{a}_i$ is $r - \mathrm{ar}(T_i)$. It suffices to add the new constants in the signature.

▶ **Example 2.** The formula $IS(T) \equiv \forall x \forall y\ T(x) \wedge T(y) \Rightarrow \neg E(x, y)$ holds if and only if T is an independent set. Remark that the previous formula is in $\mathbf{\Pi}_1$, thus ENUM·IS is in ENUM·Π_1.

▶ **Example 3.** ENUM·HS : given a hypergraph H, enumerate the hitting sets (vertex covers) of H. The hypergraph H is represented by the incidence structure $\langle D, \{V, E, R\}\rangle$ where $V(x)$ means that x is a vertex, $E(y)$ that y is an hyperedge and $R(x, y)$ that x is a vertex of the hyperedge y.

$$HS(T) \equiv \forall x(T(x) \Rightarrow V(x)) \wedge \forall y \exists x E(y) \Rightarrow (T(x) \wedge R(x, y))$$

Therefore the problem ENUM·HS is in ENUM·$\mathbf{\Pi}_2$.

Note that in the query problem ENUM·Φ the formula is fixed i.e. is not part of the input. The complexity is evaluated in terms of the structure/data only. For such problems, the notion of constant delay makes sense:

- when the free variables are all first order (in that case each output is of constant size)
- *but also* and more interestingly when there are second order variables and that computing the next solution from the preceeding one can be done by changing a constant number of tuples.

2 Enumeration for Σ_0 formulas

In this section, we give enumeration algorithms for the most simple class, that is ENUM·$\mathbf{\Sigma}_0$.

Since it is a core procedure of our algorithms, we need to recall how to enumerate all k-ary relations over any domain with constant delay.

▶ **Lemma 4** (Gray code enumeration). *Let D be a finite set, $k \in \mathbb{N}$ and $t_1, \ldots, t_a$, $s_1, \ldots, s_b$ in D^k. Let $\mathcal{R} = \{R \subseteq D^k : t_1, \ldots, t_a \in R, s_1, \ldots, s_b \notin R\}$. Then, starting from the relation $R = \{t_1, \ldots, t_a\}$, one can enumerate the relations belonging to $\mathcal{R}$ with precomputation and delay in $O(1)$. Moreover, the process ends by producing a relation R' such that $|R'| = |R|+1$.*

Proof. Since the tuples $t_1, ..., t_a$ must belong to each output, they can simply be fixed and the problem reduces to generate all subsets of $D^k \backslash \{t_1, \ldots, t_a, s_1, \ldots, s_b\}$ of size up to $n = |D|^k - a - b$ starting from the empty set. Clearly, it is equivalent to generate all subsets of $[n]$. Such problems have been widely studied under the name of Gray code enumeration. It is well known that the enumeration can be done in such a way that the size of the symmetric difference $R_1 \Delta R_2$ between two successive outputs R_1 and R_2 is 1. Given an output R_1, one can proceed as follows. If R_1 has an even number of elements, then set $R_2 = R_1 \Delta \{n\}$. If not, let $R_2 = R_1 \Delta \{i-1\}$ where i is the greatest element in R_1. Clearly, the delay is constant provided we have access to the information on the parity of number of elements in R_1 and on the value of such i above. The parity can be stored in one bit, that is changed at each step, while the latter is easy to maintain in constant time by a linked list on the tuples of each produced relation. To start, one only needs to build this datastructure on the first relation R which is of size a. It is easy to see that the enumeration ends up with the relation R' containing $t_1, ..., t_a$ and the tuple of $D^k \backslash \{t_1, \ldots, s_b\}$ indexed by 1. ◀

▶ Remark. Note that the memory space required by the preceeding algorithm is linear in n, the size of one output. It may seem important since, in contrast, the enumeration itself is constant delay. However, some datastructure is required only to navigate inside each output relation and make the necessary local changes easily.

There is a standard way to represent a first order query problem by a propositional satisfiability problem. We recall it below. We will later introduce a more complex representation.

Let σ be a relational signature and let $\mathcal{S}$ be a σ-structure of domain D with $|D| = n$. Let $\Phi(\mathbf{z}, T)$ be a first order formula where $\mathbf{z}$ is a k-uple of first order variables and T is a second order variable of arity r. One rewrites $\Phi(\mathbf{z}, T)$ by $\bigvee_{i=0}^{n^k-1} \Phi(\mathbf{z}_i, T)$ where $\mathbf{z}_i$ is the i^{th} element of D^k (for, say, lexicographic ordering on D^k). In each $\Phi(\mathbf{z}_i, T)$ one replaces inductively (bottom-up in the tree representation of the formula) each sub-formula $\exists \mathbf{y} \varphi(\mathbf{z}_i, \mathbf{y}, T)$ by a disjunction $\bigvee_{j=0}^{n^p-1} \varphi(\mathbf{z}_i, \mathbf{y}_j, T)$ with $|\mathbf{y}| = p$ (and similarly universal quantification by conjunction). Finally, one calls $\tilde{\Phi}_i$ the propositional formula obtained from $\Phi(\mathbf{z}_i, T)$ by replacing every atomic formula $R(\mathbf{w})$ with $R \in \sigma$ by its truth value in $\mathcal{S}$ and we set $\tilde{\Phi} = \bigvee_{i=0}^{n^k-1} \tilde{\Phi}_i$. Variables of $\tilde{\Phi}$ are of the form $T(\mathbf{w})$ with $\mathbf{w} \in D^r$.

We are now ready to state the first result of this section.

▶ **Theorem 5.** Enum·$\mathbf{\Sigma}_0 \subseteq$ Constant-Delay. *More precisely, it can be computed with precomputation $O(|D|^k)$ and delay $O(1)$ where k is the number of free first order variables of the formula and D is the domain of the input structure.*

Proof. Let $\mathcal{S}$ be a σ-structure of domain D. Let $\Phi(\mathbf{z}, T) \in \mathbf{\Sigma}_0$ with T of arity r and let $\tilde{\Phi} = \bigvee_{i=0}^{n^k-1} \tilde{\Phi}_i$ be its associated propositional formula.

The idea of the proof consists in determining some canonical assignements for each $\tilde{\Phi}_i$ from which one can enumerate all models of $\tilde{\Phi}_i$ and then all models of $\tilde{\Phi}$ by disjoint union. Since constant delay is expected, one has to be careful that two consecutive partial enumerations, say for models of $\tilde{\Phi}_i$ and of $\tilde{\Phi}_j$ with $i \neq j$, respectively ends and starts with solutions that are "close" to each other.

Since there is no first order variable other than $\mathbf{z}$, the number of propositional variables appearing in each $\tilde{\Phi}_i$ is bounded by a constant c_i independent of $|\mathcal{S}|$. Let $T(\mathbf{y}_{i,j})$, $j \leq c_i$, be such variables with $\mathbf{y}_{i,j} \in D^r$.

Let $\mathcal{I}(\Phi_i)$ be the set of up to 2^{c_i} models of $\tilde{\Phi}_i$. For $I \in \mathcal{I}(\Phi_i)$, let I_0 (resp. I_1) the set of variables set to false (resp. true) in I. Let $T(\mathbf{z}_i, I)$ be the set of r-uples $\mathbf{y}_{i,j}$ such that $T(\mathbf{y}_{i,j}) \in I_1$. This relation contains at most c_i tuples. Let now $[T(\mathbf{z}_i, I)]$ be the set of

relations generated by $T(\mathbf{z}_i, I)$ i.e. the relations T^* that agrees with $T(\mathbf{z}_i, I)$ on $\mathbf{y}_{i,1}, \ldots, \mathbf{y}_{i,c_i}$. Clearly, for each $i \leq n^k - 1$, the set $\{(\mathbf{z}_i, T^*) : (\mathbf{z}_i, T^*) \in \Phi(\mathcal{S})\}$ is equal to the set:

$$\bigcup_{I \in \mathcal{I}(\Phi_i)} \bigcup_{T^* \in [T(\mathbf{z}_i, I)]} (\mathbf{z}_i, T^*).$$

The enumeration process to compute $\Phi(\mathcal{S})$ when $\Phi \in \mathbf{\Sigma}_0$ can now be described. The precomputation steps are as follows.

- For each $\mathbf{z}_i \in D^k$, compute $\tilde{\Phi}_i$ and the set $\mathcal{I}(\Phi_i)$.
- Compute the set $\mathcal{Z} = \{\mathbf{z}_i \in D^k : \mathcal{I}(\Phi_i) \neq \emptyset\}$.

It holds that $|\mathcal{I}(\Phi_i)| \leq 2^{c_i}$ i.e. is constant. Thus, the precomputation requires time $O(|D|^k)$. Now, the enumeration itself proceeds as follows.

- For each $\mathbf{z}_i \in \mathcal{Z}$, for each $I \in \mathcal{I}(\Phi_i)$, generate all relations $T^* \in [T(\mathbf{z}_i, I)]$ and output $(\mathbf{z}_i, T^*)$.

From Lemma 4, for given $\mathbf{z}_i$ and I, one can enumerate the set $[T(\mathbf{z}_i, I)]$ in delay $O(1)$. Note that for two distinct $\mathbf{z}_i$ and $\mathbf{z}_j$, the set of outputs are disjoints. Similarly, since two distinct assignments $I, I' \in \mathcal{I}(\Phi_i)$ differ for at least one variable, it holds that $[T(\mathbf{z}_i, I)] \cap [T(\mathbf{z}_i, I')] = \emptyset$.

For each $\mathbf{z}_i$ and I, one starts the enumeration with the relation $T(\mathbf{z}_i, I)$ of size less than c_i and ends by a relation $T'(\mathbf{z}_i, I) \in [T(\mathbf{z}_i, I)]$ with $|T'(\mathbf{z}_i, I)| = |T(\mathbf{z}_i, I)| + 1$. Then, for all $I' \in \mathcal{I}(\Phi_i)$, $I' \neq I$:

$$|T'(\mathbf{z}_i, I) \Delta T(\mathbf{z}_i, I')| \leq 2c_i + 1.$$

Similarly, for all $\mathbf{z}_j \neq \mathbf{z}_i$, and all $I' \in \mathcal{I}(\Phi_j)$ it holds:

$$|T'(\mathbf{z}_i, I) \Delta T(\mathbf{z}_j, I')| \leq c_i + c_j + 1.$$

Then, the enumeration process remains constant delay when branching from one assignment I to the next and when branching from one $\mathbf{z}_i$ to the next. ◀

Is it possible to improve Theorem 5 to find a constant delay enumeration algorithm for $\mathbf{\Sigma}_0$ formulas with a *fixed* polynomial (i.e. of degree independent of the formula size) precomputation? A partial negative answer comes from the following remark. Note that the k-Clique problem can be expressed at this level on finite ordered graph. For instance, for $k = 3$ (see [12]):

$$\Phi(z_1, z_2, z_3) \equiv z_1 < z_2 \wedge z_2 < z_3 \wedge E(z_1, z_2) \wedge E(z_2, z_3) \wedge E(z_3, z_1)$$

Recall that the precomputation plus the delay (which is constant in Theorem 5) correspond to the time necessary to produce the first output, hence to decide if the problem has at least one solution. Then, a *fixed* polynomial precomputation for Enum·$\mathbf{\Sigma}_0$ would provide a fixed parameter tractable algorithm for the parameterized clique problem (see [8] for definition and references on parameterized complexity). Such an algorithm is generally not believed to exist (unless the two parameterized classes $\mathbf{W}[1]$ and $\mathbf{FPT}$ coincide). However,

as shown below, such an improvement of Theorem 5 can be found for some restricted class of structures as input.

A structure $\mathcal{S} = \langle D, R_1, \ldots, R_i\rangle$ is of degree bounded by $d \in \mathbb{N}$ (i.e. is d-degree bounded), if for every $x \in D$, x occurs in at most d tuples of each relation R_i. The following result shows that in the case of bounded degree structures as input an algorithm with linear precomputation can be found. It is proved by using a representation of the query problem by a mixed problem combining querying (but without second order variable) and satisfiability testing.

▶ **Theorem 6.** *Let $d \in \mathbb{N}$. On d-degree bounded input structures,* $\textsc{Enum}\cdot\mathbf{\Sigma}_0 \in \textsc{Delay}(|D|, 1)$ *where D is the domain of the input structure $\mathcal{S}$.*

Proof. One difference with the proof of Theorem 5 is that $\Phi(\mathbf{z}, T)$ is now represented by a pair made of a propositional formula $\overline{\Phi}$ and a set of $\mathbf{\Sigma}_0$ formulas (interpreted on bounded structure) without second order free variables.

Let c be the total number of distinct atomic formulas that appears in $\Phi(\mathbf{z}, T)$. Each atom is of the form $T(\mathbf{y})$ or of the form $R(\mathbf{x})$ with $R \in \sigma$ and $\mathbf{x}, \mathbf{y}$ subsets of $\mathbf{z}$. Clearly, $\Phi(\mathbf{z}, \mathbf{T})$ can be seen as an "abstract" propositional formula denoted by $\overline{\Phi}$ over propositional variables $T(\mathbf{y})$ and $R(\mathbf{x})$ where $\mathbf{y}$ and $\mathbf{x}$ are simply viewed as indices. Let $\mathcal{J}(\overline{\Phi})$ be the set of up to 2^c models of $\overline{\Phi}$. One can recover elements $(\mathbf{z}^*, \mathbf{T}^*)$ of $\Phi(\mathcal{S})$ from the satisfying assignments of $\overline{\Phi}$ as follows. Let $J \in \mathcal{J}(\overline{\Phi})$ and J_0 (resp. J_1) the set of variables set to false (resp. true) in J. Let us consider the first-order formula $\varphi_J(\mathbf{z})$ on signature σ below:

$$\bigwedge_{T(\mathbf{y})\in J_0} \bigwedge_{T(\mathbf{y}')\in J_1} \bigvee_{i=1}^{r} y_j \neq y'_j \wedge \bigwedge_{R(\mathbf{x})\in J_1} R(\mathbf{x}) \wedge \bigwedge_{R(\mathbf{x})\in J_0} \neg R(\mathbf{x}),$$

Let also:

$$\varphi_J(\mathcal{S}) = \{\mathbf{z}^* : \ \langle S, \mathbf{z}^*\rangle \models \varphi_J(\mathbf{z})\}.$$

For $\mathbf{z}^* \in \varphi_J(\mathcal{S})$, we denote by $J(\mathbf{z}^*)$ the truth assignements of the c tuples (of the form $T(\mathbf{y})$ or $R(\mathbf{x})$ with $\mathbf{x}$ and $\mathbf{y}$ subsets of variables taken from $\mathbf{z}$) induced by J after instantiation of the variables in $\mathbf{z}$ by $\mathbf{z}^*$. In other words, in $J(\mathbf{z}^*)$, $T^*(\mathbf{y}^*)$ is true iff $T(\mathbf{y}) \in J_1$ and $R^*(\mathbf{x}^*)$ is true iff $R(\mathbf{x}) \in J_1$.

We now compare with the formulas $\tilde{\Phi}$ in Theorem 5. The following are true:

- Let $J \in \mathcal{J}(\overline{\Phi})$ and $\mathbf{z}_i$, the ith element of D^k. Suppose that $\mathbf{z}_i \in \varphi_J(S)$ then, $J(\mathbf{z}_i) \in \mathcal{I}(\Phi_i)$.

- Conversely, let $\mathbf{z}_i \in D^k$ and $I \in \mathcal{I}(\Phi_i)$ then, there exists $J \in \mathcal{J}(\overline{\Phi})$ such that $\mathbf{z}_i \in \varphi_J(S)$ and $I = J(\mathbf{z}_i)$

From the discussion above, the following holds:

$$\Phi(\mathcal{S}) = \bigcup_{J\in\mathcal{J}(\overline{\Phi})} \bigcup_{\mathbf{z}^*\in\varphi_J(S)} \bigcup_{T^*\in[T(z^*, J(\mathbf{z}^*))]} (\mathbf{z}^*, T^*) \tag{1}$$

Let J and J' be distinct assignments. Observe that, if there exists an atomic formula $R(\mathbf{x})$ over which J and J' has a different value then $\varphi_J(S) \cap \varphi_{J'}(S) = \emptyset$. In this case, the two sets

$$\bigcup_{\mathbf{z}^*\in\varphi_J(S)} \bigcup_{T^*\in[T(z^*, J(\mathbf{z}^*))]} (\mathbf{z}^*, T^*) \text{ and } \bigcup_{\mathbf{z}^*\in\varphi_{J'}(S)} \bigcup_{T^*\in[T(z^*, J'(\mathbf{z}^*))]} (\mathbf{z}^*, T^*)$$

are obviously disjoint. Moreover, if J and J' agree on all atomic formulas of the form $R(\mathbf{x})$, then they differ on at least one $T(\mathbf{y})$ and, in this case $[T(z^*, J(\mathbf{z}^*))] \cap [T(z^*, J'(\mathbf{z}^*))] = \emptyset$. Thus the two above sets are also disjoint even if there might exist $\mathbf{z}^* \in \varphi_J(S) \cap \varphi_{J'}(S)$.

We can now describe how to enumerate $\Phi(\mathcal{S})$. The precomputation process is as follows.

- Compute $\overline{\Phi}$, $\mathcal{J}(\overline{\Phi})$ and, for each $J \in \mathcal{J}(\overline{\Phi})$, the formula $\varphi_J(\mathbf{z})$. All this can be achieved in constant time.
- For each $J \in \mathcal{J}(\overline{\Phi})$, run the necessary precomputation phase to enumerate the elements of $\varphi_J(\mathcal{S})$. From [6] it is known that enumerating the result of a first order query over a structure of bounded degree i.e. computing $\varphi_J(\mathcal{S})$ can be done with a $O(|D|)$ precomputation and a $O(1)$ delay. Hence, the total precomputation phase requires $O(|D|)$ steps.

For the enumeration phase, we conclude as for Theorem 5, taking into account that all components in Equation (1) are pairwise disjoints. ◀

▶ Remark. Each query $\varphi_J(\mathbf{z})$ in the above proof is evaluated on a bounded degree structure which makes the global enumeration tractable. However, the representation of a $\mathbf{\Sigma}_0$ formula $\Phi(\mathbf{z}, T)$ by an abstract propositional formula $\overline{\Phi}$ and a collection of $\mathbf{\Sigma}_0$ formulas $\varphi_J(\mathbf{z})$ without second order variable is general. Then, if $\mathscr{S}$ is any class of structures on which queries of the form $\varphi_J(\mathbf{z})$ admit a linear precomputation and constant delay algorithm then, on $\mathscr{S}$, it also holds that $\textsc{Enum}\cdot\mathbf{\Sigma}_0 \subseteq \textsc{Delay}(|D|, 1)$.

3 Enumeration for Σ_1 formulas

In this section, we prove a lemma, which allows to enumerate the union of the solutions of two enumeration problems with a manageable delay. It is then used to prove that $\textsc{Enum}\cdot\mathbf{\Sigma}_1 \subseteq \textsc{DelayP}$.

▶ **Definition 7.** Let $R(x, y)$ and $S(x, y)$ be two polynomially balanced predicates. The union of R and S, denoted by $(R \cup S)$, is defined by: for all x, y, $(R \cup S)(x, y)$ holds if and only if $R(x, y)$ holds or $S(x, y)$ holds.

Recall that $R(x)$ denotes the finite set $\{y \mid R(x, y)\}$. Assume that $\textsc{Enum}\cdot R$ and $\textsc{Enum}\cdot S$ are in $\textsc{Delay}(g(n), f(n))$. If, for all x, $R(x) \cap S(x) = \emptyset$ then $\textsc{Enum}\cdot(R \cup S) \in \textsc{Delay}(g(n), f(n))$. Similarly, if there exist algorithms with precomputation $g(n)$ and delay $f(n)$ that enumerate the solutions of $\textsc{Enum}\cdot R$ and $\textsc{Enum}\cdot S$ with respect to the same linear ordering $<$ on the output space, then $\textsc{Enum}\cdot(R \cup S) \in \textsc{Delay}(g(n), f(n))$ ([6, 2]). The following result shows that some kind of closure under union can be established without disjointness conditions nor assumption on the ordering of enumeration.

▶ Proposition 8. Let $f : \mathbb{N} \to \mathbb{N}$, $g : \mathbb{N} \to \mathbb{N}$, $h : \mathbb{N} \to \mathbb{N}$ and R, S be two polynomially balanced predicates such that S can be decided in time $O(h(n))$. Suppose that $\textsc{Enum}\cdot R$ and $\textsc{Enum}\cdot S$ are in $\textsc{Delay}(g(n), f(n))$ then, $\textsc{Enum}\cdot R \cup S$ is in $\textsc{Delay}(g(n), f(n) + h(n))$.

Proof. Let M_R and M_S be two RAM machines, which solve $\textsc{Enum}\cdot R$ and $\textsc{Enum}\cdot S$. One builds a machine $M_{(R \cup S)}$ which solves $\textsc{Enum}\cdot(R \cup S)$ by running M_R and M_S in parallel on the instance x. The behavior of $M_{(R \cup S)}$ is described in Algorithm 1.

At each step $M_{(R \cup S)}$ produces a new solution y of $R(x)$ thanks to M_R and it tests if $y \in S(x)$ in time $h(|x|)$, by hypothesis. If $y \notin S(x)$ it outputs it, otherwise it is discarded

Algorithm 1: Enumeration algorithm for Enum·$(R \cup S)$

```
Data: An instance x
Result: The elements of R(x) ∪ S(x)
y1 ⟵ First element of the enumeration of R(x)
y2 ⟵ First element of the enumeration of S(x)
while y1 ≠ END ∨ y2 ≠ END do
    if y1 ≠ END ∧ y1 ∉ S(x) then
        Output y1
    else
        Output y2 ;
        y2 ⟵ next element of the enumeration of S(x)
    if y1 ≠ END then
        y1 ⟵ next element of the enumeration of R(x)
```

and the next solution of $S(x)$ given by M_S is computed and outputted[2]. If there is no solution left in $R(x)$ (resp. $S(x)$), it finishes the enumeration thanks to M_R (resp. M_S).

Remark that if $M_{(R\cup S)}$ has enumerated k elements of $S(x)$ thanks to M_S then it has also found and discarded k elements of $R(x) \cap S(x)$ given by M_R. Therefore if $M_{(R\cup S)}$ has outputted all $S(x)$, it has used M_R to produce $|S(x)|$ elements of $R(x) \cap S(x)$, which must then satisfy $S(x) = S(x) \cap R(x)$. Therefore the enumeration of the remaining elements of $R(x)$ does not create any repetition. Moreover all elements of $R(x) \cap S(x)$ are enumerated by M_S only, thus the algorithm makes no repetition.

Since, at each step of the algorithm we simulate M_R and M_S to let them produce at most one solution, the delay of $M_{(R\cup S)}$ is bounded by the sum of the delays of M_R and M_S, that is $2f(|x|)$ plus $h(|x|)$ the time to do one membership test. ◀

▶ **Corollary 9.** *Let $\Phi(\mathbf{y}, T) = \exists \mathbf{x} \varphi(\mathbf{x}, \mathbf{y}, T)$ be a first order formula with $|\mathbf{x}| = k$. Assume that there is an algorithm such that, for all input structures $\mathcal{S}$ of domain D and for all k-tuples $\mathbf{x}^*$ of $\mathcal{S}$, enumerates the elements of $\Phi_{\mathbf{x}^*}(\mathcal{S})$ where $\Phi_{\mathbf{x}^*}(\mathbf{y}, T) = \varphi(\mathbf{x}^*, \mathbf{y}, T)$, with precomputation $g(|D|)$ and delay $f(|D|)$. Then* Enum·Φ *can be computed with a $O(g(|D|)|D|^k)$ precomputation and a delay $O(f(|D|)|D|^k)$.*

Proof. Remark that, for all models $\mathcal{S}$ of domain D, $\Phi(\mathcal{S}) = \cup_{\mathbf{x}^* \in D^k} \Phi_{\mathbf{x}^*}(\mathcal{S})$. We can apply the previous proposition to this union of $|D|^k$ enumerations problems. For each $\mathbf{x}^* \in D^k$, one has to compute $\Phi_{\mathbf{x}^*}(\mathbf{y}, T)$ and do the corresponding precomputation in time $O(g(|D|))$, which accounts for a total precomputation of $O(g(|D|)|D|^k)$. A formula $\Phi_{\mathbf{x}^*}(\mathbf{y}, T)$ is of constant size, therefore checking if $(\mathcal{S}, \mathbf{y}^*, T^*) \models \Phi_{\mathbf{x}^*}(\mathbf{y}, T)$ can be done in constant time. By induction, one can easily generalize Proposition 8 to handle the union of $|D|^k$ predicates. This yields a delay in $O(|D|^k \times f(|D|) + |D|^k) = O(f(|D|)|D|^k)$. ◀

The previous corollary allows to remove the first level of existential quantification of any formula with a polynomial slowdown only. As a consequence, we have a polynomial delay enumeration algorithm for any problem in Enum·$\mathbf{\Sigma}_1$.

▶ **Theorem 10.** Enum·$\mathbf{\Sigma}_1 \subseteq$ DelayP. *More precisely,* Enum·$\mathbf{\Sigma}_1$ *can be computed with precomputation $O(|D|^{h+k})$ and delay $O(|D|^k)$ where h is the number of free first order vari-*

[2] note that it can be y itself

ables of the formula, k the number of existentially quantified variables and D is the domain of the input structure.

Proof. Let $\exists \mathbf{x}\varphi(\mathbf{x}, \mathbf{y}, T)$ be a formula of $\mathbf{\Sigma}_1$ and $\mathcal{S}$ be a structure. By Theorem 5, we know that for each k-uple $\mathbf{x}^*$, the solutions of $\varphi(\mathbf{x}^*, \mathbf{y}, T)$ can be enumerated with precomputation $O(|D|^h)$ and a delay $O(1)$. Thus, by Corollary 9, we know that Enum·$\exists \mathbf{x}\varphi(\mathbf{x}, y, T)$ can be computed with precomputation $O(|D|^{h+k})$ and delay $O(|D|^k)$. ◀

Again, for $\mathbf{\Sigma}_1$ queries on structures of bounded degree a better bound can be found at least for the model checking problem. The following holds with a proof similar to (the first steps of) that of Theorem 6.

▶ Proposition 11. Let $d \in \mathbb{N}$. Checking whether $\Phi(\mathcal{S}) = \emptyset$ where $\Phi(\mathbf{z}, T) \in \mathbf{\Sigma}_1$ and $\mathcal{S}$ is a d-degree bounded input structures can be done in time $O(|D|)$ (in data complexity).

It is however open whether the result can be extended in the enumeration setting to prove a linear delay algorithm for this latter kind of query.

3.1 Relation with DNF formulas

In this part, we examine more closely the relationships between the enumeration problem for $\mathbf{\Sigma}_1$-queries and the enumeration of the solutions of restricted DNF-formulas. Let ψ be a DNF-formula such that each clause is of size at most l. Remark that the number of clauses cannot be larger than n^l where n is the number of variables. We say that such a formula is in DNF(l) and we note Enum·DNF(l) the problem Enum·DNF restricted to DNF(l).

Let now be Φ a formula $\exists \mathbf{x}\varphi(\mathbf{x}, T)$ where T is a second order variable of arity 1 and the tuple $\mathbf{x}$ is such that $|\mathbf{x}| = k$. We also assume that φ is quantifier-free in disjunctive normal form and that each of its clauses contains at most l occurrences of a term involving T. Remark that $l \leq k$, because each occurrence of T in a clause must be applied to a different variable. On the other hand, if $l < k$, one can rename the variables used in each clause of φ so that we obtain an equivalent formula with only k variables. Therefore the parameters k and l are essentially the same. We denote by $\Sigma_1(l)$ the set of such formulas (with $k = l$).

▶ Remark. Here we do not allow free first-order variables. It is always possible to take care of them with a polynomial slowdown in the precomputation only.

Moreover, the restriction on the arity of the second order variable could be lifted and we would obtain essentially the same results. We choose this restriction, because in this setting, we have the parameter k and l equal which makes the next propositions easier to state and understand.

▶ Proposition 12. If Enum·DNF(l) can be solved with precomputation $g(n)$ and delay $f(n)$, where n is the number of variables of the formula, then for all formulas $\Phi \in \Sigma_1(l)$, Enum·Φ can be solved with precomputation $g(n)$ and delay $f(n)$, where n is the size of the domain.

Proof. To prove that, fix a formula $\Phi \equiv \exists \mathbf{x}\varphi(\mathbf{x}, T) \in \Sigma_1(l)$. Let $\mathcal{S}$ be the input structure and D its domain. Let $\tilde{\Phi}$ be the propositional formula associated to Φ as before. It is the disjunction of the n^l formulas $\Phi(\mathbf{x}^*, T)$. Each of the formula $\Phi(\mathbf{x}^*, T)$ is a DNF-formula with clauses of size at most l. Therefore the formula $\tilde{\Phi}$ is in DNF(l), has $|D|$ variables and its solutions are in bijection with the solutions of Φ. ◀

▶ Proposition 13. There is a formula Φ in $\Sigma_1(l, l)$ such that the following holds. If Enum·Φ can be solved with precomputation $g(n)$ and delay $f(n)$, where n is the size of the domain, then Enum·DNF(l) can be solved with precomputation $g(n)$ and delay $f(n)$, where n is the number of variables of the formula.

Proof. Let σ be the language $\{P_{i,j}\}_{i+j\leq l}$, where $P_{i,j}$ is a l-ary predicate. A predicate $P_{i,j}$ represents, in the reduction, a clause whose first i variables appear positively and the next j appear negatively. The second order variable T represents the set of variables set to true. Let

$$\theta_{i,j}(T, x_1, \ldots, x_l) \equiv \bigwedge_{s\leq i} x_s \in T \wedge \bigwedge_{i<s\leq l} x_s \notin T$$

and let

$$\Phi \equiv \exists x_1, \ldots, x_l \bigvee_{i+j\leq l} (P_{i,j}(x_1, \ldots, x_l) \wedge \theta_{i,j}(T, x_1, \ldots, x_l)).$$

Let now ψ be a DNF-formula over the variables $V = \{v_1, \ldots, v_n\}$. We reduce the enumeration of the solutions of ψ to the enumeration of $\Phi(\mathscr{S})$. The domain of $\mathscr{S}$ is the set V and $P_{i,j}(x_1, \ldots, x_l)$ holds if and only if there is a clause in ψ whose variables appearing positively are $x_1, \ldots, x_i$ and those appearing negatively are $x_{i+1}, \ldots, x_{i+j}$. Remark now that $T^* \in \Phi(\mathcal{S})$ if and only if T^* represents an assignment of the variables in V which satisfies ψ. Thus the solutions of ψ are in bijection with $\Phi(\mathcal{S})$ which achieves the proof. ◀

The above propositions shows how the enumeration complexity of $\mathbf{\Sigma}_1$ queries and Enum·DNF are intimately related. Hence, to improve our results on Enum·$\mathbf{\Sigma}_1$, one has to study the problem Enum·DNF(l). The following question seems quite challenging:

Open Question: prove (or disprove) that there exists an enumeration algorithm for Enum·DNF(l) whose delay does not depend on l or whose delay is better than $O(n^l)$.

4 Enumeration for $\mathbf{\Pi}_1$ formulas and beyond

In [12], it is shown that the propositional satisfiability problem for a 3-CNF formula can be expressed as a query problem for a $\mathbf{\Pi}_1$ formula. The following result then holds.

▶ Proposition 14. Unless P = NP, there is no polynomial delay algorithm for Enum·$\mathbf{\Pi}_1$. The results still holds even for structure of bounded degree as input.

Proof. See [12]. For the case of bounded degree structures, remark that it is well-known that the satisfiability problem is hard even for 3-CNF formulas such that each variable appears (positively or negatively) in at most 3 clauses. For such propositional formulas, the structures obtained after reduction in [12] is of bounded degree. ◀

As it is shown below, it is even possible to define the satisfiability problem by a quite restricted $\mathbf{\Pi}_1$ formula. A 3-CNF formula φ can be encoded by a structure $\mathcal{S}_\varphi$ of signature $\{C, a_1, a_2, a_3, a_4\}$ where C is a 4-ary predicate and a_1, a_2, a_3, a_4 are constants. The domain of $\mathcal{S}_\varphi$ contains as many elements as variables in φ. Let x, y and z be elements of the domain, $C(a_i, x, y, z,)$ is true if the clause $\neg_{i,1}x \vee \neg_{i,2}y \vee \neg_{i,3}z$ appears in φ where $\neg_{i,j} = \neg$ if $i \leq j$ (and $\neg_{i,j} = \epsilon$ if not). In other words, i encodes the number of variables that appear negatively in the clause. Let $\Psi(T, T_1, T_2, T_3)$ be the following $\mathbf{\Pi}_1$ formula:

$$\begin{array}{l} \forall x_1 \forall x_2 \forall x_3 \forall a \\ (C(a, x_1, x_2, x_3) \rightarrow T_1(a, x_1) \vee T_2(a, x_2) \vee T_3(a, x_3)) \wedge \\ \bigwedge_{i=1}^{4} \bigwedge_{j=1}^{3} T_j(a_i, x_1) \leftrightarrow \neg_{i,j} T(x_1) \end{array} \quad (2)$$

Clearly, there is a bijective correspondence between the satisfying assignments of φ and the set $\Psi(\mathcal{S}_\varphi)$. Remark now that the quantifier free part of Ψ is in CNF and is such that all its clauses except one have at most two occurences of a second-order free variable.

In [12], a first-order formula Φ defines the problem of computing the cardinal of $\Phi(\mathcal{S})$. Theorem 2 of [12] describes the strict inclusions of the classes of counting functions defined by the number of quantifier alternations. It can be easily transposed into the following theorem on enumeration problems, if we assume all models to have a total order.

▶ **Theorem 15.** *On linearly ordered structures, we have the following inclusions:* $\textsc{Enum}\cdot\mathbf{\Sigma}_0 \subsetneq \textsc{Enum}\cdot\mathbf{\Sigma}_1 \subsetneq \textsc{Enum}\cdot\mathbf{\Pi}_1 \subsetneq \textsc{Enum}\cdot\mathbf{\Sigma}_2 \subsetneq \textsc{Enum}\cdot\mathbf{\Pi}_2$.

Moreover, there is a problem in $\textsc{Enum}\cdot\mathbf{\Pi}_2$, which is complete up to parsimonious reduction for all problems definable by a polynomially balanced predicate (a polynomial time reduction f between two decision problems A and B is *parsimonious* if, for each valid instance x, it establishes a bijective correspondance between the solutions sets $A(x)$ and $B(f(x))$. See, for example, [12] for a precise definition). Therefore the hierarchy collapses at $\textsc{Enum}\cdot\mathbf{\Pi}_2$.

4.1 Feasible classes beyond $\mathbf{\Sigma}_1$

We now consider fragments of $\mathbf{\Pi}_2$ and $\mathbf{\Sigma}_2$ with a good expressive power and whose associated enumeration problems remain tractable. Let $\mathcal{C}$ be a subclass of propositional formulas.

$\textsc{Enum}\cdot\text{SAT}(\mathcal{C})$
Input: A propositional CNF formula φ in $\mathcal{C}$
Output: an enumeration of the satisfying assignments of φ.

A CNF formula is Horn (resp. anti-Horn) if it is equivalent to a formula whose clauses have at most one positive (resp. negative) literal. It is bijunctive if it is equivalent to a CNF formula with clauses of length two. Finally, it is affine if it is equivalent to a system of linear equations over the two-element field. We first examinate the immediate consequence of the following result and the fact that to solve $\textsc{Enum}\cdot\Phi$, one only has to enumerate the solutions obtained from those of $\tilde{\Phi}$ as in Theorem 5.

▶ Proposition 16 ([5]). The problem $\textsc{Enum}\cdot\text{SAT}(\mathcal{C})$ is in DelayP when $\mathcal{C}$ is one of the following classes: Horn formulas, anti-Horn formulas, affine formulas, bijunctive (2CNF) formulas

▶ **Corollary 17.** *Let $\Phi(\mathbf{z}, T)$ be a formula, such that, for all σ structures, all propositional formulas $\tilde{\Phi}_i$ are either Horn, anti-Horn, affine or bijunctive. Then* $\textsc{Enum}\cdot\Phi \in \textsc{DelayP}$.

Proof. Let $\Phi(\mathbf{z}, T)$ be a formula, with $|\mathbf{z}| = k$ and $\mathcal{S}$ a structure of domain D. Let $\mathbf{z}_i$ be an enumeration of the k-tuples of D. Recall that the set of solutions of $\Phi(\mathbf{z}_i, T)$ is equal to:

$$\bigcup_{I \in \mathcal{I}(\Phi_i)} \bigcup_{T^* \in [T(\mathbf{z}_i, I)]} (\mathbf{z}_i, T^*).$$

Furthermore, we know that $\tilde{\Phi}_i$ is either Horn, anti-Horn, affine or bijunctive and that it is a $\mathbf{\Pi}_1$ formula. By construction, it is of size polynomial in $|D|$, hence its models can be enumerated in polynomial delay by Proposition 16. The enumeration of the solutions of $\textsc{Enum}\cdot\Phi$ on the model $\mathcal{S}$ is done in polynomial delay as follows:

- for each $\mathbf{z}_i$ compute the formula $\tilde{\Phi}_i$

- enumerate the models of each $\tilde{\Phi}_i$ in polynomial delay
- for each model I of $\tilde{\Phi}_i$, build in polynomial time the solution $(\mathbf{z}_i, T(\mathbf{z}_i, I))$
- for each solution $(\mathbf{z}_i, T(\mathbf{z}_i, I))$, generate by Gray code enumeration the solutions $(\mathbf{z}_i, T^*)$ with $T^* \in [T(\mathbf{z}_i, I)]$

Remark that for two differents $\mathbf{z}_i$ the enumerated solutions are disjoint.

Moreover, for $I \neq J$, $T(\mathbf{z}_i, I)$ and $T(\mathbf{z}_i, J)$ differs on at least one value. Hence, $T(\mathbf{z}_i, I) \cap T(\mathbf{z}_i, J) = \emptyset$. Therefore, there are no repetition in the previously described algorithm. ◀

The condition in Corollary 17 is semantic: it applies to $\tilde{\Phi}$ and not to Φ, which makes it not obvious to characterize. The following result holds and contrast with the case of Formula (2) which shows that $\mathbf{\Pi}_1$ queries in conjunctive normal form that have one clause with three occurrences of a second order variable are hard to enumerate.

▶ **Corollary 18.** *Let $\Phi(\mathbf{z}, T) \equiv \exists \mathbf{y} \forall \mathbf{x} \Psi(\mathbf{x}, \mathbf{y}, \mathbf{z}, T)$ where Ψ is in conjunctive normal form and all its clauses contain at most 2 occurrences of a free predicate then* Enum·$\Phi \subseteq$ DelayP.

Proof. Let $\mathcal{S}$ be a finite structure and $\Phi(\mathbf{z}, T)$ as above. For such a $\Phi(\mathbf{z}, T)$, the formula $\tilde{\Phi}_i$ is of the form (set $|y| = p$)

$$\bigvee_{j=1}^{n^p - 1} \Psi_i(\mathbf{z}_i, \mathbf{y}_j)$$

where $\Psi_i(\mathbf{z}_i, \mathbf{y}_j)$ is a 2-CNF formula of size polynomial in $|S|$. From Proposition 16, models of such formulas can be enumerated with polynomial delay. The union of models of the polynomially many formulas $\Psi_i(\mathbf{z}_i, \mathbf{y}_j)$ can be enumerated following the method of Proposition 8. The delay is then polynomial. ◀

The above corollary applies to $R\Sigma_2$ formulas defined in [12]. It has been shown there that counting the models of such formulas can be done by a fully polynomial randomized approximation scheme.

▶ **Example 19.** The formula $IS(T) \equiv \forall x \forall y T(x) \wedge T(y) \Rightarrow \neg E(x, y)$ satisfies the condition of the previous corollary therefore Enum·IS $\in$ DelayP. Some other interesting objects such as vertex covers can be defined by a formula of this form.

The next result is similar to corollary 18, but it uses in its proof a Horn or an anti-Horn formula instead of a 2-CNF formula.

▶ **Corollary 20.** *Let $\Phi(z, T) \equiv \forall x \exists y \Psi_1(x, y, z, T)$ where Ψ is in disjunctive normal form such that each of its clauses contain only one occurence of a free second order variable and all these occurences are of the same polarity. Then* Enum·$\Phi \subseteq$ DelayP.

▶ **Example 21.** The formula $DS(T) \equiv \forall x \exists y\, T(y) \wedge E(x, y)$ holds if and only if T is a dominating set. Since $DS(T)$ satisfies the hypothesis of Corollary 20, Enum·DS $\in$ DelayP.

▶ **Example 22.** Recall that $HS(T) \equiv \forall x (T(x) \Rightarrow V(x)) \wedge \forall y \exists x E(y) \Rightarrow (T(x) \wedge R(x, y))$ characterizes the hitting sets of an hypergraph. It does not exactly satisfy the hypothesis of the previous corollary because T appears with different polarity in $\forall x (T(x) \Rightarrow V(x))$ and in $\forall y \exists x E(y) \Rightarrow (T(x) \wedge R(x, y))$. However, if we consider the formula $\tilde{HS}(T)_i$, we see that it is a Horn formula which enables us to conclude by Corollary 17 that Enum·HS $\in$ DelayP.

5 Concluding remarks

The results of this paper try to give a first overview of the complexity of first order query problems with possibly free second order variables. Not surprisingly, the complexity increases rapidly with alternation of quantifiers: if the first levels $\mathbf{\Sigma}_0$ and $\mathbf{\Sigma}_1$ admit efficient enumeration algorithm (with constant or polynomial delay), the $\mathbf{\Pi}_1$ is already able to express hard problems. However, some interesting subcases beyond $\mathbf{\Sigma}_1$ are exhibited which admit rather efficient enumeration algorithms.

An interesting question is whether one can extend our result for first order logic with additional operators (such as fixpoints or maximization/minimization operators). Among them, let Enum·$MaxT\Phi(T)$ be the problem of enumerating all maximal models of φ.

It is easy to see that if Φ satisfies the hypotheses of Corollary 18, then Enum·$MaxT\varphi(T)$ is in DelayP by a result of [9] (this case captures among other things the problem of enumerating the maximal (for inclusion) independent sets of a graph). On the other hand, since enumerating the maximal models of a Horn formula is hard (see [9] also), obtaining such a result when hypotheses of Corollary 20 are satisfied seems very unlikely.

References

1 G. Bagan. Mso queries on tree decomposable structures are computable with linear delay. In *Computer Science Logic*, volume 4646, pages 208–222, 2006.

2 G. Bagan. *Algorithmes et Complexité des Problèmes d'Énumération pour l'Évaluation de Requêtes Logiques*. PhD thesis, Université de Caen, 2009, 2009.

3 G. Bagan, A. Durand, and E. Grandjean. On acyclic conjunctive queries and constant delay enumeration. In *Computer Science Logic, 21st International Workshop, CSL 2007, 16th Annual Conference of the EACSL*, volume 4646 of *Lecture Notes in Computer Science*, pages 208–222. Springer, 2007.

4 B. Courcelle. Linear delay enumeration and monadic second-order logic. *Discrete Applied Mathematics*, 157(12):2675–2700, 2009.

5 N. Creignou and J.J. Hébrard. On generating all solutions of generalized satisfiability problems. *RAIRO Theoretical Informatics and Applications*, 31(6), 1997.

6 A. Durand and E. Grandjean. First-order queries on structures of bounded degree are computable with constant delay. *ACM Trans. Comput. Log.*, 8(4), 2007.

7 R. Fagin. Generalized first-order spectra and polynomial-time recognizable sets. *American Mathematical Society*, pages 43–74, 1974.

8 J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer, 2006.

9 D.J. Kavvadias, M. Sideri, and E.C. Stavropoulos. Generating all maximal models of a Boolean expression. *Information Processing Letters*, 74(3-4):157–162, 2000.

10 L. Libkin. *Elements of finite model theory*. EATCS Series. Springer, 2004.

11 S. Lindell. A normal form for first-order logic over doubly-linked data structures. *International Journal of Foundations of Computer Science*, 19(1):205–217, 2008.

12 S. Saluja, K.V. Subrahmanyam, and M.N. Thakur. Descriptive complexity of# P functions. *Journal of Computer and System Sciences*, 50(3):493–505, 1995.