

A Complexity Theory for Hard Enumeration Problems

Nadia Creignou, Markus Kröll, Reinhard Pichler, Sebastian Skritek, Heribert Vollmer

▶ **To cite this version:**

Nadia Creignou, Markus Kröll, Reinhard Pichler, Sebastian Skritek, Heribert Vollmer. A Complexity Theory for Hard Enumeration Problems. Discrete Applied Mathematics, 2019, 268, pp.191-209. 10.1016/j.dam.2019.02.025 . hal-02439418

HAL Id: hal-02439418
https://hal.science/hal-02439418

Submitted on 14 Jan 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

A Complexity Theory for Hard Enumeration Problems

Nadia Creignou[b], Markus Kröll[a], Reinhard Pichler[a], Sebastian Skritek[a], Heribert Vollmer[c]

[a] *TU Wien, Vienna, Austria*
[b] *Aix Marseille Univ, Université de Toulon, CNRS, LIS, Marseille, France*
[c] *Leibniz Universität Hannover, Hannover, Germany*

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 [1]. 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 consecutive 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 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 [2], 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 given in [3, 4, 5, 6, 7, 8]

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

Email addresses: nadia.creignou@univ-amu.fr (Nadia Creignou), kroell@dbai.tuwien.ac.at (Markus Kröll), pichler@dbai.tuwien.ac.at (Reinhard Pichler), skritek@dbai.tuwien.ac.at (Sebastian Skritek), vollmer@thi.uni-hannover.de (Heribert Vollmer)

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:

E-Π_kSAT / E-Σ_kSAT	
Instance:	$\psi = \forall\vec{y}_1\exists\vec{y}_2 \ldots Q_k\vec{y}_k\phi(\vec{x},\vec{y})$ / $\psi = \exists\vec{y}_1\forall\vec{y}_2 \ldots Q_k\vec{y}_k\phi(\vec{x},\vec{y})$
Output:	All assignments for $\vec{x}$ 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 $\mathcal{C} \in \{\mathsf{P}, \mathsf{coNP}, \Pi_2^P, \ldots\}$ [9, 10]. 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 $\mathsf{Del}\mathcal{C}$, $\mathsf{Inc}\mathcal{C}$, 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.

- *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 type of reductions.
- *Procedural-style reductions.* In Section 5, we introduce a procedural-style notion of reductions, show that they have the desirable properties any reduction should possess, and show that the enumeration problems associated with the typical quantified Boolean satisfiability problems are complete in our hierarchy of enumeration problems.
- *Completeness results for natural problems.* Starting with these completeness results, we establish a chain of reductions among several natural enumeration problems from areas such as generalized satisfiability, circumscription, model-based diagnosis, abduction, and repairs of inconsistent databases in Section 6, thus proving completeness of these problems in classes of our hierarchy.

This work is an extension of the conference article [11].

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 $(x, y) \in R$, $|y| \leq p(|x|)$ holds. 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 such a relation R, we can associate several natural computational problems:

EXIST_R Instance: $x \in \Sigma^*$ Question: Is $R(x) \neq \emptyset$?
EXIST-ANOTHERSOL_R Instance: $x \in \Sigma^*, Y \subseteq R(x)$ Output: Is $(R(x) \setminus Y) \neq \emptyset$?
ANOTHERSOL_R Instance: $x \in \Sigma^*, Y \subseteq R(x)$ Output: $y \in R(x) \setminus Y$ 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 $y' \in \Sigma^*$ such that $(x, yy') \in R$?

In counting complexity, the following computational problem arises:

COUNT_R Instance: $x \in \Sigma^*$ Output: $\big\lvert R(x) \big\rvert$.

A binary relation R also gives rise to an enumeration problem, which aims at outputting the set $R(x)$ of solutions for x.

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 $k \in \{0, 1, \dots\}$.

As we usually deal with exponential runtime of algorithms in the course of an enumeration process, we will also make use of the weak EXP hierarchy (see [12]) in the following. It is an exponential time analogue of the polynomial hierarchy defined as follows:

Definition 1 (Exponential Hierarchy).

$$\Delta_0^{\mathsf{EXP}} = \Sigma_0^{\mathsf{EXP}} = \Pi_0^{\mathsf{EXP}} = \mathsf{EXP}$$
$$\Delta_k^{\mathsf{EXP}} = \mathsf{EXP}^{\Sigma_{k-1}^P}, \text{ for } k \geq 1$$
$$\Sigma_k^{\mathsf{EXP}} = \mathsf{NEXP}^{\Sigma_{k-1}^P}, \text{ for } k \geq 1$$
$$\Pi_k^{\mathsf{EXP}} = \mathsf{coNEXP}^{\Sigma_{k-1}^P}, \text{ for } k \geq 1$$

A hierarchy of counting complexity classes similar to the polynomial hierarchy was defined [9, 10]. Let $\mathcal{C}$ be a complexity class of decision problems. Then $\# \cdot \mathcal{C}$ denotes the class of all counting problems $\mathcal{P}$, such that there exists a relation R with $\textsc{Check}_R \in \mathcal{C}$ and $\mathcal{P} = \textsc{Count}_R$. For $\#\mathsf{P}$ – the best known counting complexity class – we have $\#\mathsf{P} = \# \cdot \mathsf{P}$. Moreover, the inclusions $\#\mathsf{P} \subseteq \# \cdot \mathsf{NP} \subseteq \# \cdot \mathsf{coNP} \subseteq \# \cdot \Sigma_2^P \subseteq \# \cdot \Pi_2^P \subseteq \ldots$ hold.

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 $R_1 \leq_r R_2$ imply $R_1 \in \mathcal{C}$. 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 for Enumeration

In Section 1, we have already recalled two important tractable enumeration complexity classes, DelayP and IncP from [1]. Note that in [2, 13], these classes are defined slightly differently by restricting DelayP and IncP to those problems $\textsc{Enum}_R$ such that the corresponding problem $\textsc{Check}_R$ is in P — in addition to the constraints on the allowed delays. We adhere to the definition of tractable enumeration classes from [1].

In contrast to counting complexity, defining a hierarchy of enumeration problems via the $\textsc{Check}_R$ problem of binary relations R does not seem appropriate. Note that while the same counting problem $\mathcal{P}$ can be defined by different relations, i.e., $\mathcal{P} = \textsc{Count}_R_1$ and $\mathcal{P} = \textsc{Count}_R_2$ for $R_1 \neq R_2$, the relation associated with an enumeration problem is fixed.

Thus, we need an alternative approach for defining meaningful enumeration complexity classes. To this end, we first fix our computation model: Observe that there is a subtle difference between how Turing Machines and random access machines (RAMs) can access data in their "memory". Due to the linear nature of the tapes of Turing Machines, accessing an exponential size data structure requires exponential time, even if just a small portion of its data is actually read. This is usually not a problem when studying decision or search problems, since writing an exponentially sized data structure onto the tape already requires exponential time, and as a result the additional exponential time to read the data structure has no more effect on the overall runtime.

This situation, however, changes for enumeration problems and polynomial delay, since it is now possible that, while computing an exponential number of solutions, an exponential size data structure accumulates. Thus computing another solution with only polynomial delay is not possible with Turing Machines if this requires some information from such a data structure. However, by maintaining suitable index structures, it might be possible to gather the necessary information in polynomial time from such a data structure on a random access machine (RAM) that allows to access its memory directly and not first needs to move the read/write head over the correct tape position. In fact, due to this property that allows to access (polynomially sized) parts of exponential size data in polynomial time, it is common to use the RAM model as a computational model for the study of enumeration problems (cf. [2]). We restrict ourselves here to polynomially bounded RAMs, 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 RAMs with an output-instruction, as defined in [2]. 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 [14], 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 [2]. 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.

Before introducing enumeration complexity classes with oracles, we recall the definitions of precomputation and postcompuation time, see [15]. Given a binary relation R, an instance x and a RAM M that enumerates $R(x)$, the precomputation time is the time between the start of the RAM M and the first output of $R(x)$, and the postcomputation time is the time between the last output of $R(x)$ and the termination of M. It is convenient to think of these time bounds in terms of terms of additional solutions of $R(x)$: As a first step in computing $R(x)$, M outputs a 0-th solution, and after the termination of M, a $(|R(x)|+1)$-th solution is output. Thus the pre- and postcompuation time correspond to the delay between outputting the $0-th$ and first solution, respectively outputting the $|R(x)|$-th and $(|R(x)|+1)$-th solution.

Definition 2 (enumeration complexity classes). Let R be a binary relation, and $\mathcal{C}$ a decision complexity class. Then we say that:

- ENUM_$R \in \mathsf{Del}^{+}\mathcal{C}$ if there is a RAM M with oracle L in $\mathcal{C}$ and a polynomial p, such that for any instance x, M enumerates $R(x)$ with delay $p(|x|)$ between the i-th and $(i+1)$-th solution for every $0 \leq i \leq |R(x)|$.
- ENUM_$R \in \mathsf{Inc}^{+}\mathcal{C}$ if there is a RAM M with oracle L in $\mathcal{C}$ and a polynomial p, such that for any instance x, M enumerates $R(x)$ with delay $p(|x|+i)$ between the i-th and $(i+1)$-th solution for every $0 \leq i \leq |R(x)|$.
- ENUM_$R \in \mathsf{Del}\mathcal{C}$ if there is a RAM M with oracle L in $\mathcal{C}$ and a polynomial p, such that for any instance x, M enumerates $R(x)$ with delay $p(|x|)$ between the i-th and $(i+1)$-th solution for every $0 \leq i \leq |R(x)|$. Moreover, the size of every oracle call is bound by $p(|x|)$.
- ENUM_$R \in \mathsf{Inc}\mathcal{C}$ if there is a RAM M with oracle L in $\mathcal{C}$ and a polynomial p, such that for any instance x, M enumerates $R(x)$ with delay $p(|x|+i)$ between the i-th and $(i+1)$-th solution for every $0 \leq i \leq |R(x)|$. Moreover, the size of every oracle call made between outputting the i-th and $(i+1)$-th solution is bound by $p(|x|+i)$ for every $0 \leq i \leq |R(x)|$.

It is easy to see that by taking $\mathcal{C} = \mathsf{P}$, the definitions of $\mathsf{Del}\mathcal{C}$ and $\mathsf{Inc}\mathcal{C}$ are equivalent to the definitions of the classes DelayP respectively IncP. Note that the restriction of the oracle inputs to polynomial size makes a crucial difference when it comes to $\mathsf{Del}\mathcal{C}$, 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 $\mathsf{Inc}\mathcal{C}$, 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:

Proposition 1. *Let $\mathcal{C}$ be a decision complexity class. Then* $\mathsf{Inc}\mathcal{C} = \mathsf{Inc}^{+}\mathcal{C}$

Proof. The inclusion $\mathsf{Inc}\mathcal{C} \subseteq \mathsf{Inc}^{+}\mathcal{C}$ is given by definition, thus we need to show that $\mathsf{Inc}^{+}\mathcal{C} \subseteq \mathsf{Inc}\mathcal{C}$. Let R be a binary relation with ENUM_$R \in \mathsf{Inc}^{+}\mathcal{C}$, x an instance to the enumeration problem and let M be a RAM with oracle $\mathcal{C}$ that outputs $R(x)$ with incremental polynomial delay $\mathcal{O}((|x|+i)^k)$ between the i-th and $(i+1)$-th

solution for every $0 \leq i \leq |R(x)|$. The input to the $\mathcal{C}$-oracle is of size at most $\mathcal{O}((|x|+i)^{k+1})$ between the i-th and $(i+1)$-th solution and thus also polynomially bounded. □

In fact, the lower computational power of $\mathsf{Del}\mathcal{C}$ compared with $\mathsf{Inc}\mathcal{C}$ can be compensated by equipping the lower class with a slightly more powerful oracle.

Theorem 2. *Let $k \geq 0$. Then $\mathsf{Del}^{+}\Delta_{k+1}^{P} = \mathsf{Inc}\Sigma_{k}^{P}$.*

Proof. Let $k \geq 0$. We start the proof by showing that $\mathsf{Del}^{+}\Delta_{k+1}^{P} \supseteq \mathsf{Inc}\Sigma_{k}^{P}$. So let $\textsc{Enum}_R \in \mathsf{Inc}\Sigma_{k}^{P}$ with corresponding binary relation R and let $x \in \Sigma^{*}$. Fix an incremental delay algorithm $\mathcal{A}$ which uses a Σ_{k}^{P}-oracle witnessing the membership $\textsc{Enum}_R \in \mathsf{Inc}\Sigma_{k}^{P}$ and let $<^{*}$ be an order on $R(x)$ induced by algorithm $\mathcal{A}$, i.e. the i-th output of $\mathcal{A}$ on input x is the i-th element in $<^{*}$. We define the following decision problem:

$\textsc{AnotherSolExt}_{R}^{\mathcal{A}}$	
Instance:	$y_1, \ldots, y_n, y', x \in \Sigma^{*}$
Question:	Is y' a prefix of y_{n+1}, where y_{n+1} is the $(n+1)$-th element in $R(x)$ w.r.t. $<^{*}$?

We first note that $\textsc{AnotherSolExt}_{R}^{\mathcal{A}} \in \Delta_{k+1}^{P}$. Indeed, assume that we have given an instance $y_1, \ldots, y_n, y', x \in \Sigma^{*}$. Then we can use $\mathcal{A}$ to enumerate the first $n+1$ elements of $R(x)$ in time $\mathcal{O}(\text{poly}(|x|, |n+1|)) = \mathcal{O}(\text{poly}(|y_1|+\ldots+|y_n|+|y'|+|x|))$ and then check whether y' is a prefix of y_{n+1}. As $\mathcal{A}$ uses a Σ_{k}^{P}-oracle, this decision can be made within $\mathsf{P}^{\Sigma_{k}^{P}} = \Delta_{k+1}^{P}$. The membership $\textsc{Enum}_R \in \mathsf{Del}^{+}\Delta_{k+1}^{P}$ follows immediately, as we can construct a polynomial delay algorithm with an $\textsc{AnotherSolExt}_{R}^{\mathcal{A}}$-oracle that enumerates $\textsc{Enum}_R$ in a similar way to commonly used enumeration algorithms, see [2, 5] or Proposition 6. This enumeration algorithm starts by computing some $a \in \Sigma$ such that the oracle on this input returns 'yes'. Then y_1 (and also every other y_i) can be computed by repeatedly extending the previous input to the oracle by the unique element in Σ such that the decision oracle $\textsc{AnotherSolExt}_{R}^{\mathcal{A}}$ returns 'yes'.
Next we need to show that $\mathsf{Del}^{+}\Delta_{k+1}^{P} \subseteq \mathsf{Inc}\Sigma_{k}^{P}$, so let $\textsc{Enum}_R \in \mathsf{Del}^{+}\Delta_{k+1}^{P}$, and let $\mathcal{A}$ be an algorithm witnessing this membership. Moreover, let $L \in \mathsf{P}^{\Sigma_{k}^{P}}$ be the language used for the Δ_{k+1}^{P}-oracle in $\mathcal{A}$, with a polynomial q and a language $L' \in \Sigma_{k}^{P}$ such that $L \in \mathsf{DTIME}^{L'}(q(n))$. Let p be be the polynomial for the delay of $\mathcal{A}$. We describe an algorithm $\mathcal{B}$ that has an incremental delay of $p(X) \cdot Y \cdot q(p(X) \cdot Y)$ (here the indeterminate X stands for the size of the input and the indeterminate Y for the number of previously output solutions) that uses a Σ_{k}^{P}-oracle, such that $\mathcal{B}$ enumerates $\textsc{Enum}_R$. For this, let $x \in \Sigma^{*}$ and assume that we want to enumerate $R(x)$. The algorithm $\mathcal{B}$ works as follows:

- Let y_1 be the first element output by algorithm $\mathcal{A}$ on input x. As this output can be computed by $\mathcal{A}$ in time $p(|x|)$, at most $p(|x|)$ many calls to the L-oracle have been made with an input of size at most $p(|x|)$. Thus the answer of every oracle call can be computed in time $q(p(|x|))$ using an L'-oracle. Therefore y_1 can be output by $\mathcal{B}$ in time $p(|x|) \cdot q(p(|x|))$ by running $\mathcal{A}$ until the first output, and simulating the oracle calls accordingly.

- For $n \geq 2$, let y_n be the n-th element output by $\mathcal{A}$. As with y_1, we can make the n-th output of $\mathcal{B}$ by running $\mathcal{A}$ until y_n is output and simulating the oracle calls accordingly. Indeed, $\mathcal{A}$ takes $p(|x|) \cdot n$ steps to output y_n, with at most $p(|x|) \cdot n$ oracle calls to L with an input of size bounded by $p(|x|) \cdot n$. Thus y_n can be computed in time $p(|x|) \cdot n \cdot q(p(|x| \cdot n))$ using an L'-oracle.

□

In fact, given that the classes Δ_k^P describe problems that can be solved in polynomial time by a deterministic machine with a Σ_{k-1}^P-oracle, it is not very surprising that for enumeration algorithms, Δ_k^P-oracles can be simulated by Σ_{k-1}^P-oracles, as long as one accounts for the potentially accumulating input to the Δ_k^P-oracles.

This property is not restricted to the case shown in Theorem 2, but applies to all Δ_k^P-oracles. As a result, in the remainder of this work, we omit any result regarding the classes $\mathsf{Del}\Delta_k^P, \mathsf{Del}^+\Delta_k^P$ or $\mathsf{Inc}\Delta_k^P$, as they are given implicitly by a result on $\mathsf{Del}\Sigma_k^P$ or $\mathsf{Inc}\Sigma_k^P$ respectively. Indeed, Theorem 2 shows the equality of $\mathsf{Del}^+\Delta_k^P$ and $\mathsf{Inc}\Sigma_{k-1}^P$. Concerning the remaining classes, observe that $\mathsf{DelP}^{\mathcal{C}} = \mathsf{Del}\mathcal{C}$ for any decision complexity class $\mathcal{C}$; in particular we obtain $\mathsf{Del}\Delta_k^P = \mathsf{Del}\Sigma_{k-1}^P$. A similar results holds for the $\mathsf{Inc}\mathcal{C}$ classes even without the bound on the input size to the decision oracle. This is due to the fact that the size of any input α to the decision oracle is at most polynomial in the combined size of the instance and number of outputs. Thus an incremental delay gives us enough time to make $\mathsf{poly}(|\alpha|)$ many oracles calls, meaning that $\mathsf{Inc}\Delta_k^P = \mathsf{Inc}\Sigma_{k-1}^P$.

3.1. A Hierarchy of Enumeration Complexity Classes

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

Theorem 3. *Let $k \geq 0$. Then, unless the polynomial hierarchy collapses to the $(k+1)$-st level,*

$$\mathsf{Del}\Sigma_k^P \subsetneq \mathsf{Del}\Sigma_{k+1}^P, \mathsf{Del}^+\Sigma_k^P \subsetneq \mathsf{Del}^+\Sigma_{k+1}^P, \mathsf{Inc}\Sigma_k^P \subsetneq \mathsf{Del}^+\Sigma_{k+1}^P$$
$$\textit{and } \mathsf{Inc}\Sigma_k^P \subsetneq \mathsf{Inc}\Sigma_{k+1}^P.$$

Proof. Let $k \geq 0$. By Theorem 2 and the definition of our classes we have that

$$\mathsf{Del}^+\Sigma_k^P \subseteq \mathsf{Del}^+\Delta_{k+1}^P = \mathsf{Inc}\Sigma_k^P \subseteq \mathsf{Del}^+\Sigma_{k+1}^P \subseteq \mathsf{Inc}\Sigma_{k+1}^P,$$

so we only need to show that $\mathsf{Del}\Sigma_k^P \subsetneq \mathsf{Del}\Sigma_{k+1}^P$ and $\mathsf{Inc}\Sigma_k^P \subsetneq \mathsf{Del}^+\Sigma_{k+1}^P$. 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 $\mathsf{Del}\Sigma_{k+1}^P$ (thus also in $\mathsf{Del}^+\Sigma_{k+1}^P$). Assume that Enum_$R_L \in \mathsf{Del}\Sigma_k^P$ (or Enum_$R_L \in \mathsf{Inc}\Sigma_k^P$). Then, as there is only one output to the enumeration problem, 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 statement of the following proposition is twofold: First, it shows that the complexity classes based on DelayP with a polynomial bound on the decision oracle and the complexity classes based on DelayP without such a bound, respectively, are very likely to be distinct. Second, the gap between $\mathsf{Del}\Sigma_k^P$ and $\mathsf{Del}^+\Sigma_k^P$ cannot be overcome by equipping the oracle-bounded enumeration complexity class with a slightly more powerful oracle, in contrast to the result of Theorem 2. Note that the proposition refers to the weak EXP hierarchy as defined in Definition 1.

Proposition 4. *Let $k \geq 1$. If $\mathsf{EXP} \subsetneq \Delta_{k+1}^{\mathsf{EXP}}$, then*

$$\mathsf{Del}\Sigma_k^P \subsetneq \mathsf{Del}^+\Sigma_k^P \not\subseteq \mathsf{Del}\Sigma_{k+1}^P.$$

To show Proposition 4, we first need to prove the following result.

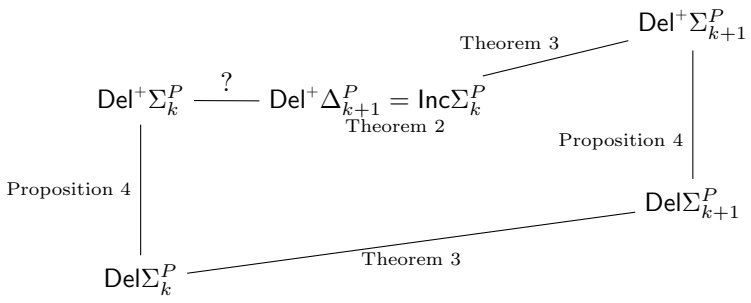


Figure 1: Hierarchy among enumeration complexity classes for all $k \geq 1$. The lines without '?' represent strict inclusions under some reasonable complexity theoretic assumption. For the line with '?', inclusion holds, but it is not clear whether it is strict.

Lemma 5. *Let R be a binary relation and $k \geq 0$. If* ENUM_$R \in \mathsf{Del}\Sigma^P_k$*, then* CHECK_$R \in \mathsf{EXP}$*.*

Proof. Let (x, y) be an instance of CHECK_R and let $k \geq 0$. Further let $\mathcal{A}$ be an enumeration algorithm witnessing the membership ENUM_$R \in \mathsf{Del}\Sigma^P_k$. In order to decide whether $(x, y) \in R$, we simply enumerate all of $R(x)$ and check whether $y \in R(x)$. Let q, r be a polynomials such that the decision of the Σ^P_k-oracle can be computed in $\mathcal{O}(2^{q(n)})$, and the size of the input to any such decision oracle while executing $\mathcal{A}$ is bounded by r. By the definition of polynomial delay and the fact that R is a polynomial relation, there is some polynomial h such that $R(x)$ can be enumerated in time $\mathcal{O}(2^{q(r(n))+h(n)})$, i. e., in exponential time. □

of Proposition 4. Assume that $\mathsf{EXP} \subsetneq \Delta^{\mathsf{EXP}}_{k+1}$. Then there exists some polynomial q and a language L such that $L \in \Delta^{\mathsf{EXP}}_{k+1} \setminus \mathsf{EXP}$ and L can be decided in time $\mathcal{O}(2^{q(n)})$ using a Σ^P_k-oracle. Define the following enumeration problem:

ENUM_D_0	
Instance:	$x \in \Sigma^*$.
Output:	All $\{0,1\}$-words of length $q(\|x\|)$, and 2 if $x \in L$

First note that ENUM_$D_0 \in \mathsf{Del}^+\Sigma^P_k$ by an algorithm $\mathcal{A}$ that enumerates all $2^{q(|x|)}$ words in $\{0,1\}^{q(|x|)}$ in $\mathcal{O}(2^{q(|x|)})$. While enumerating the trivial part of the output, $\mathcal{A}$ also has enough time to compute whether $x \in L$, and then makes the last output ('2' or nothing) accordingly. Next assume that ENUM_$D_0 \in \mathsf{Del}\Sigma^P_k$ (or $\mathsf{Del}\Sigma^P_{k+1}$). Then, by Lemma 5, CHECK_$D_0 \in \mathsf{EXP}$. Therefore we can check for all $x \in \Sigma^*$ whether $(x, 2) \in D_0$, which is equivalent to $x \in L$. Thus we can decide L in exponential time, a contradiction. This proves also the second claim. □

By generalizing the statement of Lemma 5, we even have that $\mathsf{Del}\mathcal{C}$ and $\mathsf{Del}^+\Sigma^P_k$ are incomparable for any $k \geq 1$ and any $\mathcal{C}$ within EXP under the assumption that the weak EXP hierarchy does not collapse. This is due to the fact that CHECK_R for an enumeration problem ENUM_$R \in \mathsf{Del}\mathcal{C}$ is within EXP, whereas $\mathsf{Del}^+\Sigma^P_k$ contains an enumeration problem with corresponding check problem in $\Delta^{\mathsf{EXP}}_{k+1} \setminus \mathsf{EXP}$.

The relations among the enumeration complexity classes introduced in this chapter are summarized in Figure 1.

3.2. Connections to Decision Complexity

For a lot of problems, the set of all solutions can be enumerated by repeatedly solving differing instances of a corresponding decision problem and using the information gained from to construct the solutions.

For the class $\mathsf{Del}\mathcal{C}$, it turns out that the decision problem ExtSol_R is most relevant for achieving the aforementioned goal. Indeed, the standard enumeration algorithm that constructs solutions bit by bit and tests whether the current partial candidate solution can still be extended to a solution [2, 5], and which outputs the solutions in lexicographical order, gives the following relationship.

Proposition 6. *Let R be a binary relation. Further let $k \geq 0$ and $\mathcal{C} \in \{\Delta_k^P, \Sigma_k^P\}$. If* ExtSol_$R \in \mathcal{C}$ *then* Enum_$R \in \mathsf{Del}\mathcal{C}$.

Proof. Let $k \geq 1$. Further let q be a polynomial such that for all $(x, y) \in R$, $|y| \leq q(|x|)$ holds. As $\mathsf{Del}\Delta_{k+1}^P = \mathsf{Del}\Sigma_k^P$, we show that Enum_$R \in \mathsf{Del}\mathcal{C}$ by giving an enumeration algorithm $\mathcal{A}$ with access to ExtSol_R as an oracle. Assume that we have given x as an input for Enum_R. For every $y_1 \in \Sigma$, we make an oracle call with (x, y_1) as input. If the oracle returns 'yes', y_1 can be extended to some $y \in R(x)$, so for every $y_2 \in \Sigma$, we query (x, y_1y_2) to the oracle. Repeating this $q(|x|)$ times gives a first answer $y \in R(x)$. With the use of backtracking, we can enumerate all of $R(x)$ with a polynomial delay and access to the oracle ExtSol_R, where every input to an oracle call is polynomially bounded by q. □

When considering decision problems, typically not the problem ExtSol_R is studied, but instead the problem Exist_R that just asks, given some instance, whether there exists any solution at all. While in general unrelated, for a big class of problems, the so-called self-reducible problems, the problem ExtSol_R can be solved via Exist_R.

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.

Self-reducibility thus allows to reduce enumeration to the typical decision problem associated with R. More formally, for self-reducible problems, Proposition 6 can be refined as follows.

Proposition 7. *Let R be a binary relation, which is self-reducible, and $k \geq 0$. Then* Exist_$R \in \Delta_k^P$ *if and only if* Enum_$R \in \mathsf{Del}\Sigma_k^P$.

The above proposition gives a characterization of the class $\mathsf{Del}\Sigma_k^P$ 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, that directly relates the search problem AnotherSol_R to its corresponding decision problem [4], allows a characterization of the class $\mathsf{Inc}\Sigma_k^P$.

Definition 4 ([4], enumeration self-reducibility). A binary relation R is *enumeration self-reducible* if AnotherSol_$R \leq_T$ Exist-AnotherSol_R.

The following proposition generalizes two results to the higher levels of the enumeration hierarchy, by following analogous arguments as for the basic level IncP. The first result is given by Strozecki in [2, Prop. 2.14], and the second result is given by Kimelfeld and Kolaitis [4, Prop. 2.2].

Proposition 8. *Let R be a binary relation and let $k \geq 0$.*

- *The functional problem* AnotherSol_R *can be solved in polynomial time with access to a Δ_k^P-oracle if and only if* Enum_$R \in \mathsf{Inc}\Sigma_k^P$.
- *Let R be enumeration self-reducible. Then* Exist-AnotherSol_$R \in \Delta_k^P$ *if and only if* Enum_$R \in \mathsf{Inc}\Sigma_k^P$.

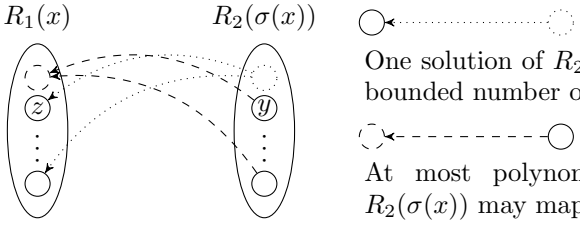


Figure 2: Illustration of relation τ from Definition 5.

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 implicitly described in [16]. It establishes a bijection between sets of solutions. Different approaches introduced in [17] and in [18] relax this condition and allow for non-bijective reduction relations. We go further in that direction in proposing a declarative style reduction relaxing the isomorphism requirement while closing the relevant enumeration classes.

Definition 5 (e-reduction). Let $R_1, R_2 \subseteq \Sigma^* $ be binary relations. Then ENUM_R_1 reduces to ENUM_R_2 via an e-reduction, ENUM_R_1 $\leq_e$ 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:

1. $R_1(x) = \bigcup_{y \in R_2(\sigma(x))} \tau(x, y, -)$;
2. For all $y \in R_2(\sigma(x))$, either $\tau(x, y, -) = \emptyset$, or $\emptyset \subsetneq \tau(x, y, -) \subseteq R_1(x)$ and $\tau(x, y, -)$ can be enumerated with polynomial delay in $|x|$; moreover $\tau(x, y, -) = \emptyset$ can only hold for a number of y's which is polynomially bounded in $|x|$;
3. For all $z \in R_1(x)$, we have $\tau(x, -, z) \subseteq R_2(\sigma(x))$ and the size of $\tau(x, -, z)$ is polynomially bounded in $|x|$.

Intuitively, τ establishes a relationship between instances x, solutions $y \in R_2(\sigma(x))$ and solutions $z \in R_1(x)$. We can thus use τ to design an enumeration algorithm for $R_1(x)$ via an enumeration algorithm for $R_2(\sigma(x))$. The conditions imposed on τ have the following meaning: By condition 1, the solutions $z \in R_1(x)$ can be computed by iterating through the solutions $y \in R_2(\sigma(x))$ and computing $\tau(x, y, -) \subseteq R_1(x)$. Conditions 2 and 3 make sure that the delay of enumerating $R_1(x)$ only differs by a polynomial from the delay of enumerating $R_2(\sigma(x))$: Condition 2 ensures that, for every y, the set $\tau(x, y, -)$ can be enumerated with polynomial delay and that we do not encounter more than a polynomial number of "useless" y (i.e., a solution $y \in R_2(\sigma(x))$ which is associated with no solution $z \in R_1(x)$).

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, cf. the proof of Proposition 9 or [2]. Figure 2 illustrates the idea of τ, and a concrete example of an e-reduction is provided next.

Example 1. The idea of the relation τ can also be nicely demonstrated on an e-reduction from E-3-COLOURABILITY to E-4-COLOURABILITY (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 an 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 e-reductions have two desirable and important properties, as stated next.

Proposition 9. *1. Reducibility via e-reductions is a transitive relation.*

2. Let $\mathcal{C} \in \{\Sigma_k^P, \Delta_k^P \mid k \geq 0\}$. The classes $\mathsf{Del}\mathcal{C}$, $\mathsf{Del}^+\mathcal{C}$, and $\mathsf{Inc}\mathcal{C}$ are closed under e-reductions.

Proof. We first show that the enumeration classes $\mathsf{Del}^+\mathcal{C}$, $\mathsf{Del}\mathcal{C}$ and $\mathsf{Inc}\mathcal{C}$ are closed under e-reductions for $\mathcal{C} \in \{\Sigma_k^P, \Delta_k^P \mid k \geq 0\}$. Let R_1, R_2 be binary relations with $\textsc{Enum_}R_1 \leq_e \textsc{Enum_}R_2$. Further let σ and τ be relations corresponding to the reduction $\textsc{Enum_}R_1 \leq_e \textsc{Enum_}R_2$, and assume that $\textsc{Enum_}R_2 \in \mathsf{Del}^+\mathcal{C}$ (the cases where $\textsc{Enum_}R_2 \in \mathsf{Del}\mathcal{C}$ or $\textsc{Enum_}R_2 \in \mathsf{Inc}\mathcal{C}$ work along the same lines). Let $\mathcal{A}$ denote the enumeration algorithm for $\textsc{Enum_}R_2$ with a polynomial delay and decision oracle $\mathcal{C}$, and let $\mathcal{B}$ be the polynomial delay algorithm enumerating $\tau(x, y, -)$ for all $x, y \in \Sigma^*$. Moreover, let p be a polynomial such that for all $z \in R_1(x)$, we have $|\tau(x, -, z)| \leq p(|x|)$. The idea for an enumeration algorithm for $\textsc{Enum_}R_1$ is to enumerate (without output) $R_2(\sigma(x))$ via $\mathcal{A}$, and for every element y that would be output by $\mathcal{A}$, repeatedly add $p(|x|)$ elements of $\tau(x, y, -)$ to a priority queue. Then, whenever those elements are added to the queue, an element of the queue w.r.t. some order is output. This way, one can ensure polynomial (respectively incremental) delay albeit producing an exponentially large priority queue.
To give a detailed explanation of the enumeration algorithm for $\textsc{Enum_}R_1$, fix some $x \in \Sigma^*$. Denote by $\mathsf{Newoutput}_{\mathcal{A}}(\sigma(x))$ a new output made by the enumeration algorithm $\mathcal{A}$ when enumerating $R_2(\sigma(x))$, and similarly by $\mathsf{Newoutput}_{\mathcal{B}}(y)$ a new output made by the enumeration algorithm $\mathcal{B}$ when enumerating $\tau(x, y, -)$. Algorithm 1 gives the algorithm for enumerating $R_1(x)$. It is easy to see that this algorithm indeed works with a polynomial (respectively incremental) delay. When adding some element z to the priority queue (line 10 of Algorithm 1), if z is already in the queue, then `output_queue` remains unchanged, otherwise z is added as the last element. Moreover, note that we do not delete elements in the priority queue in order to avoid the duplicates in the output.

To show that reducibility via e-reductions is a transitive relation, let $R_1, R_2, R_3 \subseteq \Sigma^*$ be binary relations. Further let $\textsc{Enum_}R_1 \leq_e \textsc{Enum_}R_2$ with corresponding polynomial p_1 and relations τ_1 and σ_1 and $\textsc{Enum_}R_2 \leq_e \textsc{Enum_}R_3$ with corresponding polynomial p_2 and relations τ_2 and σ_2. Define relations σ_3 as $\sigma_3 := \sigma_2 \circ \sigma_1$ and τ_3 as

$$\tau_3 := \{(x, y, z) \in \Sigma^* \times \Sigma^* \times \Sigma^* \mid \exists \zeta \in \Sigma^* \text{ with } (x, \zeta, z) \in \tau_1 \text{ and } (\sigma_1(x), y, \zeta) \in \tau_2\}.$$

Let $x, y \in \Sigma^*$. To show that $\tau_3(x, y, -)$ can be enumerated with a polynomial delay in $|x|$, we use the same idea for a polynomial delay enumeration as we did to show that the reduction closes the enumeration classes: First a single $y' \in R_2(\sigma_1(x))$ is computed by $\tau_2(\sigma_1(x), y, -)$ (with a polynomial delay in $\sigma_1(|x|)$ and thus polynomial delay in $|x|$), and then at most polynomially many y'' from $\tau_1(x, y', -)$ are

Algorithm 1 Enumerate $R_1(x)$

$i = 1$
output_queue $= \emptyset$

while $\mathcal{A}$ has not output all of $R_2(\sigma(x))$ **do**
  $y = \mathsf{Newoutput}_{\mathcal{A}}(\sigma(x))$
  output_delay $= 1$
  while output_delay $< p(|x|)$ **do**
    $z = \mathsf{Newoutput}_{\mathcal{B}}(y)$
    if $z \neq \emptyset$ **then**
      Add z to output_queue
      output_delay $=$ output_delay $+ 1$
    else
      $y = \mathsf{Newoutput}_{\mathcal{A}}(\sigma(x))$
    end if
    if $y = \emptyset$ **then**
      Output (without deleting) the elements of output_queue starting from the i-th element
      output_delay $= p(|x|)$
    end if
  end while
  if $y \neq \emptyset$ **then**
    Output (without deleting) the i-th element of output_queue
  end if
  $i = i + 1$
end while

added to some priority queue. An element of the queue is output, and again polynomially many elements are added by the queue (possibly by first computing some new $y' \in R_2(\sigma_1(x))$. This way, all of $\tau_3(x, y, -)$ can be enumerated with a polynomial delay in $|x|$. Moreover, it is easy to see that for all $x, z \in \Sigma^*$ we have $|\tau_3(x, -, z)| \leq p_1(p_2(|x|))$ and that $R_2(x) = \bigcup_{y \in R_3(\sigma_3(x))} \tau_3(x, y, -)$. It follows that indeed Enum_$R_1 \leq_e$ Enum_R_3. □

Note that, as an additional benefit, Enum_$R_1 \leq_e$ Enum_R_2 does not imply Exist_$R_1 \leq_m$ Exist_R_2. An example for this behavior can be found in Theorem 15 below.

Thus, besides being a natural extension of existing reductions, Proposition 9 also shows that e-reductions have important desirable properties. As a result, they are very well suited for reasoning about enumeration algorithms and complexities. For example, they allow to compare different enumeration problems, thus supporting to make statements on one problem "relative" to some other problem, and can be used to derive enumeration algorithms – hence complexity upper bounds.

In fact, they have already been implicitly used previously for these purposes. For example, an important result from [19] compares the complexity of two well-known problems by implicitly using an e-reduction:

Example 2. We denote by Dom-Enum the problem of enumerating all subset-minimal dominating sets of a given graph, and by Trans-Enum the problem of enumerating all minimal hitting sets of a hypergraph. In [19], it was shown that Trans-Enum can be enumerated via an enumeration algorithm for Dom-Enum, by mapping a hypergraph to an instance of Dom-Enum, and giving a (non-bijective) map between the minimal dominating sets and the minimal hitting sets. In fact this defines the following e-reduction Trans-Enum $\leq_e$ Dom-Enum:

- Given a hypergraph $\mathcal{H}$, $\sigma(\mathcal{H})$ is defined as the co-bipartite incidence graph of $\mathcal{H}$, with

$$\begin{aligned} V(\sigma(\mathcal{H})) =& V(\mathcal{H}) \cup \{y_e \mid e \in E(\mathcal{H})\} \cup \{v\}, \\ E(\sigma(\mathcal{H})) =& \{\{x, y_e\} \mid x \in V(\mathcal{H}), x \in e\} \cup \{\{x, y\} \mid x, y \in V(\mathcal{H})\} \\ & \cup \{\{v, x\} \mid x \in V(\mathcal{H})\} \cup \{\{y_e, y_f\} \mid e, f \in E(\mathcal{H})\}. \end{aligned}$$

- Given $\sigma(\mathcal{H})$ and some minimal dominating set $D \subseteq \sigma(\mathcal{H}))$, we have that

$$\tau(\sigma(\mathcal{H}), D, -) = \begin{cases} \emptyset & : D = \{x, y_e\} \text{ for some } x \in V(\mathcal{H}), e \in E(\mathcal{H}), \\ D & : \text{ otherwise.} \end{cases}$$

Note that Dom-Enum $\leq_e$ Trans-Enum holds as well. The original question studied in [19] was whether any of these problems can be enumerated in a time that is polynomially in both, the size of the input and the size of the output. For the two problems Dom-Enum and Trans-Enum, this is a longstanding open problem. The conclusion in [19] was that if such an enumeration algorithm exists for one of the problems, then this is also possible for the other one. It can be shown that the existence of the e-reductions between Dom-Enum and Trans-Enum also induces the same result. Moreover, they are also equivalent for all of the classes that are closed under e-reductions as given in Proposition 9, i.e., if one of the two problems is in any of these classes, then the other problem is as well.

Another example that implicitly uses e-reductions is [20]. In later sections of this paper, we will use them to retrieve reductions among enumeration problems as well. And even though we do not show any completeness results under e-reduction, we will actually use them to show hardness under alternative reductions that will be introduced in the next section.

5. Procedural-style Reductions

Although Turing reductions are too strong to show completeness results for classes in the polynomial hierarchy, because the classes Σ_k^P and Δ_{k+1}^P have the same closure, 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 e-reductions seem to be too weak.

Towards this goal, we first have to define the concept of RAMs with an *oracle for enumeration problems* (such oracles were already introduced in [2]). 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 $R(x)$, and then $\mathcal{A}$ can proceed according to this output.

Definition 6 (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 that has a sequence of new registers $A_e, O^e(0), O^e(1), \ldots$ 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 $y_i \in R(x)$ to register A_e, where x is the word stored in $O^e(0), O^e(1), \ldots$ and y_i is defined as follows:

Definition 7 (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., $\pi_k = \mathsf{NOO}$. Denote with x_i the word stored in $O^e(0), O^e(1), \ldots$ at step i. Let $K = \{\pi_i \in \{\pi_1, \ldots, \pi_{k-1}\} \mid \pi_i = \mathsf{NOO} \text{ and } x_i = x_k\}$. Then the *oracle output* y_k *in* π_k is defined as an arbitrary $y_k \in R(x_k)$ 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 register A_e contains some special symbol in step π_{k+1}. Otherwise in step π_{k+1} the register A_e contains y_k .

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

Using EOMs, we can now define another type of reductions among enumeration 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 8 (D-reductions, I-reductions). Let R_1 and R_2 be binary relations.

- We say that ENUM_R_1 reduces to ENUM_R_2 via D-reductions, ENUM_$R_1 \leq_D$ ENUM_R_2, if there is an oracle-bounded EOM_R_2 that enumerates R_1 in DelP and is independent of the order in which the ENUM_R_2-oracle enumerates its answers.
- We say that ENUM_R_1 reduces to ENUM_R_2 via I-reductions, 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.

The following proposition shows that these reductions have desirable properties:

Proposition 10. *Reducibility via D-reductions as well as reducibility via I-reductions are transitive relations.*

Proof. This can be proven along the same lines as Proposition 11 by substituting occurrences of enumeration RAMs with decision oracles by enumeration RAMs □

For D-reductions, we require 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 DelC are not closed under the resulting reduction. Indeed, suppose that the D_u-reduction is defined similar to the D-reduction but without the restriction to an oracle-bounded EOM. Then, abusing the decision problem ANOTHERSOLEXT$_R^{\mathcal{A}}$ from the proof of Theorem 2 as an enumeration problem ENUM_R_A (with output 'yes' or 'no'), we can show that for any $k \geq 1$ and any ENUM_$R \in \mathsf{Inc}\Sigma_k^P$ we have that ENUM_$R \leq_{D_u}$ ENUM_R_A but ENUM_$R_A \in \mathsf{Del}\Sigma_k^P$.

Proposition 11. *Let $\mathcal{C} \in \Sigma_k^P, k \geq 0$. The class* DelC *is closed under D-reductions and the class* IncC *is closed under I-reductions.*

Proof. Let M be an oracle-bounded enumeration oracle machine with an enumeration oracle ENUM_R_2 witnessing that ENUM_$R \leq_D$ ENUM_R_2. Let $\mathcal{A}$ be the polynomial delay algorithm with access to a polynomially bounded $\mathcal{C}$-oracle We can

construct a RAM M' that enumerates ENUM_R_1 with a polynomial delay using a polynomially bounded decision oracle $\mathcal{C}$, by modifying the RAM M as follows: Every time M makes a call to an ENUM_R_2-oracle, we use the algorithm $\mathcal{A}$ to retrieve what should be written to the register A_e. Assume that x is the input to an oracle call of the RAM M. Then the new RAM N assigns two fixed addresses a_x^0 and a_x^1 to x. Then N can simulate the algorithm $\mathcal{A}$ on the registers $R(2^{a_x^0}), \ldots, R(2^{a_x^1})$ until $\mathcal{A}$ would output some $y \in \Sigma^*$. The RAM N writes y to A_e, and a simulation of a single oracle call is completed. Whenever x is the input for a NOO-call, N continues to simulate $\mathcal{A}$ on those registers; this way, the enumeration of $R_2(x)$ does not need to start from the beginning every time x is the input of an oracle call. The proof of the closure of Inc$\mathcal{C}$ under I-reductions can be done along the same lines. □

Observe that the e-reductions introduced in the previous section are a particular case of D- and I-reductions. We will also use e-reductions later when we establish completeness results for specific problems.

Proposition 12. *Let R_1 and R_2 be binary relations.*

- *If* ENUM_$R_1 \leq_e$ ENUM_R_2, *then* ENUM_$R_1 \leq_D$ ENUM_R_2.
- *If* ENUM_$R_1 \leq_D$ ENUM_R_2, *then* ENUM_$R_1 \leq_I$ ENUM_R_2.

Proof. The proof of the second claim follows immediately from the definitions. Now, if ENUM_$R_1 \leq_e$ ENUM_R_2, then Algorithm 1 developed in the proof of Proposition 9 provides an oracle bounded EOM_R_2 that enumerates R_1 in DelP. For enumerating $R_1(x)$ the calls to the enumeration oracle ENUM_R_2 are only made on the input $\sigma(x)$, where σ is the function used in the e-reduction. □

Now, unlike for e-reductions, the next theorem shows that the D- and I-reduction induce complete problems for the enumeration complexity classes introduced in Section 3.

Theorem 13. *Let R be a binary relation and $k \geq 1$ such that* EXIST_R *is Σ_k^P-hard.*

- ENUM_R *is* DelΣ_k^P*-hard via D-reductions.*
- ENUM_R *is* IncΣ_k^P*-hard via I-reductions.*
- *If additionally* EXIST_R *is in Σ_k^P and R is self-reducible, then* ENUM_R *is* DelΣ_k^P*-complete via D-reductions and* IncΣ_k^P*-complete via I-reductions.*

Proof. Let R be a relation such that EXIST_R is Σ_k^P-complete for some $k \geq 1$.

- We have to prove that for any ENUM_$R' \in$ DelΣ_k^P, ENUM_$R' \leq_D$ ENUM_R (IncΣ_k^P-hardness via I-reductions can be shown along the same lines). So let R' be a binary relation such that ENUM_$R' \in$ DelΣ_k^P. By definition there is some $L \in \Sigma_k^P$ such that ENUM_$R' \in$ Del$L$. Moreover let $\mathcal{A}$ be an algorithm witnessing this membership. As EXIST_R is Σ_k^P-complete, we have that $L \leq_m$ EXIST_R, so any input x to an L-decision oracle when enumerating ENUM_R can be transformed to an instance $x' \in$ EXIST_R such that $x \in L$ iff $x' \in$ EXIST_R, and this transformation can by done in polynomial time in the size of x. Moreover, since the size of the oracle input is polynomial, this reduction can be computed within the time bounds of a polynomial delay, i.e. whenever a polynomial delay algorithm with an L-oracle makes an oracle call with an input x, the same algorithm can also perform a transformation to some x' before that oracle call, without violating the polynomial delay restriction. Therefore we can enumerate ENUM_R' with an oracle bounded enumeration

oracle machine with ENUM_R as follows: Whenever $\mathcal{A}$ would make a decision oracle call to L with input x, instead the machine transforms this to some $x' \in \Sigma^*$, and then makes a NOO-instruction with input x' to the ENUM_R-oracle. The NOO-instruction writes a nonempty string to the register A_e if and only if $x' \in$ EXIST_R and thus if and only if $x \in L$. It follows that we can simulate the decision oracle call with an enumeration oracle call.

- Membership of ENUM_R in $\mathsf{Del}\Sigma_k^P$ in this case follows immediately from Proposition 6. Since $\mathsf{Del}\Sigma_k^P \subseteq \mathsf{Inc}\Sigma_k^P$, this also shows membership in $\mathsf{Inc}\Sigma_k^P$.

□

As a consequence, the enumeration problems E-Σ_kSAT and also E-Π_kSAT are natural complete problems for our enumeration complextiy classes:

Corollary 14. *Let $k \geq 1$. Then*

1. E-Σ_kSAT *is complete for* $\mathsf{Del}\Sigma_k^P$ *via D-reductions.*
2. E-Π_kSAT *and* E-Σ_{k+1}SAT *are complete for* $\mathsf{Inc}\Sigma_{k+1}^P$ *via I-reductions. In particular* E-SAT *the enumeration variant of the traditional* SAT *problem is* InclNP*-complete via I-reductions.*

Proof. The results for E-Σ_kSAT follow immediately from Theorem 13. It only remains to prove that that E-Π_kSAT and E-Σ_{k+1}SAT are equivalent under I-reductions. Note that E-Π_kSAT $\leq_I$ E-Σ_{k+1}SAT follows immediately from the fact that E-Π_kSAT is a special case of E-Σ_{k+1}SAT, so it suffices to show that E-Σ_{k+1}SAT $\leq_I$ E-Π_kSAT. Thus consider an instance ψ of E-Σ_{k+1}SAT given as $\psi(x) := \exists y_0 \forall y_1 \dots Q_k y_k \phi(x, y_0, \dots, y_k)$. We can enumerate all solutions to ψ as follows: The first input to a E-Π_kSAT-oracle is $\psi_0(x, y_0) := \forall y_1 \dots Q_k y_k \phi(x, y_0, \dots, y_k)$, with free variables x and y_0. A single NOO instruction thus gives a solution x_0, y_0' for ψ_0, and x_0 can be output as a solution to ψ. The next solution can be found by calling a NOO instruction with the input $\psi_1(x, y_0) = \forall y_1 \dots Q_k y_k (\phi(x, y_0, \dots, y_k) \wedge (x_0 \neq x))$. We only need to add the clauses of $(x_0 \neq x)$ to the input registers of the oracle tape, and we can choose an encoding such that this does not alter the previous input, but extends it. The output x_1, y_0'' of the second oracle call gives the second output x_1 for the E-Σ_{k+1}SAT problem. By repeating this method until an oracle call gives back the empty solution, we can enumerate the solutions of ψ. □

Observe that, via different reductions, E-Σ_kSAT is complete for both, $\mathsf{Inc}\Sigma_k^P$ and for the presumably smaller class $\mathsf{Del}\Sigma_k^P$. This provides additional evidence that the two reductions nicely capture $\mathsf{Inc}\Sigma_k^P$ and $\mathsf{Del}\Sigma_k^P$, respectively.

Roughly speaking Theorem 13 says that any 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 the next section.

6. Completeness Results

In this section, we prove completeness results under procedural-style reductions for several problems. We start by considering generalized satisfiability in Schaefer's framework, and then continue by looking into minimal models of Boolean formulæ. Next, we study enumeration within the framework of abduction, and finally turn to the problem of enumerating repairs of an inconsistent database.

6.1. Generalized Satisfiability

In this subsection, we revisit a classification theorem obtained for the enumeration of generalized satisfiability [20] in our framework. 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 [21, 20]. We say that a constraint language is *Schaefer* if every relation in Γ is either Horn, dualHorn, bijunctive, or affine.

E-SAT(Γ)	
Instance:	ϕ a Γ-formula
Output:	All satisfying assignments of ϕ.

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

Theorem 15. *Let Γ be a finite constraint language. If Γ is Schaefer, then* E-SAT(Γ) *is in* DelP*, otherwise it is* DelNP*-complete via D-reductions and* IncNP*-complete via I-reductions.*

Proof. The polynomial cases were studied in [20]. Let us now consider the case where Γ is not Schaefer. Membership of E-SAT(Γ) in DelNP 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 [21], deciding whether a $\Gamma \cup \{\mathrm{F}, \mathrm{T}\}$-formula is satisfiable is NP-complete. Since this problem is self-reducible, according to Theorem 13, E-SAT($\Gamma \cup \{\mathrm{F}, \mathrm{T}\}$) is DelNP-complete via D-reductions. Let Imp denote the binary implication relation $\mathrm{Imp}(x, y) \equiv (x \longrightarrow y)$. Given a $\Gamma \cup \{\mathrm{F}, \mathrm{T}\}$-formula φ, one can construct a $\Gamma \cup \{\mathrm{Imp}\}$ formula φ' as follows. Let V denote the set of variables in φ, V_0 the set of variables x on which lies the constraint $F(x)$, and V_1 the set of variables x on which lies the constraint $T(x)$. Let f and t be two fresh variables. Given a set of variables W and a variable z, $\varphi[W/z]$ denotes the formula obtained from φ in replacing each occurrence of a variable from W by the variable z. Then we set $\varphi' = \varphi[V_0/f] \wedge \varphi[V_1/t] \wedge \bigwedge_{x \in V} \mathrm{Imp}(x, t) \wedge \bigwedge_{x \in V} \mathrm{Imp}(f, x)$. Clearly models of φ coincide with models of φ' in which f is set to false and t to true, and φ' might have two additional models, the all-zero one and the all-one one. In [20] it is essentially proven that if Γ is not Schaefer then the implication relation Imp can be expressed by a Γ-formula. Therefore, in this case the construction described above shows that E-SAT($\Gamma \cup \{\mathrm{F}, \mathrm{T}\}$) $\leq_e$ E-SAT(Γ), thus proving the DelNP-completeness via D-reductions and IncNP-completeness via I-reductions of the latter problem, according to Proposition 12. □

To come back to the above discussion, we point out that there exist constraint languages Γ such that the decision problem SAT(Γ) is in P, while the enumeration problem E-SAT(Γ) is DelNP-complete, namely 0-valid or 1-valid constraint languages that are not Schaefer.

6.2. Enumeration of Minimal Models

Let us now turn to the complexity of enumerating minimal models of a Boolean formula. Such an enumeration is ubiquitous in practical settings, ranging from verification, to databases and to knowledge representation among many others. One can

consider subset-minimality as well as cardinal-minimality. Therefore we consider the two following enumeration problems: E-Circumscription denotes the problem of enumerating all subset-minimal models of a boolean formula and E-CardMinSAT denotes the problem of enumerating all cardinality-minimal models of a boolean formula. We pinpoint the complexity of these two enumeration problems.

Observe that contrary to the satisfiability problems we discussed above, enumeration of minimal models is not obviously self-reducible. In particular, the problem E-Circumscription is very unlikely to be self-reducible: The problem of deciding if a partial truth assignment can be extended to a subset minimal model is Σ_2^P-complete [22], while deciding the existence of a minimal model is clearly NP-complete. Thus E-Circumscription is not self-reducible unless the polynomial hierarchy collapses to the first level.

Theorem 16. E-Circumscription *is* IncNP*-complete via I-reductions.*

Proof. Hardness follows immediately from Theorem 13, since deciding the existence of a (minimal) model of a Boolean formula is NP-complete.

To obtain membership, we show that E-Circumscription can be even enumerated in $\mathsf{Del}^+\mathsf{NP}$. Consider a boolean formula $\phi(x_1, \ldots, x_n)$, and assume that we want to enumerate the minimal models of ϕ. We start by copying ϕ to the oracle registers. A very first minimal model x' can be achieved by a greedy algorithm using an NP oracle. Next we extend ϕ in the oracle registers to $\phi' = \phi \wedge ((x_1 < x'_1) \vee \ldots \vee (x_n < x'_n))$ and again get a minimal model x'' for ϕ' using a greedy algorithm with an NP-oracle. This is also a minimal model for ϕ; repeatedly extending ϕ and then computing a minimal model via a greedy algorithm achieves the membership E-Circumscription $\in \mathsf{Del}^+\mathsf{NP}$ and thus also E-Circumscription $\in$ IncNP. □

What makes the completeness result obtained for E-Circumscription surprising is the discrepancy from the behaviour of the counting variant of the problem: The counting variant of E-Circumscription is a prototypical $\# \cdot \mathsf{coNP}$-complete problem [23], and thus of the same hardness as the counting variant of E-Π_1SAT. However, for enumeration we have that E-Circumscription shows the same complexity as E-Σ_1SAT, which is considered to be lower than that of E-Π_1SAT.

Theorem 17. E-CardMinSAT *is* DelNP*-complete via D-reductions and* IncNP*-complete via I-reductions.*

Proof. Hardness follows again immediately from Theorem 13.

Membership of E-CardMinSAT in DelNP (and a fortiori in IncNP) follows from the fact that polynomial time with access to an NP-oracle gives us the possibility to compute first the minimal cardinality of models. Having this information we can enumerate all cardinality-minimal models by the standard binary search tree with an NP-oracle that enumerates them in lexicographical order. □

6.3. Model-based Diagnosis

The problem of model-based diagnosis [24] is ubiquitous in practical settings. A usual formulation of this problem is as follows: given a system description composed of some components, and an observation inconsistent with the system description, the goal is to identify a cardinality-minimal set of components which, if declared faulty and thus put out of the system description, restores the consistency between the model and the observation. In recent years the practical enumeration of minimum-size diagnoses has gained interest through translations to MaxSAT instances and the use of efficient MaxSAT solvers (see e.g. [25]).

Formally a system description can be represented by a consistent set of propositional formulæ, each encoding the normal behavior of a component. The observation

is represented by an additional formula that is inconsistent with the former set of formulæ. Thus we get the following enumeration problem.

E-Model-Based Diagnosis	
Instance:	$\mathcal{B} = \{\phi_1, \ldots, \phi_n\}$ a consistent set of formulæ, μ a satisfiable formula inconsistent with $\mathcal{B}$
Output:	All cardinality-maximal subsets $\mathcal{B}' \subseteq \mathcal{B}$ such that $\bigwedge_{\phi \in \mathcal{B}'} \phi \wedge \mu$ is satisfiable.

Observe that for this enumeration problem, the corresponding decision problem Exist_R is trivial because there always is a solution $\mathcal{B}'$. Nevertheless we prove hardness by an e-reduction from a previously identified hard problem. So contrary to all results above, we obtain here a hardness result for an enumeration problem which does not rely on hardness of any related decision problem.

The following theorem settles the complexity of E-Model-Based Diagnosis in our hierarchy.

Theorem 18. E-Model-Based Diagnosis *is* DelNP*-complete via D-reductions and* IncNP*-complete via I-reductions.*

Proof. Membership of E-Model-Based Diagnosis in DelNP (and a fortiori in IncNP) follows from the fact that polynomial time with access to an NP-oracle gives us the possibility to compute first the maximal cardinality of subsets of $\mathcal{B}$ that are consistent with μ. Having this information we can enumerate all such cardinality-maximal subsets by the standard binary search tree with an NP-oracle that enumerates them in lexicographical order.

For hardness we show that E-CardMinSAT $\leq_e$ E-Model-Based Diagnosis, thus proving that E-Model-Based Diagnosis is DelNP-hard via D-reductions and IncNP-hard via I-reductions, according to Proposition 12 and Proposition 16. Let ϕ be a formula given as an instance of E-CardMinSAT, and assume $\mathrm{var}(\phi) = \{x_1, \ldots, x_n\}$. Without loss of generality suppose in addition that ϕ is not 0-valid, i.e., is not satisfied by the all-zero assignment. We define a corresponding instance $\sigma(\phi)$ of E-Model-Based Diagnosis as follows: Let $\mathcal{B}$ consist of the n formulæ $\phi'_i = (\phi \vee x_0) \wedge (\neg x_i \vee x_0)$ for $i = 1, \ldots, n$, and let $\mu = \neg x_0$, where x_0 is a fresh variable. Observe that on the one hand the set $\mathcal{B}$ is consistent (set x_0 to true), and on the other hand it is inconsistent with μ (since by assumption ϕ is not 0-valid). It is easy to see that there is a bijection π mapping the cardinality-minimal models x' of ϕ to cardinality-maximal subsets of $\mathcal{B}$ that are consistent with μ. Thus we can define a relation τ with $\tau(\sigma(\phi), x', -) = \pi(x')$, showing that E-CardMinSAT $\leq_e$ E-Model-Based Diagnosis. □

6.4. Abduction

We now provide an example of an enumeration problem hard in a higher level of the hierarchy, namely abduction. Abduction is a well-studied form a non-monotonic reasoning. Given a certain consistent knowledge, abductive reasoning is used to generate explanations for observed manifestations. We consider here propositional abduction, in which the knowledge base as well as the manifestations are represented by propositional formulæ [22].

The enumeration problem can be formalized as follows:

E-Abduction	
Instance:	Γ a set of formulæ (knowledge), H a set of literals (hypotheses), q a variable that does not occur in H (manifestation).
Output:	All sets $E \subseteq H$ such that $\Gamma \wedge E$ is satisfiable and $\Gamma \wedge E \models q$ (equivalently, $\Gamma \wedge E \wedge \neg q$ is unsatisfiable).

The decision problem Abduction corresponding to this variant of abduction is known to be Σ_2^P-complete, see, e.g., [26]. Based on this we obtain hardness of the enumeration problem. Though this is maybe not very surprising, we mention this problem here as a first example of a problem whose enumeration is even harder than those we studied before, at least under the assumption that the polynomial hierarchy does not collapse.

Theorem 19. E-Abduction *is* $\mathsf{Del}\Sigma_2^P$*-complete via D-reductions and* $\mathsf{Inc}\Sigma_2^P$*-complete via I-reductions.*

Proof. As we have mentioned above, the associated decision problem is Σ_2^P-complete. Also, note that the problem is self-reducible in the sense of Definition 3. Given an instance (Γ, H, q) of E-Abduction and a partial solution $E \subseteq H$, E can be extended to an explanation iff $(\Gamma \wedge E, H, q) \in$ Abduction. Therefore, according to the last statement of Theorem 13, the theorem follows. □

6.5. Database Repairs

Finally, we provide an example of an enumeration problem from the database domain, more specifically the problem of enumerating all "repairs" of an inconsistent database. In this problem, we are given a *database instance* D (i.e, simply a finite structure given by a domain and a collection of relations; for our purposes, it is most convenient to represent D as a set of ground atoms) and a set C of *integrity constraints* (i.e., in general, a set of formulæ from some fragment of first-order logic). Now suppose that D is *inconsistent*, i.e., $D \not\models C$ holds. Then we are interested in enumerating the possible repairs. A *repair* is a database instance I which satisfies C and which *differs minimally* from D. The term "differs minimally" can be defined in several ways. The most common approach is the one introduced by Arenas et al. [27]: here, repairs are obtained from the original database by inserting or deleting ground atoms to or from D, respectively. Then "minimal difference" means that the symmetric set difference $\Delta(D, I)$ is minimal w.r.t. subset inclusion. More formally, let $\Delta(D, I) = (D \setminus I) \cup (I \setminus D)$. Then I is a repair of D w.r.t. C if $I \models C$ (i.e., I is consistent w.r.t. C) and there does not exist an instance I' that satisfies C and $\Delta(D, I') \subsetneq \Delta(D, I)$.

Many different fragments of first-order logic have been studied for the definition of the integrity constraints C. We only look at one such fragment here, namely *equality generating dependencies* (EGDs, for short). EGDs are a classical formalism for defining database constraints [28]. They generalize functional dependencies (FDs), which in turn generalize key dependencies (KDs). Formally, they are defined as formulæ of the form

$$\forall \vec{x}\big(\phi(\vec{x}) \to x_i = x_j\big),$$

where ϕ is a conjunction of atoms over the predicate symbols occurring in D, $\vec{x}$ denotes the variables occurring in these atoms, and x_i, x_j are two variables from $\vec{x}$. The meaning of such an EGD is that, for every variable binding λ on the variables $\vec{x}$, such that applying λ to the atoms in $\phi(\vec{x})$ sends all atoms of $\phi(\vec{x})$ into the database instance D, it must be the case that $\lambda(x_i) = \lambda(x_j)$ holds. Following the usual notational convention in the database literature, we will omit the quantifiers and

denote EGDs of the above form simply as $\phi(\vec{x}) \rightarrow x_i = x_j$ with the understanding that all variables occurring in an EGD are universally quantified.

The enumeration problem considered here is defined as follows:

E-REPAIR		
Instance:	Database instance D, set C of EGDs	
Output:	D	if $D \models C$ and
	All repairs I of D w.r.t. C	if $D \not\models C$.

As in Section 6.3, we observe that for this enumeration problem, the corresponding decision problem EXIST_R is trivial because for every inconsistent database instance, the existence of a repair is guaranteed. Moreover, it is worth mentioning that a different variant of repair enumeration has been recently studied in [29], where a different type of integrity constraints was considered and prereferences were taken into account.

Several decision problems have been studied in the context of repairs. The most intensively studied one is the CONSISTENT QUERY ANSWERING problem, i.e., given a database instance D, a set C of integrity constraints and a boolean query Q, is the query Q true in every repair I of D, w.r.t. C. For EGDs as constraint language and conjunctive queries as query language, this problem has been shown to be Π_2^P-complete [30]. Another decision problem is the REPAIR CHECKING problem, i.e., given database instances D and I, and a set C of integrity constraints, is I a repair of D w.r.t. C. In other words, this is the CHECK problem corresponding to the enumeration problem E-REPAIR. For EGDs, this decision problem is DP-complete [30]. Definitely a bit more surprising than in case of E-ABDUCTION, we get $\mathsf{Inc}\Sigma_2^P$-completeness also for the E-REPAIR problem. In particular, the hardness-proof is more involved than for the previously studied problems.

Theorem 20. E-REPAIR *is* $\mathsf{Inc}\Sigma_2^P$*-complete via* I*-reductions.*

Proof. For the *membership* proof, it is important to note that repairs of D w.r.t. a set of EGDs can only be obtained by deletions, i.e., if a constraint is violated, it does not help to add further atoms. In other words, EGDs have a monotonic behaviour in the sense that if a database instance I violates an EGD, then also every superset of I does. This allows us to redefine E-REPAIR as ENUM_R, where the relation R consists of pairs $((D, C), V)$ such that

- D is a database instance containing $N \geq 0$ atoms, which are numbered as $A_1, \ldots, A_N$,
- C is a set of EGDs and
- V is a word of length N over $\{0, 1\}$, where $V(i) = 0$ means that we delete A_i from D and $V(i) = 1$ means that we retain A_i in D.

We observe that EXTSOL_$R \in \Sigma_2^P$: Given some word $V_1 \in \{0,1\}^{N-k}$, we need a non-deterministic guess for the instantiation of some $V_2 \in \{0,1\}^k$ and then we need to call a DP-oracle (or, equivalently, two NP-oracles) to check if the word V_1V_2 represents a repair. Thus ENUM_$R \in \mathsf{Del}\Sigma_2^P$ by Proposition 6, meaning that E-REPAIR $\in \mathsf{Inc}\Sigma_2^P$.

For the *hardness* proof, we first note that the $\mathsf{Inc}\Sigma_2^P$-completeness of E-Π_1SAT via I-reductions from Corollary 14 can be strengthened in the sense that we may restrict the instances to 3-DNF (i.e., the matrix of the formulæ under investigation is in DNF such that each implicant consists of 3 literals. Then our $\mathsf{Inc}\Sigma_2^P$-completeness proof is by an e-reduction from E-Π_1SAT, where we assume w.l.o.g., that the instances of E-Π_1SAT are in 3-DNF.

Consider an arbitrary instance of E-Π_1SAT, i.e., a quantified boolean formula $\psi(\vec{x}) = \forall\vec{y}\phi(\vec{x},\vec{y})$. Let $\vec{x} = x_1,\ldots,x_k$ and $\vec{y} = y_1,\ldots,y_\ell$. By the restriction to 3-DNF, $\phi(\vec{x},\vec{y})$ is of the form $\phi(\vec{x},\vec{y}) = \bigvee_{i=1}^{m}(l_{i1}\wedge l_{i2}\wedge l_{i3})$, where m denotes the number of implicants in ϕ and l_{ij} with $j\in\{1,2,3\}$ is the j-th literal in the i-th implicant, i.e., each l_{ij} is of the form x_α, $\neg x_\alpha$, y_β, or $\neg y_\beta$ with $\alpha\in\{1,\ldots,k\}$ and $\beta\in\{1,\ldots,\ell\}$.

From this we define an instance $\sigma(\psi(\vec{x})) = (D,C)$ of E-Repair. The relation symbols used in this instance are $\{a,q\}\cup\{p_1,\ldots,p_k\}\cup\{b_0,b_1\}$ It is convenient to define the following subformulæ:

$$\begin{aligned}
\chi &= b_0(v_0)\wedge b_1(v_1)\wedge q(v_0,v_1)\wedge q(v_1,v_0)\wedge\\
&\quad a(v_0,v_0,v_0)\wedge a(v_0,v_0,v_1)\wedge a(v_0,v_1,v_0)\wedge a(v_0,v_1,v_1)\wedge\\
&\quad a(v_1,v_0,v_0)\wedge a(v_1,v_0,v_1)\wedge a(v_1,v_1,v_0)\\
\pi &= \bigwedge_{i=1}^{k} p_i(w_i,w_i')\\
\psi^* &= \bigwedge_{i=1}^{m} a(l_{i1}^*,l_{i2}^*,l_{i3}^*)
\end{aligned}$$

where we define $l_{ij}^* = z$ if l_{ij} is the positive literal z and $l_{ij}^* = z'$ if l_{ij} is the negative literal $\neg z$. For example, $[(x\wedge\neg y\wedge z)\vee(\neg x\wedge y\wedge z)]^* = a(x,y'z)\wedge a(x',y,z)$.

Then the database D of our instance of E-Repair looks as follows:

$$\begin{aligned}
&D = P\cup Q\cup A\cup E \text{ with}\\
&P = \{p_1(0,1),p_1(1,0),p_2(0,1),p_2(1,0),\ldots,p_k(0,1),p_k(1,0)\}\\
&Q = \{q(0,1),q(1,0)\}\\
&A = \{a(0,0,0),a(0,0,1),a(0,1,0),a(0,1,1),a(1,0,0),a(1,0,1),a(1,1,0)\}\\
&E = \{b_0(0),b_1(1)\}
\end{aligned}$$

Finally, we define the set C of integrity constraints:

$$C = \bigcup_{1\le i\le k}\{\chi\wedge\pi\wedge p_i(x,x')\wedge p_i(x',x)\to x = x'\} \tag{C1}$$

$$\cup\{\chi\wedge\pi\wedge\bigwedge_{i=1}^{k}p_i(x_i,x_i')\wedge\bigwedge_{j=1}^{\ell}q(y_j,y_j')\wedge\psi^*\to x_1 = x_1'\} \tag{C2}$$

The intuition of this reduction is as follows: all of the EGDs in line (C1) are clearly violated, i.e.: χ can be sent into the database D by instantiating v_0 to 0 and v_1 to 1; likewise π can be sent into the database D by instantiating each pair (w_i,w_i') either to $(0,1)$ or to $(1,0)$. Finally, each of the atoms $p_i(x,x')$ and $p_i(x',x)$ can be sent into D by instantiating (x,x') to either $(0,1)$ or $(1,0)$. In either case, the left-hand side of the EGD is satisfied but the right-hand side is not. Hence, there are three possibilities to eliminate this EGD-violation:

(D1) we can delete one of the atoms in $Q\cup A\cup E$,

(D2) we can delete both atoms $p_i(0,1)$ and $p_i(1,0)$ for some $i\in\{1,\ldots,k\}$, or

(D3) we can delete one of the atoms $p_i(0,1),p_i(1,0)$ for every $i\in\{1,\ldots,k\}$.

Both (D1) as well as (D2) lead to a repair, as the subformulæ χ respectively π are always violated, so we need to consider (D3). The effect of retaining $p_i(0,1)$ with $i \in \{1,\ldots,k\}$ is that the variables (x_i, x_i') in line (C2) are instantiated to (0,1); likewise, the effect of retaining $p_i(1,0)$ is that the the variables (x_i, x_i') in line (C2) are instantiated to (1,0). Hence, there is a one-to-one correspondence between the choices of retained ground atom $p_i(.,.)$ for every $i \in \{1,\ldots,k\}$ and truth assignments to the x_i-variables in ψ, namely retaining $p_i(0,1)$ (resp. $p_i(1,0)$) corresponds to setting the propositional variable x_i in ψ to false (resp. true) and to set its dual $\neg x_i$, which is encoded by x_i' according to our definition of ψ^*, to true (resp. false). For a repair I that only uses deletions of the form (D3), denote by $\rho(I)$ the corresponding truth assignment on the propositional variables $\{x_1,\ldots,x_k\}$. We then define the relation τ for the e-reduction as follows:

$$\tau(\sigma(\psi(\vec{x})), I, -) = \begin{cases} \emptyset & : I \text{ is the result of a deletion of the form (D1) or (D2),} \\ \rho(I) & : \text{otherwise.} \end{cases}$$

To show that τ and σ indeed define a valid e-reduction from E-Π_1SAT, first note that $\tau(\sigma(\psi(x)), I, -)$ maps to the empty set at most $k+11$ times. So it suffices to show that the repairs we can get if we apply a deletion of the form (D3) are indeed models of ψ. We distinguish two cases: First suppose that the EGD in line (C2) is violated by the instance resulting from the deletion of exactly one of $p_i(0,1), p_i(1,0)$ for every $i \in \{1,\ldots,k\}$. Then we have to delete at least one more atom from D. This means that we either delete one atom from $Q \cup A \cup E$ to falsify χ (and possibly also further atoms in $q(y_i, y_i') \wedge \psi^*$) or we delete the remaining atom with leading symbol p_i to falsify atom $p_i(x_i, x_i')$ on the left-hand side of the EGD. In either case, the resulting database instance is not a repair due to the minimality condition.

It remains to consider the case that the EGD in line (C2) is satisfied by the instance I resulting from the deletion of exactly one of $p_i(0,1), p_i(1,0)$ for every $i \in \{1,\ldots,k\}$. Clearly, the equality on the right-hand side is always violated, no matter how we instantiate (x_1, x_1'), since the only options are $(0,1)$ and $(1,0)$. This means that the only way to satisfy the EGD in line (C2) is that there is no way to send the left-hand side into the database instance I. This means that no matter how we instantiate (y_j, y_j') – i.e., either to $(0,1)$ or to $(1,0)$ – there is always one atom in ψ^* that is instantiated to an a-atom outside I, namely $a(1,1,1)$. By our encoding of ψ by ψ^* this has the following meaning for the quantified boolean formula ψ: no matter how we extend the truth assignment $\rho(I)$ to an assignment on $\{y_1,\ldots,y_\ell\}$ (corresponding to the possible ways of instantiating (y_j, y_j') for every $j \in \{1,\ldots,\ell\}$), there is at least one implicant in ψ for which all three literals evaluate to true. In other words, there is a one-to-one correspondence between repairs obtained by deleting exactly one of $p_i(0,1), p_i(1,0)$ for every $i \in \{1,\ldots,k\}$ and models of ψ. □

7. 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 declarative type of reduction among enumeration problems which is transitive and under which the classes in our hierarchies are closed. Moreover, they turn out to be special cases of the procedural-style D- and I-reductions that we defined afterwards. These procedural style reductions allow one to get completeness results for the classes of our hierarchy. The e-reductions are conceptually simpler and easier to obtain; so they will be used to obtain further hardness results for enumeration problems. In

Enumeration Problem	Membership	Hardness via D-reductions	Hardness via I-reductions
E-Σ_kSAT Corollary 14	$\mathsf{DelP}^{\Sigma}_{k}$	$\mathsf{DelP}^{\Sigma}_{k}$	$\mathsf{IncP}^{\Sigma}_{k}$
E-Π_kSAT Corollary 14	$\mathsf{DelP}^{\Sigma}_{k+1}$	-	$\mathsf{IncP}^{\Sigma}_{k+1}$
E-Circumscription Theorem 16	$\mathsf{Del}^{+}\mathsf{NP}$	-	IncNP
E-CardMinSAT Theorem 17	DelNP	DelNP	IncNP
E-Model-Based Diagnosis Theorem 18	DelNP	DelNP	IncNP
E-Abduction Theorem 19	$\mathsf{DelP}^{\Sigma}_{2}$	$\mathsf{DelP}^{\Sigma}_{2}$	$\mathsf{IncP}^{\Sigma}_{2}$
E-Repair Theorem 20	$\mathsf{DelP}^{\Sigma}_{2}$	-	$\mathsf{IncP}^{\Sigma}_{2}$

Table 1: Summary of the main results for all $k \geq 0$.

this way we obtain completeness results for diverse problems such as generalized satisfiability in the Schaefer framework, circumscription, model-based diagnosis, abduction, and repairs.

Prior to our work, lower bounds for enumeration problems were only of the form "Enum_R is not in DelayP (or IncP) unless P = NP". We have provided here a framework which allows one to pinpoint the complexity of such problems in a better way in terms of completeness in our enumeration hierarchy.

As far as future work is concerned, of course, we want to close the gap left in Figure 1, i.e., clarifying the precise relationship between the enumeration complexity classes $\mathsf{Del}^{+}\Sigma^{P}_{k}$ and $\mathsf{Del}^{+}\Delta^{P}_{k+1}$. For a wider field of research activities in the area of hard enumeration problems, recall that our hierarchies of enumeration complexity classes build upon the tractable classes DelayP and IncP. Note that in [1], yet another notion of tractable enumeration is proposed, namely the class OutputP (also referred to as TotalP in [2]). A problem is in OutputP if the time needed to output all solutions to a given instance is bounded by a polynomial in the combined size of the input *plus* the output. It is easy to show that this class is closed w.r.t. e-reductions and only minor modifications of the D- and I-reductions are needed to close OutputP also under these reductions. However, while we have seen quite a close relationship between DelayP and IncP in this work, the class OutputP seems to behave very differently from the others. For instance, OutputP seems to be incomparable (under common complexity-theoretic assumptions) with *all* higher classes from our hierarchies built upon DelayP or IncP. Sure, we may also define a hierarchy of classes by allowing, for instance, a Σ^{P}_{k} oracle for some $k \geq 1$ on top of OutputP. However, it is unclear what complete problems for any of the resulting classes would look like. A major research effort is required to even get a basic understanding of such classes.

Finally, a natural next research target is to put the machinery developed here to work. In Section 6, we have already proved a few completeness results for our new complexity classes. Many more intractable enumeration problems (especially in the AI and database domains) wait for a precise complexity classification. We believe that such work – complementing the search for restrictions to make intractable enumeration problems tractable, cf. [31, 29] – is needed to get a better understanding of the true sources of complexity of hard enumeration problems.

Acknowledgments

We would like to thank Phokion Kolaitis for his encouragement on this work during his visit to Marseilles and for suggesting to consider database repairs as complete problems for our complexity classes.

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, AG-GREG project reference ANR-14-CE25-0017.

References

[1] D. S. Johnson, C. H. Papadimitriou, M. Yannakakis, On generating all maximal independent sets, Inf. Process. Lett. 27 (3) (1988) 119–123.

[2] Y. Strozecki, Enumeration complexity and matroid decomposition, Ph.D. thesis, Université Paris Diderot – Paris 7, available at `http://www.prism.uvsq.fr/~ystr/these_strozecki` (Dec. 2010).

[3] C. L. Lucchesi, S. L. Osborn, Candidate keys for relations, J. Comput. Syst. Sci. 17 (2) (1978) 270–279.

[4] B. Kimelfeld, P. G. Kolaitis, The complexity of mining maximal frequent subgraphs, ACM Trans. Database Syst. 39 (4) (2014) 32:1–32:33.
URL `http://doi.acm.org/10.1145/2629550`

[5] N. Creignou, H. Vollmer, Parameterized complexity of weighted satisfiability problems: Decision, enumeration, counting, Fundam. Inform. 136 (4) (2015) 297–316.
URL `http://dx.doi.org/10.3233/FI-2015-1159`

[6] N. Carmeli, B. Kenig, B. Kimelfeld, Efficiently enumerating minimal triangulations, in: Proc. PODS 2017, 2017, pp. 273–287.
URL `http://doi.acm.org/10.1145/3034786.3056109`

[7] G. Bagan, A. Durand, E. Grandjean, On acyclic conjunctive queries and constant delay enumeration, in: Computer Science Logic, Springer, 2007, pp. 208–222.

[8] A. Durand, N. Schweikardt, L. Segoufin, Enumerating answers to first-order queries over databases of low degree, in: Proc. PODS 2014, ACM, 2014, pp. 121–131.

[9] L. A. Hemaspaandra, H. Vollmer, The satanic notations: counting classes beyond #P and other definitional adventures, SIGACT News 26 (1) (1995) 2–13.

[10] S. Toda, PP is as hard as the polynomial-time hierarchy, SIAM J. Comput. 20 (5) (1991) 865–877.

[11] N. Creignou, M. Kröll, R. Pichler, S. Skritek, H. Vollmer, On the complexity of hard enumeration problems, in: Proc. LATA 2017, 2017, pp. 183–195.

[12] L. A. Hemachandra, The strong exponential hierarchy collapses, in: Proc. STOC 1987, ACM, 1987, pp. 110–122.

[13] Y. Strozecki, On enumerating monomials and other combinatorial structures by polynomial interpolation, Theory Comput. Syst. 53 (4) (2013) 532–568.

[14] S. Arora, B. Barak, Computational Complexity: A Modern Approach, Cambridge University Press, Cambridge, 2009.

[15] J. Schmidt, Enumeration: Algorithms and complexity, Master's thesis, Leibniz Universität Hannover (2009).

[16] A. Durand, E. Grandjean, First-order queries on structures of bounded degree are computable with constant delay, ACM Trans. Comput. Log. 8 (4).
URL `http://doi.acm.org/10.1145/1276920.1276923`

[17] J. Brault-Baron, 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).
URL `https://tel.archives-ouvertes.fr/tel-01081392`

[18] A. Mary, Enumération des dominants minimaux d'un graphe, Ph.D. thesis, Université Blaise Pascal, Clermont-Ferrand, France (2013).

[19] M. M. Kanté, V. Limouzy, A. Mary, L. Nourine, On the enumeration of minimal dominating sets and related notions, SIAM J. Discrete Math. 28 (4) (2014) 1916–1929.
URL `http://dx.doi.org/10.1137/120862612`

[20] N. Creignou, J.-J. Hébrard, On generating all solutions of generalized satisfiability problems, Informatique théorique et applications 31 (6) (1997) 499–511.

[21] T. J. Schaefer, The complexity of satisfiability problems, in: Proc. STOC 1978, ACM Press, 1978, pp. 216–226.

[22] T. Eiter, G. Gottlob, The complexity of logic-based abduction, J. ACM 42 (1) (1995) 3–42.

[23] A. Durand, M. Hermann, P. G. Kolaitis, Subtractive reductions and complete problems for counting complexity classes, Theor. Comput. Sci. 340 (3) (2005) 496–513.

[24] R. Reiter, A theory of diagnosis from first principles, Artificial Intelligence 32 (1) (1987) 57–95.

[25] J. Marques-Silva, M. Janota, A. Ignatiev, A. Morgado, Efficient model based diagnosis with maximum satisfiability, in: Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015, AAAI Press, 2015, pp. 1966–1972.

[26] N. Creignou, B. Zanuttini, A complete classification of the complexity of propositional abduction, SIAM J. Comput. 36 (1) (2006) 207–229.
URL `http://dx.doi.org/10.1137/S0097539704446311`

[27] M. Arenas, L. Bertossi, J. Chomicki, Consistent query answers in inconsistent databases, in: Proc. PODS 1999, ACM Press, 1999, pp. 68–79.

[28] S. Abiteboul, R. Hull, V. Vianu, Foundations of Databases, Addison-Wesley, Reading, Massachusetts, 1995.

[29] E. Livshits, B. Kimelfeld, Counting and enumerating (preferred) database repairs, in: Proc. PODS 2017, ACM, 2017, pp. 289–301.
URL `http://doi.acm.org/10.1145/3034786.3056107`

[30] S. Arming, R. Pichler, E. Sallinger, Complexity of repair checking and consistent query answering, in: Proc. ICDT, 2016, pp. 21:1–21:18.
URL `https://doi.org/10.4230/LIPIcs.ICDT.2016.21`

[31] M. Kröll, R. Pichler, S. Skritek, On the complexity of enumerating the answers to well-designed pattern trees, in: Proc. ICDT 2016, Vol. 48 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016, pp. 22:1–22:18.