On the Complexity of Hard Enumeration Problems*

Nadia Creignou[1], Markus Kröll[2], Reinhard Pichler[2], Sebastian Skritek[2], and Heribert Vollmer[3]

[1] Aix-Marseille Univ, CNRS, Marseille, France, creignou@lif.univ-mrs.fr
[2] TU Wien, Vienna, Austria, {kroell | pichler | skritek}@dbai.tuwien.ac.at
[3] Leibniz Universität Hannover, Hannover, Germany, vollmer@thi.uni-hannover.de

Abstract. Complexity theory provides a wealth of complexity classes for analyzing the complexity of decision and counting problems. Despite the practical relevance of enumeration problems, the tools provided by complexity theory for this important class of problems are very limited. In particular, complexity classes analogous to the polynomial hierarchy and an appropriate notion of problem reduction are missing. In this work, we lay the foundations for a complexity theory of hard enumeration problems by proposing a hierarchy of complexity classes and by investigating notions of reductions for enumeration problems.

1 Introduction

While decision problems often ask for the *existence of a solution* to some problem instance, enumeration problems aim at outputting *all solutions*. In many domains, enumeration problems are thus the most natural kind of problems. Just take the database area (usually the user is interested in all answer tuples and not just in a yes/no answer) or diagnosis (where the user wants to retrieve possible explanations, and not only whether one exists) as two examples. Nevertheless, the complexity of enumeration problems is far less studied than the complexity of decision problems.

It should be noted that even simple enumeration problems may produce big output. To capture the intuition of easy to enumerate problems – despite a possibly exponential number of output values – various notions of tractable enumeration classes have been proposed in [13]. The class DelayP ("polynomial delay") contains all enumeration problems where, for given instance x, (1) the time to compute the first solution, (2) the time between outputting any two solutions, and (3) the time to detect that no further solution exists, are all polynomially bounded in the size of x. The class IncP ("incremental polynomial time") contains those enumeration problems where, for given instance x, the time to compute the next solution and for detecting that no further solution exists

* This is a pre-print of an article published in LATA 2017, LNCS 10168, pp. 183–195. The final authenticated version is available online at https://doi.org/10.1007/978-3-319-53733-7_13

is polynomially bounded in the size of both x and of the already computed solutions. Obviously, the relationship $\mathsf{DelayP} \subseteq \mathsf{IncP}$ holds. In [17], the proper inclusion $\mathsf{DelayP} \subsetneq \mathsf{IncP}$ is mentioned. For these tractable enumeration classes, a variety of membership results exist, a few examples are [15, 14, 6, 2, 9]

There has also been work on intractable enumeration problems. Intractability of enumeration is typically proved by showing intractability of a related decision problem rather than directly proving lower bounds by relating one enumeration problem to the other. Tools for a more fine-grained analysis of intractable enumeration problems are missing to date. For instance, up to now we are not able to make a differentiated analysis of the complexity of the following typical enumeration problems:

$\Pi_k\mathrm{SAT}^e$ / $\Sigma_k\mathrm{SAT}^e$ INSTANCE: $\psi = \forall x_1 \exists x_2 \dots Q_k x_k \phi(\boldsymbol{x}, \boldsymbol{y})$ / $\psi = \exists x_1 \forall x_2 \dots Q_k x_k \phi(\boldsymbol{x}, \boldsymbol{y})$ OUTPUT: All assignments for $\boldsymbol{y}$ such that ψ is true.

This is in sharp contrast to decision problems, where the polynomial hierarchy is crucial for a detailed complexity analysis. As a matter of fact, it makes a big difference, if an NP-hard problem is in NP or not. Indeed, NP-complete problems have an efficient transformation into SAT and can therefore be solved by making use of powerful SAT-solvers. Similarly, problems in Σ_2^P can be solved by using ASP-solvers. Finally, also for problems on higher levels of the polynomial hierarchy, the number of quantifier alternations in the QBF-encoding matters when using QBF-solvers. For counting problems, an analogue of the polynomial hierarchy has been defined in form of the $\# \cdot \mathcal{C}$–classes with $\mathcal{C} \in \{\mathsf{P}, \mathsf{coNP}, \Pi_2^P, \dots\}$ [12, 19]. For enumeration problems, no such analogue has been studied.

Goal and Results. The goal of this work is to lay the foundations for a complexity theory of hard enumeration problems by defining appropriate complexity classes for intractable enumeration and a suitable notion of problem reductions. We propose to extend tractable enumeration classes by oracles. We will thus get a hierarchy of classes $\mathsf{DelayP}^{\mathcal{C}}$, $\mathsf{IncP}^{\mathcal{C}}$, where various complexity classes $\mathcal{C}$ are used as oracles. As far as the definition of an appropriate notion of reductions is concerned, we follow the usual philosophy of reductions: if some enumeration problem can be reduced to another one, then we can use this reduction together with an enumeration algorithm for the latter problem to solve the first one. We observe that two principal kinds of reductions are used for decision problems, namely many-one reductions and Turing reductions. Similarly, we shall define a more declarative-style and a more procedural-style notion of reduction for enumeration problems. Our results are summarized below. All missing proof details can be found in the full version of this article [5].

- *Enumeration complexity classes.* In Section 3, we introduce a hierarchy of complexity classes of intractable enumeration via oracles and prove that it is strict unless the polynomial hierarchy collapses.
- *Declarative-style reductions.* In Section 4, we introduce a declarative-style notion of reductions. While they enjoy some desirable properties, we do not succeed in exhibiting complete problems under this reduction.

- *Procedural-style reductions and completeness results.* In Section 5, we introduce a procedural-style notion of reductions and show that they remedy some shortcomings of the declarative-style notion. In particular we prove completeness results. We obtain a Schaefer-like dichotomy complexity classification for the enumeration of models of generalized CNF-formulas.

2 Preliminaries

In the following, Σ denotes a finite alphabet and R denotes a polynomially bounded, binary relation $R \subseteq \Sigma^* \times \Sigma^*$, i.e., there is a polynomial p such that for all $(x, y) \in R$, $|y| \leq p(|x|)$. For every string x, $R(x) = \{y \in \Sigma^* \mid (x, y) \in R\}$. A string $y \in R(x)$ is called a *solution* for x. With a polynomially bounded, binary relation R, we can associate several natural problems:

EXIST_R INSTANCE: $x \in \Sigma^*$ QUESTION: Exists $y \in \Sigma^*$ s.t. $(x, y) \in R$?	EXIST-ANOTHERSOL_R / ANOTHERSOL_R INSTANCE: $x \in \Sigma^*, Y \subseteq R(x)$ OUTPUT: Is $(R(x) \setminus Y) \neq \emptyset$? / $y \in R(x) \setminus Y$ or declare that no such y exists.
CHECK_R INSTANCE: $(x, y) \in \Sigma^* \times \Sigma^*$ QUESTION: Is $(x, y) \in R$?	EXTSOL_R INSTANCE: $(x, y) \in \Sigma^* \times \Sigma^*$ QUESTION: Is there some (possibly empty) $y' \in \Sigma^*$ such that $(x, yy') \in R$?

A binary relation R also gives rise to an enumeration problem, which aims at outputting the function $\mathsf{Sol}_R : \Sigma^* \to 2^{\Sigma^*}, x \mapsto \{y \in \Sigma^* \mid (x, y) \in R\}$.

ENUM_R INSTANCE: $x \in \Sigma^*$ OUTPUT: $R(x) = \{y \in \Sigma^* \mid (x, y) \in R\}$.

We assume the reader to be familiar with the polynomial hierarchy – the complexity classes P, NP, coNP and, more generally, Δ_k^P, Σ_k^P, and Π_k^P for $k \in \{0, 1, \dots\}$. For a definition of the counting hierarchy $\# \cdot \mathcal{C}$ via the complexity of the CHECK_R problem, we refer to [12].

In Section 1, we have already recalled two important tractable enumeration complexity classes, DelayP and IncP from [13]. Note that in [17, 18], these classes are defined slightly differently by allowing only those ENUM_R problems in DelayP and IncP where the corresponding CHECK_R problem is in P. We adhere to the definition of tractable enumeration classes from [13].

A complexity class $\mathcal{C}$ is *closed under a reduction* $\leq_r$ if, for any two binary relations R_1 and R_2 we have that $R_2 \in \mathcal{C}$ and $R_1 \leq_r R_2$ implies $R_1 \in \mathcal{C}$. Furthermore, a reduction $\leq_r$ is *transitive* if for any three binary relations R_1, R_2, R_3, it is the case that $R_1 \leq_r R_2$ and $R_2 \leq_r R_3$ implies $R_1 \leq_r R_3$.

3 Complexity Classes

In contrast to counting complexity, defining a hierarchy of enumeration problems via the CHECK_R problem of binary relations R is not appropriate. This

can be seen by considering artificial problems obtained by padding the set of solutions of any problem with an exponential number of fake (and trivial to produce) solutions. While these fake solutions do not change the complexity of the check problem, enumerating these exponentially many fake solutions first gives an enumeration algorithm enough time to search for the non trivial ones.

Thus, we need an alternative approach for defining meaningful enumeration complexity classes. To this end, we first fix our computation model. We have already observed in the previous section that an enumeration problem may produce exponentially big output. Hence, the runtime and also the space requirements of an enumeration algorithm may be exponential in the input. Therefore, it is common (cf. [17]) to use the RAM model as a computational model, because a RAM can access parts of exponential-size data in polynomial time. We restrict ourselves here to polynomially bounded RAM machines, i.e., throughout the computation of such a machine, the size of the content of each register is polynomially bounded in the size of the input.

For enumeration, we will also make use of RAM machines with an output-instruction, as defined in [17]. This model can be extended further by introducing decision oracles. The input to the oracle is stored in special registers and the oracle takes consecutive non-empty registers as input. Moreover, following [1], we use a computational model that does not delete the input of an oracle call once such a call is made. For a detailed definition, refer to [17] or [5]. It is important to note that due to the exponential runtime of an enumeration algorithm and the fact that the input to an oracle is not deleted when the oracle is executed, the input to an oracle call may eventually become exponential as well. Clearly, this can only happen if exponentially many consecutive special registers are non-empty, since we assume also each special register to be polynomially bounded.

Using this we define a collection of enumeration complexity classes via oracles:

Definition 1 (enumeration complexity classes). *Let* Enum_R *be an enumeration problem, and* $\mathcal{C}$ *a decision complexity class. Then we say that:*

- Enum_$R \in \mathsf{DelayP}^{\mathcal{C}}$ *if there is a RAM machine* M *with an oracle* L *in* $\mathcal{C}$ *such that* M *enumerates* Enum_R *with polynomial delay. The class* $\mathsf{IncP}^{\mathcal{C}}$ *is defined analogously.*
- Enum_$R \in \mathsf{DelayP}^{\mathcal{C}}_{p}$ *if there is a RAM machine* M *with an oracle* L *in* $\mathcal{C}$ *such that for any instance* x*,* M *enumerates* $R(x)$ *with polynomial delay and the size of the input to every oracle call is polynomially bounded in* $|x|$*.*

Note that the restriction of the oracle inputs to polynomial size only makes sense for $\mathsf{DelayP}^{\mathcal{C}}$, where we have a discrepancy between the polynomial restriction (w.r.t. the input x) on the time between two consecutive solutions are output and the possibly exponential size (w.r.t. the input x) of oracle calls. No such discrepancy exists for $\mathsf{IncP}^{\mathcal{C}}$, where the same polynomial upper bound w.r.t. the already computed solutions (resp. all solutions) applies both to the allowed time and to the size of the oracle calls.

We now prove several properties of these complexity classes. First, we draw a connection between the complexity of enumeration and decision problems.

It turns out that in order to study the class $\mathsf{DelayP}_p^{\mathcal{C}}$ the ExtSol_R problem is most relevant. Indeed, the standard enumeration algorithm [17, 6], which outputs the solutions in lexicographical order, gives the following relationship.

Proposition 2. *Let R be a binary relation, $k \geq 0$, and $\mathcal{C} \in \{\Delta_k^P, \Sigma_k^P\}$. If* ExtSol_$R \in \mathcal{C}$ *then* Enum_$R \in \mathsf{DelayP}_p^{\mathcal{C}}$.

An important class of search problems are those for which search reduces to decision, the so-called self-reducible problems. This notion can be captured by the following definition.

Definition 3 (self-reducibility). *Let $\leq_T$ denote Turing reductions. We say that a binary relation R is* self-reducible*, if* ExtSol_$R \leq_T$ Exist_R,

For self-reducible problems the above proposition can be refined as follows.

Proposition 4. *Let R be a binary relation, which is self-reducible, and $k \geq 0$. Then the following holds:* Exist_$R \in \Delta_k^P$ *if and only if* Enum_$R \in \mathsf{DelayP}_p^{\Delta_k^P}$.

The above proposition gives a characterization of the class $\mathsf{DelayP}_p^{\Delta_k^P}$ in terms of the complexity of decision problems in the case of self-reducible relations. Analogously, the notion of "enumeration self-reducibility" introduced by Kimelfeld and Kolaitis [14] allows a characterization of the class $\mathsf{IncP}^{\Delta_k^P}$.

Definition 5 ([14], enumeration self-reducibility). *A binary relation R is* enumeration self-reducible *if* AnotherSol_$R \leq_T$ Exist-AnotherSol_R.

Proposition 6. *Let R be a binary relation, which is enumeration self-reducible, and $k \geq 0$. Then the following holds:* Exist-AnotherSol_$R \in \Delta_k^P$ *if and only if* Enum_$R \in \mathsf{IncP}^{\Delta_k^P}$.

We now prove that our classes provide strict hierarchies under the assumption that the polynomial hierarchy is strict.

Theorem 7. *Let $k \geq 0$. Then, unless the polynomial hierarchy collapses to the $(k+1)$-st level,*

$$\mathsf{DelayP}_p^{\Sigma_k^P} \subsetneq \mathsf{DelayP}_p^{\Sigma_{k+1}^P}, \mathsf{DelayP}^{\Sigma_k^P} \subsetneq \mathsf{DelayP}^{\Sigma_{k+1}^P} \text{ and } \mathsf{IncP}^{\Sigma_k^P} \subsetneq \mathsf{IncP}^{\Sigma_{k+1}^P}$$

Proof. Let $k \geq 0$, let L be a Σ_{k+1}^P-complete problem. Define a relation $R_L = \{(x, 1) \mid x \in L\}$. It is clear that Check_R_L is Σ_{k+1}^P-complete. Moreover, the enumeration problem Enum_R_L is in $\mathsf{DelayP}_p^{\Sigma_{k+1}^P}$ (thus also in $\mathsf{DelayP}^{\Sigma_{k+1}^P}$ and $\mathsf{IncP}^{\Sigma_{k+1}^P}$). Assume that Enum_$R_L \in \mathsf{DelayP}_p^{\Sigma_k^P}$ (or Enum_$R_L \in \mathsf{DelayP}^{\Sigma_k^P}$ or Enum_$R_L \in \mathsf{IncP}^{\Sigma_k^P}$). Then Check_$R_L$ can be decided in polynomial time using a Σ_k^P-oracle, meaning that Check_$R_L \in \Delta_{k+1}^P$ and thus the polynomial hierarchy collapses to the $(k+1)$-st level.

The following proposition states that the complexity classes based on DelayP_p and DelayP, respectively, are very likely to be distinct. We refer to the definition of the exponential hierarchy in [11]. We only recall here that $\Delta_{k+1}^{\mathsf{EXP}}$ denotes the class of decision problems decidable in exponential time with a Σ_k^P-oracle.

Proposition 8. *Let $k \geq 0$. If $\mathsf{EXP} \subsetneq \Delta_{k+1}^{\mathsf{EXP}}$, then $\mathsf{DelayP}_p^{\Sigma_k^P} \subsetneq \mathsf{DelayP}^{\Sigma_k^P} \not\subseteq \mathsf{DelayP}_p^{\Sigma_{k+1}^P}$.*

I.e., the lower computational power of DelayP_p compared with DelayP or IncP cannot be compensated by equipping the lower class with a slightly more powerful oracle. While complementing this result, we now also show that in contrast, the lower computational power of DelayP compared with IncP can be compensated by equipping the lower class with a slightly more powerful oracle.

Theorem 9. *Let $k \geq 0$. Then the following holds.*

1. $\mathsf{DelayP}_p^{\Sigma_{k+1}^P} \not\subseteq \mathsf{DelayP}^{\Sigma_k^P}$ *and* $\mathsf{DelayP}_p^{\Sigma_{k+1}^P} \not\subseteq \mathsf{IncP}^{\Sigma_k^P}$*, unless the polynomial hierarchy collapses to the $(k+1)$-st level.*
2. $\mathsf{DelayP}^{\Delta_{k+1}^P} = \mathsf{IncP}^{\Sigma_k^P}$.

Proof (Idea). The first claim follows from the proof of Theorem 7. For the second claim, the inclusion $\mathsf{DelayP}^{\Delta_{k+1}^P} \subseteq \mathsf{IncP}^{\Sigma_k^P}$ holds since the incremental delay with access to a Σ_k^P-oracle gives enough time to compute the answers of a Δ_{k+1}^P-oracle. To show that $\mathsf{DelayP}^{\Delta_{k+1}^P} \supseteq \mathsf{IncP}^{\Sigma_k^P}$, let $\textsc{Enum}\_R \in \mathsf{IncP}^{\Sigma_k^P}$ and $\mathcal{A}$ be a corresponding enumeration algorithm. We define a decision problem $\textsc{AnotherSolExt}_R^{\leq^*}$ that, on an input $y_1, \ldots, y_n, y', x \in \Sigma^*$, decides whether y' is the prefix of the $(n+1)$-st output of $\mathcal{A}(x)$. Since $\mathcal{A}$ witnesses the membership $\textsc{Enum}\_R \in \mathsf{IncP}^{\Sigma_k^P}$, it follows that $\textsc{AnotherSolExt}_R^{\leq^*} \in \Delta_{k+1}^P$, and using this language as an oracle, we have that $\textsc{Enum}\_R \in \mathsf{DelayP}^{\Delta_{k+1}^P}$.

Concerning the effect of the allowed input size to the oracles, observe that it follows immediately that $\mathsf{DelayP}^{\Delta_{k+1}^P} \neq \mathsf{DelayP}^{\Sigma_k^P}$, but $\mathsf{DelayP}_p^{\Delta_{k+1}^P} = \mathsf{DelayP}_p^{\Sigma_k^P}$.

4 Declarative-style Reductions

As far as we know, only a few kinds of reductions between enumeration problems have been investigated so far. One such reduction is implicitely described in [7]. It establishes a bijection between sets of solutions. A different approach introduced in [3] relaxes this condition and allows non-bijective reduction functions. We go further in that direction in proposing a declarative style reduction relaxing the isomorphism requirement while closing the relevant enumeration classes.

Definition 10 (reduction $\leq_e$). *Let $R_1, R_2 \subseteq \Sigma^*$ be binary relations. Then we define $\textsc{Enum}\_R_1 \leq_e \textsc{Enum}\_R_2$ if there exist a function $\sigma : \Sigma^* \to \Sigma^*$ computable in polynomial time and a relation $\tau \subseteq \Sigma^* \times \Sigma^* \times \Sigma^*$, s.t. for all $x \in \Sigma^*$ the following holds. For $y \in \Sigma^*$, let $\tau(x, y, -) := \{z \in \Sigma^* \mid (x, y, z) \in \tau\}$ and for $z \in \Sigma^*$, let $\tau(x, -, z) := \{y \in \Sigma^* \mid (x, y, z) \in \tau\}$. Then:*

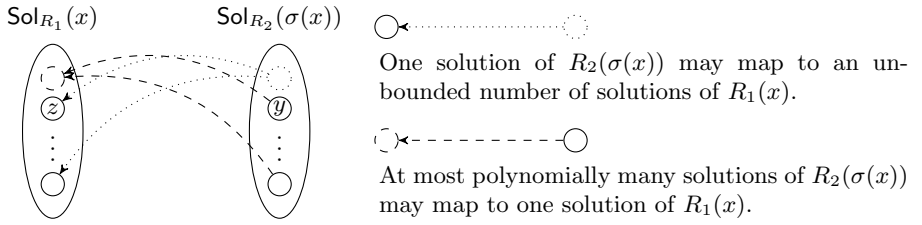

Fig. 1. Illustration of relation τ from Definition 10.

1. $\mathsf{Sol}_{R_1}(x) = \bigcup_{y \in \mathsf{Sol}_{R_2}(\sigma(x))} \tau(x, y, -)$;
2. $\forall y \in \mathsf{Sol}_{R_2}(\sigma(x))$, *we have* $\emptyset \subsetneq \tau(x, y, -) \subseteq \mathsf{Sol}_{R_1}(x)$ *and* $\tau(x, y, -)$ *can be enumerated with polynomial delay in* $|x|$;
3. $\forall z \in \mathsf{Sol}_{R_1}(x)$, *we have* $\tau(x, -, z) \subseteq \mathsf{Sol}_{R_2}(\sigma(x))$ *and the size of* $\tau(x, -, z)$ *is polynomially bounded in* $|x|$.

Intuitively, τ establishes a relationship between instances x, solutions $y \in \mathsf{Sol}_{R_2}(\sigma(x))$ and solutions $z \in \mathsf{Sol}_{R_1}(x)$. We can thus use τ to design an enumeration algorithm for $\mathsf{Sol}_{R_1}(x)$ via an enumeration algorithm for $\mathsf{Sol}_{R_2}(\sigma(x))$. The conditions imposed on τ have the following meaning: By condition 1, the solutions $\mathsf{Sol}_{R_1}(x)$ can be computed by iterating through the solutions $y \in \mathsf{Sol}_{R_2}(\sigma(x))$ and computing $\tau(x, y, -) \subseteq \mathsf{Sol}_{R_1}(x)$. Conditions 2 and 3 make sure that the delay of enumerating $\mathsf{Sol}_{R_1}(x)$ only differs by a polynomial from the delay of enumerating $\mathsf{Sol}_{R_2}(\sigma(x))$: condition 2 ensures that, for every y, the set $\tau(x, y, -)$ can be enumerated with polynomial delay and that we never encounter a "useless" y (i.e., a solution $y \in \mathsf{Sol}_{R_2}(\sigma(x))$ which is associated with no solution $z \in \mathsf{Sol}_{R_1}(x)$). In principle, we may thus get duplicates z associated with different values of y. However, condition 3 ensures that each z can be associated with at most polynomially many values y. Using a priority queue storing all z that are output, we can avoid duplicates, c.f. the proof of Proposition 12 or [17]. Figure 1 illustrates τ.

Example 11. The idea of the relation τ can also be nicely demonstrated on an $\leq_e$ reduction from 3-Colourability^e to 4-Colourability^e (enumerating all valid 3- respectively 4-colourings of a graph). We intentionally choose this reduction since there is no bijection between the solutions of the two problems.

Recall the classical many-one reduction between these problems, which takes a graph G and defines a new graph G' by adding an auxiliary vertex v and connecting it to all the other ones. This reduction can be extended to a $\leq_e$ reduction with the following relation τ: With every graph G in the first component of τ, we associate all valid 4-colourings (using 0, 1, 2, and 3) of G' in the third component of τ. With each of those we associate the corresponding 3-colouring of G in the second component. They are obtained from the 4-colourings by first making sure that v is coloured with 3 (by "switching" the colour of v with 3) and then by simply reading off the colouring of the remaining vertices.

The reductions $\leq_e$ have two desirable important properties, as stated next.

Proposition 12. *Let $\mathcal{C} \in \{\Sigma_k^P, \Delta_k^P \mid k \geq 0\}$. The classes $\mathsf{DelayP}_p^{\mathcal{C}}$, $\mathsf{DelayP}^{\mathcal{C}}$, and $\mathsf{IncP}^{\mathcal{C}}$ are closed under $\leq_e$. In addition, the reductions $\leq_e$ are transitive.*

Nevertheless their main drawback is that it is very unlikely that completeness results under $\leq_e$ reductions can be obtained, since even the most natural problems are not complete under such a reduction.

Proposition 13. *Let $k \geq 1$. The problem $\Sigma_k\mathrm{SAT}^e$ is not complete for $\mathsf{DelayP}_p^{\Sigma_k^P}$ under $\leq_e$ reductions unless the polynomial hierarchy collapses to the k^{th} level.*

5 Procedural-style Reductions and Completeness Results

Although Turing reductions are too strong to show completeness results for classes in the polynomial hierarchy, Turing style reductions turn out to be meaningful in our case. In this section we introduce two types of reductions that are motivated by Turing reductions. Both of them are able to reduce between enumeration problems for which the reduction $\leq_e$ seems to be too weak.

Towards this goal, we first have to define the concept of RAMs with an *oracle for enumeration problems*. The intuition behind the definition of such enumeration oracle machines is the following: For algorithms (i.e., Turing machines or RAMs in the case of enumeration) using a decision oracle for the language L, we usually have a special instruction that given an input x decides in one step whether $x \in L$, and then executes the next step of the algorithm accordingly. For an algorithm $\mathcal{A}$ using an enumeration oracle, an input x to some ENUM_R-oracle returns in a single step (using the instruction NOO, see the definition below) a single element of $\mathsf{Sol}_R(x)$, and then $\mathcal{A}$ can proceed according to this output.

Definition 14 (Enumeration Oracle Machines). *Let* ENUM_R *be an enumeration problem. An* Enumeration Oracle Machine with an enumeration oracle ENUM_R (EOM_R) *is a RAM with a sequence of new registers $O^e(0), O^e(1), \ldots$ and a new instruction* NOO *(next Oracle output). An EOM_R is* oracle-bounded *if the size of all inputs to the oracle is at most polynomial in the size of the input to the EOM_R.*

When executing NOO, the machine writes – in one step – some $y_i \in \mathsf{Sol}_R(x)$ to the accumulator A, where x is the word stored in $O^e(0), O^e(1), \ldots$ and y_i is defined as follows:

Definition 15 (Next Oracle Output). *Let R be a binary relation, $\pi_1, \pi_2, \ldots$ be the run of an EOM_R and assume that the k^{th} instruction is* NOO*, i.e., $\pi_k =$ NOO. Denote with x_i the word stored in $O^e(0), O^e(1), \ldots$ at step i. Let $K = \{\pi_i \in \{\pi_1, \ldots, \pi_{k-1}\} \mid \pi_i =$ NOO *and* $x_i = x_k\}$. Then the* oracle output y_k in π_k *is defined as an arbitrary $y_k \in \mathsf{Sol}_R(x_k)$ s.t. y_k has not been the oracle output in any $\pi_i \in K$. If no such y_k exists, then the oracle output in π_k is undefined.*

When executing NOO *in step* π_k*, if the oracle output* y_k *is undefined, then the accumulator* A *contains some special symbol in step* π_{k+1}*. Otherwise in step* π_{k+1} *the accumulator* A *contains* y_k *.*

Observe that since an EOM M^e is a polynomially bounded RAM and the complete oracle output is stored in the accumulator A, only such oracle calls are allowed where the size of each oracle output is guaranteed to be polynomially in the size of the input of M^e.

Using EOMs, we can now define another type of reductions among enumerations problems, reminiscent of classical Turing reductions. I.e., we say that one problem ENUM_R_1 reduces to another problem ENUM_R_2 if ENUM_R_1 can be solved by an EOM using ENUM_R_2 as an enumeration oracle.

Definition 16 (Reductions $\leq_D$, $\leq_I$). *Let* R_1 *and* R_2 *be binary relations.*

- *We say that* ENUM_R_1 $\leq_D$ ENUM_R_2 *if there is an oracle-bounded EOM_*R_2 *that enumerates* R_1 *in* DelayP *and is independent of the order in which the* ENUM_R_2 *oracle enumerates its answers.*
- *We say that* ENUM_R_1 $\leq_I$ ENUM_R_2 *if there is an EOM_*R_2 *that enumerates* R_1 *in* IncP *and is independent of the order in which the* ENUM_R_2 *oracle enumerates its answers.*

For $\leq_D$, we required the EOM_R_2 to be oracle-bounded. We would like to point out that this restriction is essential: if we drop it, then the classes $\mathsf{DelayP}^{\mathcal{C}}$ are not closed under the resulting reduction. They are, however, closed under the reductions as defined above.

Proposition 17. *Let* $\mathcal{C} \in \{\Sigma_k^P, \Delta_k^P \mid k \geq 0\}$*. The classes* $\mathsf{DelayP}^{\mathcal{C}}$ *and* $\mathsf{DelayP}_p^{\mathcal{C}}$ *are closed under* $\leq_D$*. The classes* $\mathsf{IncP}^{\mathcal{C}}$ *are closed under* $\leq_I$*.*

We note that all of these properties still hold when there is no oracle at all, i.e., for the classes DelayP and IncP.

Proposition 18. *The reductions* $\leq_D$ *and* $\leq_I$ *are transitive.*

Now, unlike for $\leq_e$, the next theorem shows that the reductions $\leq_D$ and $\leq_I$ induce complete problems for the enumeration complexity classes introduced in Section 3.

Theorem 19. *Let* R *be a binary relation and* $k \geq 1$ *such that* EXIST_R *is* Σ_k^P*-complete.*

- ENUM_R *is* $\mathsf{DelayP}_p^{\Sigma_k^P}$*-hard under* $\leq_D$ *reductions.*
- ENUM_R *is* $\mathsf{IncP}^{\Sigma_k^P}$*-hard under* $\leq_I$ *reductions.*
- *If* R *is self-reducible, then* ENUM_R *is* $\mathsf{DelayP}_p^{\Sigma_k^P}$*-complete under* $\leq_D$ *reductions and* $\mathsf{IncP}^{\Sigma_k^P}$*-complete under* $\leq_I$ *reductions.*

Proof (Idea). Let ENUM_R' $\in$ DelayP_p^L for some $L \in \Sigma_k^P$, and assume that z is the input to an L-oracle when enumerating $\mathsf{Sol}_{R'}(x)$ for some $x \in \Sigma^*$. As EXIST_R is Σ_k^P-complete and the enumeration is oracle-bounded, z can be transformed to an equivalent instance z' of EXIST_R in time polynomial only in $|x|$. Therefore by calling the ENUM_R-oracle once and by checking whether $\mathsf{Sol}_R(z') = \emptyset$, one can decide whether $z \in L$. The membership ENUM_$R \in \mathsf{DelayP}_p^{\Sigma_k^P}$ in the case of self-reducibility follows by Proposition 2.

As a consequence, the enumeration problems $\Sigma_k\mathrm{SAT}^e$ and also $\Pi_k\mathrm{SAT}^e$ are natural complete problems for our enumeration complextiy classes:

Corollary 20. *Let $k \geq 0$. Then*

1. *$\Sigma_{k+1}\mathrm{SAT}^e$ is complete for $\mathsf{DelayP}_p^{\Sigma_{k+1}^P}$ under $\leq_D$ reductions.*
2. *$\Pi_k\mathrm{SAT}^e$ and $\Sigma_{k+1}\mathrm{SAT}^e$ are complete for $\mathsf{IncP}^{\Sigma_{k+1}^P}$ under $\leq_I$ reductions.*

Observe that, under different reductions, $\Sigma_k\mathrm{SAT}^e$ is complete for both, $\mathsf{IncP}^{\Sigma_k^P}$ and for the presumably smaller class $\mathsf{DelayP}_p^{\Sigma_k^P}$. This provides additional evidence that the two reductions nicely capture $\mathsf{IncP}^{\Sigma_k^P}$ and $\mathsf{DelayP}_p^{\Sigma_k^P}$, respectively. Also from Corollary 20 it follows as a special case that $\mathsf{IncP}^{\Sigma_0^P}$ and $\mathsf{IncP}^{\Sigma_1^P}$ are equivalent under $\leq_I$ reductions: Clearly, $\Sigma_0\mathrm{SAT}^e = \Pi_0\mathrm{SAT}^e$, since in both cases the formulas are quantifier free and one asks for all satisfying truth assignments. Now by the theorem we know that both, $\Sigma_1\mathrm{SAT}^e$ and $\Pi_0\mathrm{SAT}^e$, and thus also $\Sigma_0\mathrm{SAT}^e$, are complete for $\mathsf{IncP}^{\Sigma_1^P}$. As a result we have that the enumeration variant of the traditional SAT problem is $\mathsf{IncP}^{\mathsf{NP}}$-complete.

Roughly speaking Theorem 19 says that any self-reducible enumeration problem whose corresponding decision problem is hard, is hard as well. An interesting question is whether there exist easy decision problems for which the corresponding enumeration problem is hard. We answer positively to this question in revisiting, in our framework, a classification theorem obtained for the enumeration of generalized satisfiability [4]. It is convenient to first introduce some notation.

A *logical relation* of arity k is a relation $R \subseteq \{0,1\}^k$. A *constraint*, C, is a formula $C = R(x_1, \ldots, x_k)$, where R is a logical relation of arity k and the x_i's are variables. An assignment m of truth values to the variables *satisfies* the constraint C if $\big(m(x_1), \ldots, m(x_k)\big) \in R$. A *constraint language* Γ is a finite set of nontrivial logical relations. A *Γ-formula* ϕ is a conjunction of constraints using only logical relations from Γ. A Γ-formula ϕ is satisfied by an assignment $m : \mathrm{var}(\phi) \to \{0,1\}$ if m satisfies all constraints in ϕ.

Throughout the text we refer to different types of Boolean relations following Schaefer's terminology, see [16, 4]. We say that a constraint language is *Schaefer* if every relation in Γ is either Horn, dualHorn, bijunctive, or affine.

$\mathrm{SAT}(\Gamma)^e$
INSTANCE: ϕ a Γ-formula
OUTPUT: all satisfying assignments of ϕ.

The following theorem gives the complexity of this problem according to Γ.

Theorem 21. *Let Γ be a finite constraint language. If Γ is Schaefer, then* $\mathrm{SAT}(\Gamma)^e$ *is in* DelayP*, otherwise it is* $\mathsf{DelayP}_p^{\mathsf{NP}}$*-complete under $\leq_D$ reductions.*

Proof. The polynomial cases were studied in [4]. Let us now consider the case where Γ is not Schaefer. Membership of $\mathrm{SAT}(\Gamma)^e$ in $\mathsf{DelayP}_p^{\mathsf{NP}}$ is clear. For the hardness, let us introduce T and F as the two unary constant relations $\mathrm{T} = \{1\}$ and $\mathrm{F} = \{0\}$. According to Schaefer's dichotomy theorem [16], deciding whether a $\Gamma \cup \{\mathrm{F}, \mathrm{T}\}$-formula is satisfiable is NP-complete. Since this problem is self-reducible, according to Theorem 19, $\mathrm{SAT}(\cup\{\mathrm{F}, \mathrm{T}\})^e$ is $\mathsf{DelayP}_p^{\mathsf{NP}}$-complete under $\leq_D$ reductions. From the proof given in [4] it is easy to see that if Γ is not Schaefer, then $\mathrm{SAT}(\cup\{\mathrm{F}, \mathrm{T}\})^e \leq_D \mathrm{SAT}(\Gamma)^e$, thus concluding the proof.

To come back to the above discussion, we point out that there exist constraint languages Γ such that the decision problem $\mathrm{SAT}(\Gamma)$ is in P, while the enumeration problem $\mathrm{SAT}(\Gamma)^e$ is $\mathsf{DelayP}_p^{\mathsf{NP}}$-complete, namely 0-valid or 1-valid constraint languages that are not Schaefer.

A rather surprising completeness result is the following.

Proposition 22. *Let* $\mathrm{CIRCUMSCRIPTION}^e$ *denote the problem of enumerating all subset minimal models of a boolean formula. Then* $\mathrm{CIRCUMSCRIPTION}^e$ *is* $\mathsf{IncP}^{\mathsf{NP}}$*-complete under $\leq_I$ reductions.*

What makes this result surprising is the discrepancy from the behaviour of the counting variant of the problem: The counting variant of $\mathrm{CIRCUMSCRIPTION}^e$ is a prototypical $\# \cdot \mathsf{coNP}$-complete problem [8], and thus of the same hardness as the counting variant of $\Pi_1\mathrm{SAT}^e$. However, for enumeration we have that $\mathrm{CIRCUMSCRIPTION}^e$ shows the same complexity as $\Sigma_1\mathrm{SAT}^e$, which is considered to be lower than that of $\Pi_1\mathrm{SAT}^e$.

Observe that $\mathrm{CIRCUMSCRIPTION}^e$ is very unlikely to be self-reducible: In fact, the problem of deciding if a partial truth assignment can be extended to a subset minimal model is Σ_2^P-complete [10], while deciding the existence of a minimal model is clearly NP-complete. Thus $\mathrm{CIRCUMSCRIPTION}^e$ is not self-reducible unless the polynomial hierarchy collapses to the first level.

6 Conclusion

We introduced a hierarchy of enumeration complexity classes, extending the well-known tractable enumeration classes DelayP and IncP, just as the Δ_k^P-classes of the polynomial-time hierarchy extend the class P. We show that under reasonable complexity assumptions these hierarchies are strict. We introduced a type of reduction among enumeration problems under which the classes in our hierarchies are closed and which allow to exhibit complete problems. For well-studied problems like Boolean CSPs in the Schaefer framework or circumscription, we obtain completeness results for the associated enumeration problems. Up to now, lower bounds for enumeration problems were only of the form "$\textsc{Enum}\_R$ is not in DelayP (or IncP) unless $\mathsf{P} \neq \mathsf{NP}$". Our work provides a framework which allows us to pinpoint the complexity of such problems in a better way in terms of completeness.

Acknowledgments

This work was supported by the Vienna Science and Technology Fund (WWTF) through project ICT12-015, the Austrian Science Fund (FWF): P25207-N23, P25518-N23, I836-N23, W1255-N23 and the French Agence Nationale de la Recherche, AGGREG project reference ANR-14-CE25-0017-01.

References

1. Arora, S., Barak, B.: Computational Complexity: A Modern Approach. Cambridge University Press, Cambridge (2009)
2. Bagan, G., Durand, A., Grandjean, E.: On acyclic conjunctive queries and constant delay enumeration. In: Computer Science Logic. pp. 208–222. Springer (2007)
3. Brault-Baron, J.: De la pertinence de l'énumération : complexité en logiques propositionnelle et du premier ordre. (The relevance of the list: propositional logic and complexity of the first order). Ph.D. thesis, University of Caen Normandy, France (2013), https://tel.archives-ouvertes.fr/tel-01081392
4. Creignou, N., Hébrard, J.J.: On generating all solutions of generalized satisfiability problems. Informatique théorique et applications 31(6), 499–511 (1997)
5. Creignou, N., Kröll, M., Pichler, R., Skritek, S., Vollmer, H.: On the complexity of hard enumeration problems. CoRR abs/1610.05493 (2016), http://arxiv.org/abs/1610.05493, available at http://arxiv.org/abs/1610.05493
6. Creignou, N., Vollmer, H.: Parameterized complexity of weighted satisfiability problems: Decision, enumeration, counting. Fundam. Inform. 136(4), 297–316 (2015), http://dx.doi.org/10.3233/FI-2015-1159
7. Durand, A., Grandjean, E.: First-order queries on structures of bounded degree are computable with constant delay. ACM Trans. Comput. Log. 8(4) (2007), http://doi.acm.org/10.1145/1276920.1276923
8. Durand, A., Hermann, M., Kolaitis, P.G.: Subtractive reductions and complete problems for counting complexity classes. Theor. Comput. Sci. 340(3), 496–513 (2005)
9. Durand, A., Schweikardt, N., Segoufin, L.: Enumerating answers to first-order queries over databases of low degree. In: Proc. PODS 2014. pp. 121–131. ACM (2014)
10. Eiter, T., Gottlob, G.: The complexity of logic-based abduction. J. ACM 42(1), 3–42 (1995)
11. Hemachandra, L.A.: The strong exponential hierarchy collapses. In: Proc. STOC 1987. pp. 110–122. ACM (1987)
12. Hemaspaandra, L.A., Vollmer, H.: The satanic notations: counting classes beyond #P and other definitional adventures. SIGACT News 26(1), 2–13 (1995)
13. Johnson, D.S., Papadimitriou, C.H., Yannakakis, M.: On generating all maximal independent sets. Inf. Process. Lett. 27(3), 119–123 (1988)
14. Kimelfeld, B., Kolaitis, P.G.: The complexity of mining maximal frequent subgraphs. ACM Trans. Database Syst. 39(4), 32:1–32:33 (2014), http://doi.acm.org/10.1145/2629550
15. Lucchesi, C.L., Osborn, S.L.: Candidate keys for relations. J. Comput. Syst. Sci. 17(2), 270–279 (1978)
16. Schaefer, T.J.: The complexity of satisfiability problems. In: Proc. STOC 1978. pp. 216–226. ACM Press (1978)

17. Strozecki, Y.: Enumeration complexity and matroid decomposition. Ph.D. thesis, Universite Paris Diderot – Paris 7 (Dec 2010), available at `http://www.prism.uvsq.fr/~ystr/these_strozecki`
18. Strozecki, Y.: On enumerating monomials and other combinatorial structures by polynomial interpolation. Theory Comput. Syst. 53(4), 532–568 (2013)
19. Toda, S.: PP is as hard as the polynomial-time hierarchy. SIAM J. Comput. 20(5), 865–877 (1991)