Monotonic Enumerations of Complete Sets

Albrecht Hoene

Technical Report 91-02-02

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

University of Washington

Seattle 98195

Monotonic Enumerations of Complete Sets

Albrecht Hoene

Technical Report 91-02-02

Department of Computer Science and Engineering, FR-35
University of Washington, Seattle, WA 98195 USA

Monotonic Enumerations of Complete Sets

Albrecht Hoene *
Technische Universität Berlin
Fachbereich Informatik
Franklinstr. 28/29
1000 Berlin 10, Germany

January 27, 1991

Abstract

The following notion of polynomial-time enumerability was suggested by Hemachandra, Hoene, Siefkes, and Young. A set L is *polynomially enumerable by iteration* if there exists a polynomial-time computable function f and a string x such that $L = \{x, f(x), f(f(x)), \dots\}$. L is *monotonically enumerable by iteration* if the function f is lexicographically increasing. In this paper we show that many complexity classes between P and PSPACE have monotonically enumerable complete set. For the counting hierarchy we exhibit a connection between the quantifier structure of its classes and the existence of monotonically enumerable complete sets. Every class of the form $\forall K$, and $\mathsf{C}K$ in this hierarchy is shown to possess monotonically enumerable complete sets. However, classes of the form $\exists K$ have monotonically enumerable complete sets if and only if they are closed under complement, which shows a striking difference between the existential quantifier and the other two quantifiers. In particular, it holds that for all $k \geq 1$ the class Π_k^p has monotonically enumerable complete sets, whereas Σ_k^p has monotonically enumerable complete sets if and only if the polynomial-time hierarchy collapses to its kth level. Moreover, we give complete characterizations of PSPACE and the class $\oplus$OptP in terms of monotonic enumeration functions.

1 Introduction

Approaches to gain insight into the nature of the still unsolved P $\stackrel{?}{=}$ NP $\stackrel{?}{=}$ PSPACE-problem mostly deal with elaborated or refined notions of nondeterministic computations. Examples are

*Research supported by a Deutsche Forschungsgesellschaft Postdoktorandenstipendium. The author's current address is: *Department of Computer Science and Engineering, FR-35, University of Washington, Seattle, WA 98195.*

the polynomial-time hierarchy ([19]), modified acceptance criteria for nondeterministic Turing machines ([5, 17, 9]) and the counting hierarchy ([21]). However, recently in a variety of papers a broader approach has been taken. Instead of studying solely the membership problem for such classes it has been asked which properties seemingly complex sets possess that can be efficiently computed. Among the questions that have been studied in this context are: Which complex sets are paddable ([3]), which sets are efficiently near-testable ([8, 7]), which sets are efficiently hashable ([6]), and which sets have efficient implicit membership tests ([11]) (for an overview see [10]). Going further it has been tried to completely characterize classes that are defined in terms of nondeterminism by other than nondeterminism-properties—thus trying to get a new understanding of nondeterminism. Here two recent examples are the class of polynomial-time near-testable sets ([8, 7]) and the class of polynomial-time nearly near-testable sets ([11]), which yielded complete characterizations for the classes $\oplus$P and $\oplus$OptP, respectively.

A natural problem arising in this context is the question: Which sets have efficient *enumeration* algorithms? But even though in recursive function theory the notion of enumerability is robust under a multitude of intuitively appealing definitions, things turn out to be more involved in the polynomial-time bounded setting (a discussion of definitions for polynomial enumerability suggested in the literature can be found in [13]). In [13] (see also [12]) the following definition was introduced: A set L is polynomially enumerable by iteration if for a polynomial-time computable function f and some string x the set L can be enumerated by iterative application of f on x, i.e., $L = \{x, f(x), f(f(x)), \ldots\}$. The results of [13] show that all exponential-time complete sets and all so far constructed NP-complete sets are polynomially enumerable by iteration.

In this paper we study monotonic enumerations, that is, we consider sets that are polynomially enumerable via lexicographically increasing enumeration functions. Sets that are monotonically enumerable in this sense at first glance seem to have a very easy structure. In [13] both positive and negative results were obtained on this issue. It was shown that there are complete sets in $\oplus$P that are monotonically bi-enumerable (that is, the enumeration function not only enumerates the set itself but is also an enumeration function for the set's complement), and that every monotonically bi-enumerable set is contained in $\oplus$P. However, it was also shown that the Σ_k^p-classes in the polynomial-time hierarchy are not likely to have monotonically enumerable complete sets—as their existence would imply the collapse of the polynomial-time hierarchy. In Section 3 we address the question which other classes than the aforementioned between P and PSPACE do have monotonically enumerable complete sets. We show that inspite of the negative result for Σ_k^p for all k the Π_k^p-classes do have monotonically enumerable complete sets. In fact, we show a far more general result: we consider Wagner's counting hierarchy ([21]), which, roughly speaking, consists of all classes that can be obtained from P by iterative application of the $\exists$-, $\forall$-quantifier, and the counting quantifier C_f (for precise definitions see Section 2). Our main

theorem of Section 3 is that all classes in the counting hierarchy that are not of the form $\exists C$ for some C in fact do have complete sets that are monotonically enumerable. However, for each class of the form $\exists C$ in this hierarchy it holds that it has monotonically enumerable complete sets if and only if $\exists C$ is closed under complement. Thus, compared to the $\forall-$ and C_f-quantifier the existential quantifier seems to have a quite different nature.

In Section 4 give two complete characterizations of complexity classes in terms of monotonic enumeration functions. As mentioned above, one such characterization was given in [13], namely that a set L is in $\oplus$P if and only if L is polynomial-time reducible to some monotonically bi-enumerable set. Here we obtain complete characterizations for PSPACE and $\oplus$OptP, where the latter class has been introduced and studied in [11]. $\oplus$OptP can be viewed as a $\oplus$P-computation with a functional Δ_2^p-precomputation, and is closely related to the class of polynomial-time nearly near-testable sets (for definitions see [11]).

For PSPACE we show that PSPACE is exactly the class of languages that are polynomially reducible to some monotonically enumerable set, which in turn coincides with the class of sets that are polynomially reducible to some monotonically enumerable set with a one-one polynomial-time invertible enumeration function. For $\oplus$OptP we give a characterization in terms of enumeration functions that are weakly bi-enumerable in a certain sense. We introduce piecewise monotonically bi-enumerable sets, and show that a set is in $\oplus$OptP if and only if it is polynomially reducible to some piecewise monotonically bi-enumerable set.

In the next section we give definitions and collect a couple of basic facts on monotonically enumerable sets.

2 Basics

We use the standard alphabet $\Sigma = \{0,1\}^*$. For each string $w \in \Sigma^*$ by w^+ we denote its lexicographic successor and for $w \in \Sigma^+$ by w^- its lexicographic predecessor. $w \cdot b$ and wb stand for the string w appended by bit b. For comparisons between strings with respect to the lexicographical ordering we write $x <_{lex} y$, or $x \leq_{lex} y$ respectively. "max" and "min" applied to sets of strings refer to the lexicographical ordering. We denote the natural numbers by $\mathbb{N}$. Frequently we will interpret strings as binary encodings of natural numbers: for $w \in \Sigma^*$ we define $\dot{w}$ to be $\|\{v \in \Sigma^* : |v| = |w| \text{ and } v <_{lex} w\}\|$.

In our proofs we will need two types of pairing functions, the first of which has to be sensitive to the lexicographical ordering and maps tupels of a special format bijectively to Σ^*. For a polynomial p satisfying $n < m \Rightarrow p(n) < p(m)$ for all n, m[1] and some fixed k such a function will code the set $U_k^p = \{(x, x_1, \ldots, x_k) | \; |x_i| = p(|x|),\ 1 \leq i \leq k\}$ into $\{0,1\}^*$ and preserve the

[1] Throughout the paper we will implicitly assume that all the polynomials have this property.

natural lexicographical ordering between the $k+1$-tupels[2] (see also [13, 11]). For $k+1$-tupels not in U_k^p the function $pair_k^p$ is undefined. For $k+1$-tupels $(x, x_1, x_2, \ldots, x_k) \in U_k^p$ we define

$$pair_k^p(x, x_1, x_2, \ldots, x_k) = x x_1 x_2 \ldots x_k$$

and

$$[x, x_1, x_2, \ldots, x_k]_k^p = \text{the } i\text{-th string in the lexicographical ordering in } \{0,1\}^* \text{ if } pair_k^p(x, x_1, x_2, \ldots, x_k) \text{ is the lexicographically } i\text{-the string in the image of } U_k^p \text{ under } pair_k^p.$$

For any polynomial p satisfying the above requirements the function $[x, x_1, x_2, \ldots, x_k]_k^p : U_k^p \to \{0,1\}^*$ is polynomial-time computable, polynomial-time invertible and maps U_k^p to $\{0,1\}^*$ in a one-to-one, onto fashion. If the polynomial p is clear from the context we write $[\cdot, \ldots, \cdot]_k$ instead of $[\cdot, \ldots, \cdot]_k^p$.

If the strings to be paired do not fulfill the special length requirements above then we will make use of a special separation character ("#"), e.g., to pair the strings x and y we write $x\#y$. In proofs that employ this kind of pairing function the result will not depend on the lexicographical ordering.

In a variety of cases we will consider the disjoint union of two sets A and B which we define by

$$A \uplus B = \{x0 | x \in A\} \cup \{y1 | y \in B\}.$$

For complexity classes we use standard notation as P, NP, PSPACE, and so on (see [15]). For a class of languages C we denote by coC the class of languages that are complements of languages in C. If we talk about reductions we will always mean polynomial-time many-one reductions ($\leq_m^p$), and completeness will always be understood with respect to this kind of reduction.

The counting quantifiers (C^p, G^p) and the counting hierarchy were introduced in [21]—generalizing ideas of [5, 18] (see also [20]).

Definition 2.1 *For a polynomial-time computable function $f : \Sigma^* \to I\!N$, a polynomial p, and a predicate $R(x,y)$ the quantifiers G_f^p, C_f^p and $\oplus^p$ are defined by*

1. $\mathsf{C}_f^p\, y\ R(x,y) \iff \|\{y : |y| = p(|x|) \text{ and } R(x,y)\}\| > f(x)$
2. $\mathsf{G}_f^p\, y\ R(x,y) \iff \|\{y : |y| = p(|x|) \text{ and } R(x,y)\}\| = f(x)$

In the following we will drop the index p from the quantifiers.

Definition 2.2 *Let K be any class of languages.*

[2] That is $(x_1, x_2, \ldots, x_k) < (y_1, y_2, \ldots, y_k)$ iff $(\exists i : 0 \leq i < k)[x_j = y_j \text{ for } 1 \leq j \leq i \text{ and } x_{i+1} <_{lex} y_{i+1}]$.

1. *A set L is in $\mathsf{C}K$ if there is a polynomial-time computable function $f : \Sigma^* \to I\!N$, a polynomial p and a predicate $R(x,y) \in K$ such that $x \in L \iff \mathsf{C}_f\, y\ R(x,y)$ holds.*

2. *A set L is in $\mathsf{G}K$ if there is a polynomial p and a predicate $R(x,y) \in K$ such that $x \in L \iff \mathsf{G}_f\, y\ R(x,y)$ holds.*

In the same sense we will use $\exists$ and $\forall$ (see [1]).

Definition 2.3 *[21] The* polynomial counting hierarchy (CH) *is the smallest family of language classes that satisfies*

1. $\mathrm{P} \in \mathrm{CH}$, *and*

2. *If* $K \in \mathrm{CH}$ *then* $\exists K \in \mathrm{CH}$, $\forall K \in \mathrm{CH}$, *and* $\mathsf{C}K \in \mathrm{CH}$.

All of the classes in CH were shown to have complete problems in [21].

We will also consider the class $\oplus$OptP, which is closely connected to the class of nearly near-testable sets, which was introduced and studied in [11]. Let OptP be Krentel's class of polynomial-time computable optimization functions (for a definition see [16] or [11]).

Definition 2.4 *[11] A set L is in $\oplus$OptP if there exists a function $f : \Sigma^* \to \Sigma^*$, $f \in \mathrm{OptP}$ and a nondeterministic polynomial-time Turing machine N such that*

$$x \in L \iff N \text{ accepts } f(x) \text{ on an odd number of computation paths.}$$

For the characterization of this class we will use the the following fact ([11, Lemma 2.1]):

Proposition 2.5 *For a language L the following are equivalent:*

1. $L \in \oplus\mathrm{OptP}$

2. *There is a nondeterministic polynomial-time Turing machine N, a polynmomial p and a predicate $R(x,y) \in \mathrm{P}$ such that for all x and $z = \max\{y : |y| = p(|x|) \text{ and } R(x,y)\}$*

$$x \in L \iff N \text{ accepts } [x,z]_2 \text{ on an odd number of computation paths.}$$

3. *There is a function $f \in \mathrm{FP}^{\mathrm{NP}}$, i.e., f is computable by a polynomial-time Turing machine that has access to an NP-oracle, and a nondeterministic polynomial-time Turing machine N such that*

$$x \in L \iff N \text{ accepts } f(x) \text{ on an odd number of computation paths.}$$

We now define the notion of enumerability we study more formally and collect a couple of basic facts and previous results on this issue.

Definition 2.6 *[13]*

1. *A function* $f : \Sigma^* \to \Sigma^*$ *is* monotonic *if* $x <_{lex} f(x)$ *for all* $x \in \Sigma^*$.

2. *A set* L *is* (polynomially) monotonically enumerable by iteration *(short:* monotonically enumerable*) if* L *is finite or cofinite or there exists a polynomial-time computable monotonic function* f *and a string* x *such that*

$$L = \{x, f(x), f(f(x)), \ldots\}.$$

   f *is called the* enumeration function.

3. *A set* L *is* (polynomially) monotonically bi-enumerable by iteration *(short:* monotonically bi-enumerable*) if* L *is finite, cofinite, or* L *is monotonically enumerable via a function* f *and there exist strings* x *and* y *such that*

$$L = \{x, f(x), f(f(x)), \ldots\} \quad \text{and} \quad \overline{L} = \{y, f(y), f(f(y)), \ldots\}.$$

Frequently we will call sequences of the form $v, f(v), f(f(v)), \ldots$ "chains (started at v)".

It is immediate that any set that is monotonically enumerable is contained in PSPACE. A straightforward construction shows that there exist sets in P that are not monotonically enumerable. In [13] the degree of difficulty of monotonically bi-enumerable sets was determined.

Theorem 2.7 *[13]*

1. *Every monotonically bi-enumerable set is in* $\oplus$P.

2. *There is a* $\oplus$P*-complete set that is monotonically bi-enumerable.*

It follows that a set L is $\leq^{\mathrm{P}}_m$-reducible to some monotonically bi-enumerable set if and only if L is in $\oplus$P. In the following sections we will exhibit more classes that have monotonically enumerable complete sets. However, the next proposition shows that it is unlikely that all complete sets of any of these classes are monotonically enumerable as it would imply that they are equal to P (see also [7]).

Proposition 2.8 *For any set* L *there exists a set* $\hat{L}$ *such that*

1. $L \equiv^{\mathrm{P}}_m \hat{L}$ *and*

2. *If* $\hat{L}$ *is monotonically enumerable then* $\hat{L} \in$ P

Proof For a set $L \neq \emptyset$, Σ^* let $\hat{L}$ be defined by $\Sigma^* \uplus L$. Clearly, $L \equiv_m^p \hat{L}$. If f is a monotonic enumeration function for $\hat{L}$ then membership in $\hat{L}$ for some string y can be tested in the following way: if $y = y'0$ for some y' then $y \in \hat{L}$. Otherwise $y = y'1$ for some y' and it holds that $y \in \hat{L}$ if and only if $f(y'0) = y$. □

Now the following is immediate:

Corollary 2.9 *Let $C \supseteq \mathrm{P}$ be a class that has complete sets and is closed downwards under $\leq_m^p$-reductions. If every complete problem for C is monotonically enumerable then $C = \mathrm{P}$.*

3 Monotonically Enumerable Complete Sets

Which classes between P and PSPACE do have monotonically enumerable complete sets? From [13] we know that, e.g., for any $k \geq 1$ it is not likely that Σ_k^p has monotonically enumerable complete sets as it would imply $\mathrm{PH} = \Sigma_k^p$. However, in this section we show that many classes of the counting hierarchy do have complete sets that are monotonically enumerable. In [13] it was shown that there exists a monotonically enumerable coNP-complete set (e.g., under an appropriate encoding $\{(x, y) |\ x$ is a Boolean formula that has no satisfying assignment greater or equal to $y\}$ provides such a set). Our main theorem of this section is the following:

Theorem 3.1 *Let K be any class of the counting hierarchy.*

1. *If $C = \forall K$, or $C = \complement K$ then C has monotonically enumerable complete sets.*
2. *If $C = \exists K$ then C has monotonically enumerable complete sets if and only if C is closed under complement.*

As an immediate corrolary we obtain:

Corollary 3.2 *Let $k \geq 1$.*

1. *There exist Π_k^p-complete sets that are monotonically enumerable.*
2. *([13]) There is a monotonically enumerable Σ_k^p-complete set if and only if $\mathrm{PH} = \Sigma_k^p$.*

Before we give the proof of Theorem 3.1 we illustrate our basic enumeration technique by giving two examples.

Example 3.3 *There is a set that is monotonically enumerable and complete for Π_2^p.*

Proof Let L be complete for Π_2^p, p a polynomial and $R(x,y,z)$ a polynomial-time predicate such that

$$x \in L \iff \forall y \exists z\ R(x,y,z).$$

Here and throughout the rest of the proof we assume that all quantifiers range over strings of exact length $p(|x|)$ and $R(x,y,z)$ holds only for triples that satisfy $|y| = |z| = p(|x|)$. We define

$$\hat{L}_1 = \{[x,y,z]_3 :\ \forall y' <_{lex} y\ \exists z'\ R(x,y',z') \text{ and } \forall z' <_{lex} z\ \neg R(x,y,z')\},$$

$$\hat{L}_2 = \{[x, 1^{p(|x|)}, 1^{p(|x|)}]_3 :\ x \in L\},$$

and

$$\hat{L} = \hat{L}_1 \uplus \hat{L}_2.$$

We claim that $\hat{L}$ is complete for Π_2^p and monotonically enumerable. For the completeness note that $\hat{L}$ is in Π_2^p, since $\hat{L}_1$ and $\hat{L}_2$ are in Π_2^p, which is closed under disjoint union. Moreover, L trivially reduces to $\hat{L}$ via the function $g(x) = [x, 1^{p(|x|)}, 1^{p(|x|)}]_3 \cdot 1$.

It remains to show that $\hat{L}$ is monotonically enumerable. First note that for all x the string $[x, 0^{p(|x|)}, 0^{p(|x|)}]_3$ belongs to $\hat{L}_1$, which encodes all triples (x,y,z) such that for all y' smaller than y a z' has been found such that $R(x,y',z')$ holds, and for y a z satisfying $R(x,y,z)$ has not yet been found. $\hat{L}_1$ permits a polynomial-time computable function to search in a monotonic enumerating fashion for a conditional witness proving $x \in L$ or $x \notin L$. For the case $x \in L$ such a conditional witness is a string of the form $[x, 1^{p(|x|)}, z]_3$ such that $R(x, 1^{p(|x|)}, z)$ holds: such a string witnesses $x \in L$ under the condition that the enumeration function h, started at $[x, 0^{p(|x|)}, 0^{p(|x|)}]_3$, reaches by iterative application the string $[x, 1^{p(|x|)}, z]_3$. The definition of the function h—as given below—makes sure that such a conditional witness $[x, 1^{p(|x|)}, z]_3$ is reached by the chain $[x, 0^{p(|x|)}, 0^{p(|x|)}]_3, h([x, 0^{p(|x|)}, 0^{p(|x|)}]_3), \ldots$ if and only if $x \in L$.

For the case $x \notin L$ a conditional witness will be of the form $[x, y, 1^{p(|x|)}]_3$ such that $R(x, y, 1^{p(|x|)})$ does not hold; under the condition that the enumeration function h reaches by iterative application the string $[x, y, 1^{p(|x|)}]_3$ from $[x, 0^{p(|x|)}, 0^{p(|x|)}]_3$ it witnesses that for this y there is no z such that $R(x,y,z)$ holds, and thus by definition $x \notin L$.

The enumeration function h is specified as follows: on all inputs of the form $[x,y,z]_3 \cdot 1$ it is defined by $h([x,y,z]_3 \cdot 1) = [x^+, 0^{p(|x^+|)}, 0^{p(|x^+|)}]_3 \cdot 0$. This is correct, since for all x there is at most one string of the form $[x,y,z]_3 \cdot 1$, which may belong to $\hat{L}$ ($[x, 1^{p(|x|)}, 1^{p(|x|)}]_3 \cdot 1$) and its lexicographical successor (in $\hat{L}$) is $[x^+, 0^{p(|x^+|)}, 0^{p(|x^+|)}]_3 \cdot 0$.

If for inputs of the form $[x,y,z]_3 \cdot 0$ a conditional witness is found the corresponding element of $\hat{L}_2$ is accessed or omitted, that is,

$$h([x,y,z]_3 \cdot 0) = \begin{cases} [x, 1^{p(|x|)}, 1^{p(|x|)}]_3 \cdot 1 & \text{for } y = 1^{p(|x|)} \text{ and } R(x,y,z) \\ [x^+, 0^{p(|x^+|)}, 0^{p(|x^+|)}]_3 \cdot 0 & \text{for } z = 1^{p(|x|)} \text{ and } \neg R(x,y,z) \end{cases}$$

Note that for any string $[x,y,z]_3 \cdot 0 \in \hat{L}$, for which one of the two conditions above holds, there does not exist a $[x,y',z']_3 \cdot 0$ that is lexicographically greater than $[x,y,z]_3 \cdot 0$ and belongs to $\hat{L}$ as well.

In the remaining cases the search for a witness is continued by mapping the input $[x,y,z]_3 \cdot 0$ to the lexicographically next string w that satisfies "$[x,y,z]_3 \cdot 0 \in \hat{L}$ implies $w \in \hat{L}$". We define

$$h([x,y,z]_3 \cdot 0) = \begin{cases} [x,y,z^+]_3 \cdot 0 & \text{for } z \neq 1^{p(|x|)} \text{ and } \neg R(x,y,z) \\ [x,y^+,0^{p(|x|)}]_3 \cdot 0 & \text{for } y \neq 1^{p(|x|)} \text{ and } R(x,y,z) \end{cases}$$

Clearly, this function is computable in polynomial-time. The special form of our pairing function $[\cdot,\cdot,\cdot]_3$ makes sure that h is lexicographically increasing. □

The next example will show that there are GP-complete monotonically enumerable sets. Formally, this class does not belong to the counting hierarchy, but the enumerating techniques are essentially the same as for the C−quantifier. First we note that without loss of generality the quantifiers can have the following special form, which will be needed in the proof of Theorem 3.1.

Proposition 3.4 *Let K be a class in the counting hierarchy. If $L \in CK$ ($L \in GK$, respectively) then there is a polynomial p and a predicate $R(x,y) \in K$ such that for the function $f(x) = 2^{1/2 \cdot p(|x|)-1}$*

1. $x \in L \iff C_f\, y\ R(x,y)$ *(*$G_f\, y\ R(x,y)$ *for* GK*).*
2. $\forall x,y\,[R(x,y) \Rightarrow (y = y_1 y_2,\ |y_1| = |y_2| = 1/2 \cdot p(|x|),$ *and* $y_2 = 0^{1/2 \cdot p(|x|)})]$

Example 3.5 *There is a set that is monotonically enumerable and complete for* GP.

Proof Let L be a set that is complete for GP, p be a polynomial, and $R(x,y)$ be a polynomial-time computable predicate such that the requirements of Proposition 3.4 are satisfied, i.e.,

$$x \in L \iff \|\{y:\ |y| = p(|x|) \text{ and } R(x,y)\}\| = 2^{1/2 \cdot p(|x|)-1}$$

holds. The set $\hat{L}$ defined by

$$\hat{L} = \{[x,y,n]_3 :\ \text{there are exactly } n \text{ strings } y' \text{ such that } |y'| = |y|,\ y' <_{lex} y \text{ and } R(x,y)\}$$

is monotonically enumerable and complete in GP. It is monotonically enumerable, since the polynomial-time computable function

$$h([x,y,n]_3) = \begin{cases} [x,y^+,n^+]_3 & \text{for } y \neq 1^{p(|x|)} \text{ and } R(x,y) \\ [x,y^+,n]_3 & \text{for } y \neq 1^{p(|x|)} \text{ and } \neg R(x,y) \\ [x^+,0^{p(|x^+|)},0^{p(|x^+|)}]_3 & \text{otherwise} \end{cases}$$

is monotonic, and—started at $[\lambda, 0^{p(0)}, 0^{p(0)}]_3$—$h$ enumerates $\hat{L}$. Clearly, $\hat{L}$ is in GP. Note that $R(x, 1^{p(|x|)})$ does not hold. Thus, for $w \in \Sigma^*$ such that $|w| = p(|x|)$ and $\dot{w} = 2^{1/2 \cdot p(|x|)-1}$

$$x \in L \iff [x, 1^{p(|x|)}, w]_3 \in \hat{L}$$

holds. Hence $\hat{L}$ is complete for GP, since L is complete for GP. □

Proof of Theorem 3.1

1. To show the claim we will do the following: for some class C of the counting hierarchy of the form $Q_1 \ldots Q_k$P and some complete language $L \in C$, we will construct a language $\hat{L}$ that is monotonically enumerable. Moreover L will be reducible to $\hat{L}$, and if $Q_1 \neq \exists$ then $\hat{L}$ will be in C, thus proving the theorem.

Now fix some $k \geq 1$ and let $C = Q_1 Q_2 \ldots Q_k$P be a class of the counting hierarchy, i.e., $Q_i \in \{\exists, \forall, \mathsf{C}\}$ for $1 \leq i \leq k$, and let L be complete for C. There is a polynomial p and a $k+1$-ary predicate $R(x, x_1, \ldots, x_k) \in$ P such that

$$x \in L \iff Q_1 x_1 Q_2 x_2 \ldots Q_k x_k \, R(x, x_1, \ldots, x_k).$$

Again, throughout the proof we assume that all quantifiers Q_i range over strings that have exactly length $p(|x|)$. Without loss of generality we can assume that p and R are chosen such that for all $Q_i = \mathsf{C}$ the requirements from Proposition 3.4 are satisfied.

By fixing the first i variables, from L we obtain the languages L_i:

$$L_i = \{[x, x_1, \ldots, x_i]_i : \; Q_{i+1} x_{i+1} \ldots Q_k x_k \, R(x, x_1, \ldots, x_k)\},$$

and for $1 \leq i \leq k$ it holds that $L_i \in Q_{i+1} Q_{i+2} \ldots Q_k$P.

We now construct the language $\hat{L}$ that will be monotonically enumerable and complete for C if $Q_1 \neq \exists$. As in example 3.3, $\hat{L}$ will be the disjoint union of two sets $\hat{L}_1$ and $\hat{L}_R$. The latter one has the same role as the language $\hat{L}_2$ in example 3.3, namely to permit a simple reduction from L to $\hat{L}$. This language is defined by

$$\hat{L}_R = \{[x, 1^{p(|x|)}, \ldots, 1^{p(|x|)}]_k : \; x \in L\}$$

Similar to $\hat{L}_1$ in example 3.3, the sets $\hat{L}_i$ are defined to facilitate an easy search by a lexicographically increasing function for conditional witnesses proving $[x, x_1, \ldots, x_i]_i \in L_i$ or $[x, x_1, \ldots, x_i]_i \notin L_i$. These languages will be defined inductively and depend on the quantifier structure for L. There are three different cases. Let $\hat{L}_{k+1} = \Sigma^*$. For $1 \leq i \leq k$ we define

$Q_i = \forall$:

$$\hat{L}_i = \{[x, x_1, \ldots, x_k]_k : \\ \forall x_i' <_{lex} x_i \, [x, x_1, \ldots x_{i-1}, x_i']_i \in L_i \text{ and} \\ [x, x_1, \ldots, x_k]_k \in \hat{L}_{i+1}\}$$

$Q_i = \exists$:

$$\hat{L}_i = \{[x, x_1, \ldots, x_k]_k : \forall x_i' <_{lex} x_i [x, x_1, \ldots x_{i-1}, x_i']_i \notin L_i \text{ and } [x, x_1, \ldots, x_k]_k \in \hat{L}_{i+1}\}$$

$Q_i = \mathsf{C}$:

$$\hat{L}_i = \{[x, x_1, \ldots, x_k]_k : x_i = yn, |y| = |n| \text{ and there are exactly } \dot{n} \text{ strings } y' \text{ such that } |y'| = |y|, y' <_{lex} y \text{ , and } [x, x_1, \ldots, x_{i-1}, y'0^{1/2 \cdot p(|x|)}]_i \in L_i \text{ and } [x, x_1, \ldots, x_{k-1}, y0^{1/2 \cdot p(|x|)}]_k \in \hat{L}_{i+1}\}$$

Now let $\hat{L} = \hat{L}_1 \uplus \hat{L}_R$. To show that $\hat{L}$ is complete for C first note that

$$x \in L \iff [x, 1^{p(|x|)}, \ldots, 1^{p(|x|)}] \cdot 1 \in \hat{L}$$

holds. For $Q_1 \neq \exists$, $\hat{L}$ is contained in C, since $\hat{L}$ is the disjoint union of two C languages: it is easily seen that $\hat{L}_R$ is contained in C, and $\hat{L}_1 \in C$ follows from the fact that $\hat{L}_i \in Q_i Q_{i+1} \ldots Q_k \mathrm{P}$ for $Q_i \neq \exists$, which we will prove below. First note that for any $K \in \mathrm{CH}$ it holds that $\mathsf{G}K, \forall K$, and $\exists K$ are closed under intersection ([20]), and $\mathsf{G}K \subset \mathsf{C}K$ ([20, Corollary 10]). We show that for $1 \leq i \leq k$ and

$$Q_i = \exists : \quad \hat{L}_i \in \forall co Q_{i+1} \ldots Q_k \mathrm{P}$$

$$Q_i = \forall, \mathsf{C} : \ \hat{L}_i \in Q_i \ldots Q_k \mathrm{P}.$$

by backward induction on i. For the case $i = k$ the claim is easily seen to be true.

Suppose the claim holds for $i+1$ with $1 \leq i < k$, and let $K = Q_{i+1} \ldots Q_k \mathrm{P}$. $\hat{L}_i$ can be written as the intersection of some language S and $\hat{L}_{i+1}$, where $S \in \forall K$ for $Q_i = \forall$, $S \in co\exists K$ for $Q_i = \exists$, and $S \in \mathsf{G}K$ for $Q_i = \mathsf{C}$. For $Q_{i+1} = \forall, \mathsf{C}$ the claim holds by the induction hypothesis $\hat{L}_{i+1} \in K$, and the fact that $\mathsf{G}K, \forall K$, and $\exists K$ are closed under intersection.

For $Q_{i+1} = \exists$, $\hat{L}_{i+1}$ is in $\forall co Q_{i+2} \ldots Q_k \mathrm{P}$. With $\mathsf{C}K' = co\mathsf{C}K'$ for all $K' \in \mathrm{CH}$ it follows that $co Q_{i+2} \ldots Q_k \mathrm{P} \subseteq Q_{i+1} \ldots Q_k \mathrm{P}$, and thus $\hat{L}_{i+1} \in \forall K$. Hence for $Q_i = \mathsf{C}$ we get $\hat{L}_{i+1} \in \mathsf{G}K$ (using $\forall K \subseteq \mathsf{G}K$), and thus $\hat{L}_i \in Q_i \ldots Q_k \mathrm{P}$. For $Q_i = \forall$ the argument is analogous.

Showing that $\hat{L}$ indeed is monotonically enumerable requires somewhat more work. We first prove in Lemma 3.6 that there is a uniform polynomial-time computable function $\hat{f}_1$ that—started at the corresponding minimal element—enumerates the words in $\hat{L}_1(z)$ for each individual z. $\hat{L}_1(z)$ consists of the strings in $\hat{L}_1$ that are of the form $[z, x_1, \ldots, x_k]_k$. Moreover, if the input to $\hat{f}_1$ is the lexicographically biggest string in $\hat{L}_1(z)$ then the enumeration function will have the property that it returns the answer to the question "$z \in L$".

More generally, for each $0 \leq i < k$ and $(z, z_1, \ldots, z_i)$ such that $|z_j| = p(|z|)$ for $1 \leq j \leq i$ let

$$\hat{L}_{i+1}(z, z_1, \ldots, z_i) = \{[z, z_1, \ldots z_i, x_{i+1}, \ldots, x_k]_k : [z, z_1, \ldots z_i, x_{i+1}, \ldots, x_k]_k \in \hat{L}_{i+1}\}$$

Note that the use of $[\cdots]_k$ in the definition of $\hat{L}_{i+1}(z, z_1, \ldots, z_i)$ implies that all strings $[z, z_1, \ldots z_i, x_{i+1}, \ldots, x_k]_k \in \hat{L}_{i+1}(z, z_1, \ldots, z_i)$ satisfy the length requirement, i.e., $|x_j| = p(|z|)$ for $i+1 \leq j \leq k$.

Lemma 3.6 *There is a polynomial-time computable monotonic function $\hat{f}_1$, which for every x enumerates $\hat{L}_1(x)$, i.e., if $w_0, w_1, \ldots, w_j$ are the strings of $\hat{L}_1(x)$ in the lexicographical ordering then*

$$\hat{f}_1(w_i) = \begin{cases} w_{i+1} & \text{for } 0 \leq i < j \\ \text{“}x \in L\text{”} & \text{if } i = j \text{ and } x \in L \\ \text{“}x \notin L\text{”} & \text{if } i = j \text{ and } x \notin L \end{cases}$$

Proof of Lemma 3.6
First we note that for every $0 \leq i \leq k$ every string of the form $[x, x_1, \ldots, x_i, 0^{p(|x|)}, \ldots, 0^{p(|x|)}]_k$ belongs to $\hat{L}_{i+1}(x, x_1, \ldots, x_i)$ (this is easily verified by induction), and thus is the lexicographical minimum of this set.

We now show that for every i such that $0 \leq i \leq k$ there exists a function $\hat{f}_{i+1}$ that enumerates $\hat{L}_{i+1}(x, x_1, \ldots, x_i)$ for every individual $(x, x_1, \ldots, x_i)$ in the above sense. That is, if $v_0, v_1, \ldots, v_r$ are the elements of $\hat{L}_{i+1}(x, x_1, \ldots, x_i)$ in the lexicographical ordering then $\forall n \leq r : \hat{f}^n_{i+1}(v_0) = v_n$. Moreover, if $[x, x_1, \ldots, x_i]_i \in L_i$ then $\hat{f}^{r+1}_{i+1}(v_0) =$ “$[x, x_1, \ldots, x_i]_i \in L_i$” and if $[x, x_1, \ldots, x_i]_i \notin L_i$ then $\hat{f}^{r+1}_{i+1}(v_0) =$ “$[x, x_1, \ldots, x_i]_i \notin L_i$”. Then for $i = 0$ the claim of the Lemma follows.

The proof is by backward induction on i. First let $i = k$ and $w = [x, x_1, \ldots, x_k]_k$; Since $\hat{L}_{k+1} = \Sigma^*$, we have $\hat{L}_{k+1}(x, x_1, \ldots, x_k) = \{w\}$, and we define $\hat{f}_{k+1}(w) =$ “$w \in L_k$” if $R(x, x_1, \ldots, x_k)$ holds and $\hat{f}_{k+1}(w) =$ “$w \notin L_k$” otherwise.

Now fix some $i > 0$ and suppose there is a polynomial function $\hat{f}_{i+1}$, which for every $(x, x_1, \ldots, x_i)$—started at $[x, x_1, \ldots, x_i, 0^{p(|x|)}, \ldots, 0^{p(|x|)}]_k$—enumerates the set $\hat{L}_{i+1}(x, x_1, \ldots, x_i)$ as claimed. We have to distinguish three cases. Let $w = [x, x_1, \ldots, x_k]_k$. We define

$$Q_i = \forall : \quad \hat{f}_i(w) = \begin{cases} \text{“}[x, x_1, \ldots, x_{i-1}]_{i-1} \in L_{i-1}\text{”} & \text{if } x_i = 1^{p(|x|)},\ \hat{f}_{i+1}(w) = \text{“}[x, x_1, \ldots, x_i]_i \in L_i\text{”} \\ [x, x_1, \ldots, x_{i-1}, x_i^+, 0^{p(|x|)}, \ldots, 0^{p(|x|)}]_k & \text{if } x_i \neq 1^{p(|x|)},\ \hat{f}_{i+1}(w) = \text{“}[x, x_1, \ldots, x_i]_i \in L_i\text{”} \\ \text{“}[x, x_1, \ldots, x_{i-1}]_{i-1} \notin L_{i-1}\text{”} & \text{if } \hat{f}_{i+1}(w) = \text{“}[x, x_1, \ldots, x_i]_i \notin L_i\text{”} \\ \hat{f}_{i+1}(w) & \text{otherwise} \end{cases}$$

$$Q_i = \exists: \quad \hat{f}_i(w) =$$
$$\begin{cases} \text{“}[x,x_1,\ldots,x_{i-1}]_{i-1} \in L_{i-1}\text{”} & \text{if } x_i = 1^{p(|x|)},\ \hat{f}_{i+1}(w) = \text{“}[x,x_1,\ldots,x_i]_i \notin L_i\text{”} \\ [x,x_1,\ldots,x_{i-1},x_i^+,0^{p(|x|)},\ldots,0^{p(|x|)}]_k & \text{if } x_i \neq 1^{p(|x|)},\ \hat{f}_{i+1}(w) = \text{“}[x,x_1,\ldots,x_i]_i \notin L_i\text{”} \\ \text{“}[x,x_1,\ldots,x_{i-1}]_{i-1} \notin L_{i-1}\text{”} & \text{if } \hat{f}_{i+1}(w) = \text{“}[x,x_1,\ldots,x_i]_i \in L_i\text{”} \\ \hat{f}_{i+1}(w) & \text{otherwise} \end{cases}$$

For the last case let $x_i = yn$, $|y| = |n|$, and $y_i = y0^{1/2p(|x|)}$. Note that by our assumption on the counting quantifier C and the predicate R the "true" value of the counter $\dot{n}$ will be at most $2^{1/2\cdot p(|x|)}$. Thus for $w = [x,x_1,\ldots,x_k]_k$ where $y <_{lex} 1^{1/2\cdot p(|x|)}$ and $\dot{n} \geq 2^{1/2\cdot p(|x|)}$ we define $\hat{f}_i(w) = w^+$ (the definition of $\hat{f}_i$ on these inputs will not affect the enumerated set). If w is not of this form we compute $\hat{f}_{i+1}(\hat{w})$ with $\hat{w} = [x,x_1,\ldots,x_{i-1},y_i,x_{i+1},\ldots,x_k]_k$. If $\hat{f}_{i+1}(\hat{w})$ does not yield absolute information, i.e., $\hat{f}_{i+1}(\hat{w})$ is not of the form "$[x,x_1,\ldots,x_{i-1},y_i]_i \in L_i$" or "$[x,x_1,\ldots,x_{i-1},y_i]_i \notin L_i$" then let $\hat{f}_{i+1}(\hat{w}) = [x,x_1,\ldots,x_{i-1},y_i,z_{i+1},\ldots,z_k]_k$. We define $\hat{f}_i(w) = [x,x_1,\ldots,x_i,z_{i+1},\ldots,z_k]_k$.

The remaining cases are defined as follows (let $d_i = 0$ if $\hat{f}_{i+1}(\hat{w}) = \text{“}[x,x_1,\ldots,x_{i-1},y_i]_i \notin L_i\text{”}$ and $d_i = 1$ if $\hat{f}_{i+1}(\hat{w}) = \text{“}[x,x_1,\ldots,x_{i-1},y_i]_i \in L_i\text{”}$):

$$Q_i = \mathsf{C}: \quad \hat{f}_i(w) =$$
$$\begin{cases} \text{“}[x,x_1,\ldots,x_{i-1}]_{i-1} \in L_{i-1}\text{”} & \text{if } y = 1^{1/2\cdot p(|x|)},\ \dot{n}+d_i > 2^{1/2\cdot p(|x|)-1} \\ \text{“}[x,x_1,\ldots,x_{i-1}]_{i-1} \notin L_{i-1}\text{”} & \text{if } y = 1^{1/2\cdot p(|x|)},\ \dot{n}+d_i \leq 2^{1/2\cdot p(|x|)-1} \\ [x,x_1,\ldots,x_{i-1},y^+n^+,0^{p(|x|)},\ldots,0^{p(|x|)}]_k & \text{if } y \neq 1^{1/2\cdot p(|x|)},\ d_i = 1 \\ [x,x_1,\ldots,x_{i-1},y^+n,0^{p(|x|)},\ldots,0^{p(|x|)}]_k & \text{if } y \neq 1^{1/2\cdot p(|x|)},\ d_i = 0 \end{cases}$$

To prove that the function $\hat{f}_i$ satisfies the claim we have to check the possible cases. We will consider only the cases $Q_i = \forall$ and $Q_i = \mathsf{C}$. For the case $Q_i = \exists$ the arguments are analogous.

For $Q_i = \forall$ suppose $\hat{L}_i(x,x_1,\ldots,x_{i-1})$ consists of the strings $v_0, v_1, \ldots, v_r$ (lexicographically ordered) and $\hat{f}_i$ has enumerated $\hat{L}_i(x,x_1,\ldots,x_{i-1})$ correctly up to the n-th step with $n \leq r$, i.e., $\hat{f}_i^n(v_0) = v_n = [x,x_1,\ldots x_k]_k$. By definition of $\hat{L}_i(x,x_1,\ldots,x_{i-1})$ this means that $\forall x_i' <_{lex} x_i\, [x,x_1,\ldots,x_{i-1},x_i']_i \in L_i$. If x_i now has reached the lexicographically biggest value it possibly can assume, namely $x_i = 1^{p(|x|)}$, and for this value the function $\hat{f}_{i+1}$ on input v_n returns "$[x,x_1,\ldots,x_i] \in L_i$" then it is clear that for all x_i' of length $p(|x|)$ it holds that $[x,x_1,\ldots,x_{i-1},x_i']_i \in L_i$ and thus by definition $[x,x_1,\ldots,x_{i-1}]_{i-1} \in L_{i-1}$. Hence in the first case the output of $\hat{f}(v_n)$ is correct. If $x_i \neq 1^{p(|x|)}$ and $\hat{f}_{i+1}$ on input v_n yields "$[x,x_1,\ldots,x_i] \in L_i$" then for all $x_i' \leq_{lex} x_i$ it holds that $[x,x_1,\ldots,x_{i-1},x_i']_i \in L_i$, and thus the output in the second case is correct. However, if on input v_n the function $\hat{f}_{i+1}$ returns "$[x,x_1,\ldots,x_{i-1},x_i]_i \notin L_i$" then $[x,x_1,\ldots,x_{i-1},x_i']_i \in L_i$ certainly does not hold that for all x_i' of length $p(|x|)$, and thus $[x,x_1,\ldots,x_{i-1}]_{i-1}$ is not contained in $\hat{L}_{i-1}$. Hence case three is correct. The last case is a direct consequence of the induction hypothesis, and thus for $Q_i = \forall$ the function $\hat{f}_i$ works as claimed.

For $Q_i = \mathsf{C}$ recall that $[x, x_1, \ldots, x_{i-1}]_{i-1}$ is in L_{i-1} if and only if for more than $2^{1/2 \cdot p(|x|)-1}$ strings x_i' of length $p(|x|)$ it holds that $[x, x_1, \ldots, x_{i-1}, x_i']_i \in L_i$. Let $v_0, v_1, \ldots, v_r$ again be the strings in lexicographical order in $\hat{L}_i(x, x_1, \ldots, x_{i-1})$ and assume that $\hat{f}_i$ has enumerated $\hat{L}_i(x, x_1, \ldots, x_{i-1})$ correctly up to $v_n = [x, x_1, \ldots x_k]_k$.

Let $x_i = yn$, $|y| = |n|$, and $y_i = y0^{1/2p(|x|)}$. Then there are exactly $\dot{n}$ strings of the form $y'0^{1/2 \cdot p(|x|)}$ with $|y'| = 0^{1/2 \cdot p(|x|)}$ and $y' <_{lex} y$ such that $[x, x_1, \ldots, x_{i-1}, y'0^{1/2 \cdot p(|x|)}]_i \in L_i$. By our standardization of the C-quantifier any string z, for which $[x, x_1, \ldots, x_{i-1}, z]_i \in L_i$ holds, must be of the form $z = z'0^{1/2 \cdot p(|x|)}$ for some z'. Thus it also holds that there are exactly $\dot{n}$ strings $z <_{lex} y0^{1/2 \cdot p(|x|)}$ such that $[x, x_1, \ldots, x_{i-1}, z]_i \in L_i$. As for $Q_i = \forall$ the case that $\hat{f}_{i+1}(v_n)$ does not return absolute information is again covered by the induction hypothesis, i.e., $\hat{f}_i(v_n)$ is correct. Consider the case that y has reached its maximal possible value $1^{1/2 \cdot p(|x|)}$. Let $\hat{v}_n$ be the substitution of v_n as above. If $\hat{f}_{i+1}(\hat{v}_n)$ returns "$[x, x_1, \ldots, x_{i-1}, y_i]_i \in L_i$" then $[x, x_1, \ldots, x_{i-1}]_{i-1} \in L_{i-1}$ if and only if the counter $\dot{n}$ incremented by one exceeds $2^{1/2 \cdot p(|x|)-1}$ (case one). If $\hat{f}_{i+1}(\hat{v}_n)$ returns "$[x, x_1, \ldots, x_{i-1}, y_i]_i \notin L_i$" then $[x, x_1, \ldots, x_{i-1}]_{i-1} \in L_{i-1}$ if and only if the counter $\dot{n}$ exceeds $2^{1/2 \cdot p(|x|)-1}$ (case two). Finally, if y has not reached its maximal value ($y \neq 1^{1/2 \cdot p(|x|)}$) then the counter is increased—depending on whether $\hat{f}_{i+1}(\hat{v}_n)$ indicates that "$[x, x_1, \ldots, x_{i-1}, y_i]_i \in L_i$" or "$[x, x_1, \ldots, x_{i-1}, y_i]_i \notin L_i$" (cases three and four). In any case it holds that $\hat{f}_i(v_n) = v_{n+1}$.

End of the proof of Lemma 3.6 □

From the enumeration function $\hat{f}_1$ for the sets $\hat{L}_1(x)$ as constructed in Lemma 3.6 we can build the enumeration function for $\hat{L}$ very easily. Recall that $\hat{L}$ is the disjoint union of $\hat{L}_1$ and $\hat{L}_R$. We define the enumeration function h for $\hat{L}$ as follows. Again, let $w = [x, x_1, \ldots, x_k]$.

$$h(w \cdot 0) = \begin{cases} [x, 1^{p(|x|)}, \ldots, 1^{p(|x|)}]_k \cdot 1 & \text{if } \hat{f}_1(w) = \text{"}x \in L\text{"} \\ [x^+, 0^{p(|x^+|)}, \ldots, 0^{p(|x^+|)}]_k \cdot 0 & \text{if } \hat{f}_1(w) = \text{"}x \notin L\text{"} \\ \hat{f}_1(w) \cdot 0 & \text{otherwise} \end{cases}$$

and

$$h(w \cdot 1) = [x^+, 0^{p(|x^+|)}, \ldots, 0^{p(|x^+|)}]_k \cdot 0$$

This function enumerates $\hat{L}$ in lexicographical order. Clearly its outputs are lexicographically bigger than its inputs. Started at the lexicographical minimum of $\hat{L}$ it produces all its elements as an easy induction shows.

2. Let K be a class in the counting hierarchy. It holds that if $\exists K$ is closed under complement then $\exists K = \forall coK$, and since coK is a class of the counting hierarchy as well it follows from part 1 of the Theorem that in this case $\exists K$ has monotonically enumerable complete sets.

For the reverse direction note that for any language $L \in K$ that has an enumeration function h it holds that $\overline{L} \in \exists K$. This is direct consequence of the equivalence

$$x \in \overline{L} \iff \exists y <_{lex} x : \; y \in L \text{ and } x <_{lex} h(y)$$

Hence it holds that if there exists a monotonically enumerable complete set L for $\exists K$ then $\forall coK \subseteq \exists\exists K = \exists K$, which in turn implies $\exists K \subseteq \forall coK$. Thus $\exists K$ must be closed under complement.

End of the proof of Theorem 3.1 □

4 Complete Characterizations

In this section we give two complete characterization of complexity classes in terms of monotonic enumerations.

Theorem 4.1 *For any language L the following statements are equivalent:*

1. $L \in PSPACE$
2. *There is a set $\hat{L}$ such that $L \leq_m^p \hat{L}$ and $\hat{L}$ is monotonically enumerable.*
3. *There is a set $\hat{L}$ such that $L \leq_m^p \hat{L}$ and $\hat{L}$ is monotonically enumerable via a one-one enumeration function f which is polynomial-time invertible on range(f).*

To prove this Theorem we need a result of Bennett on reversible Turing machines ([2]), which was used in a similar fashion in [4]. A Turing machine is *reversible* if the function that maps its IDs (instantaneous descriptions, see [15]) to its succesing IDs is one-to-one.

Proposition 4.2 *([2]) If a function f is computable by a Turing machine on space S then f is computable by a reversible Turing machine on space $O(S^2)$.*

Proof of Theorem 4.1 It suffices to show that there exists a PSPACE-complete set $\hat{L}$ that is monotonically enumerable via a one-one poynomial-time invertible enumeration function, thus proving "1 ⇒ 3". "3 ⇒ 2" is immediate, and "2 ⇒ 1" follows from the fact that every monotonically enumerable set is in PSPACE and PSPACE is closed under polynomial-time reductions.

Let us first give an example of a set that is PSPACE-complete and monotonically enumerable ([14]). Let L be PSPACE-complete and M' be a polynomial-space bounded Turing machine that accepts L. Without loss of generality we assume that M' runs exactly $2^{r(|x|)}$ steps on every input x for some polynomial r. We choose a polynomial q and an encoding of the set of IDs of M' into Σ^* such that all IDs of M' on input x are represented by some string of exact length $q(|x|)$, and $q(n) > r(n)$ for all n. Without loss of generality we can assume that M' has some standard accepting ID, some standard rejecting ID, and some standard initial ID. For an input

x we denote the corresponding strings of length $q(|x|)$ by $C_A(x), C_R(x)$, and $C_0(x)$, respectively. Then for

$$L' = \{[x,n,C]_3 : |n| = |C| = q(|x|) \text{ and after } \dot{n} \text{ steps } M' \text{ on } x \text{ reaches the ID encoded by } C\}$$

it holds that $x \in L \iff [x,w,C_A(x)]_3 \in L'$ where w is the binary encoding of $2^{r(|x|)}$ by $q(|x|)$ bits. Thus L' is PSPACE-complete, and it is not hard to see that L' is monotonically enumerable, e.g., "map all inputs $[x,n,C]_3$ that satisfy $C \neq C_A(x), C_R(x)$ has a successing ID C' and $\dot{n} < 2^{r(|x|)}$ to $[x,n^+,C']_3$, and map all other inputs to $[x^+,0^{q(|x^+|)},C_0(x^+)]_3$", describes an enumeration function for L'. However, this enumeration function is not one-one. Even though by Proposition 4.2 there is a polynomial-space bounded reversible Turing machine M that accepts L we have to overcome the following difficulties: for reversible machines it is not clear how the assumptions on having an easily computable running time ($2^{r(|x|)}$ in the example above) and having an easily accessible accepting ID ($C_A(x)$) can be assured simultaneously. Note that both assumptions are necessary in the above construction to permit an easy reduction from L to L'. Also, we have to make sure that no string of the form $[x,0^{q(|x|)},C_0(x)]_3$ is accessed from more than one string by the enumeration function. To achieve our goal we will have to repeat and rewind the reversible computation of M on an input x in some careful way.

Let L be as above, M be a reversible Turing machine that accepts L, and r be a polynomial such that each M runs less than $2^{r(|x|)}$ steps if it accepts the input x. We choose a polynomial p such that $2^{p(n)} > 2^{12r(n)}$ for all $n \in \mathbb{N}$, and all configurations of M on input x that do not exceed size $r(|x|)$ can be encoded in strings of length $p(|x|)$ (as above). Without loss of generality we can assume that M has a standard initial ID, which has no predecessing ID; if $x \in L$, the computation of M ends in some standard accepting ID, which has no successing ID; and M does not terminate if $x \notin L$ ([2]). The strings of length $p(|x|)$, that represent the initial and the accepting ID will be denoted be $C_0(x)$ and $C_A(x)$. We assume that for each n the string $0^{p(n)}$ does not represent an ID of M. In the following we will identify the IDs with its encodings and assume that for each x the IDs of M have always the exact length $p(|x|)$.

$\hat{L}$ will be the language that is monotonically enumerable by a one-one function. We define $\hat{L}$ by explicitly giving the enumeration function h. As in the example above $\hat{L}$ will consist of tupels of the form $[x,n,C]_3$ with $|n| = |C| = p(|x|)$ where C is some ID of M on input x. For each x the string $v_1 = [x,0^{p(|x|)},C_0(x)]_3$ will belong to $\hat{L}$. In the following we will fix x and split the set of strings of the form $[x,n,C]_3$ for some n and C into intervals. Let $K = 2^{r(|x|)}$ and $I_j = \{[x,n,C]_3 : 2K(j-1) \leq \dot{n} < 2Kj\}$ for $j \in \{1,\ldots,6\}$. Strings that are not in any of these intervals will not belong to $\hat{L}$.

The intervals I_1, I_2, I_3 will be used to produce some easily accessible string that represents the outcome of the computation of M on x: for m_4 with $\dot{m}_4 = 6K$ the chain $V = v_1, h(v_1), \ldots$ will reach a string $t = [x,m_4,C]_3 \in I_4$ such that $C = C_A(x)$ holds if $x \in L$ and $C \neq C_A(x)$ if

$x \notin L$. In the latter case C will be the ID that is reached by M after K steps. Hence

$$g(x) = [x, m_4, C_A(x)]_3$$

provides a polynomial-time computable reduction from L to $\hat{L}$—thus ensuring the completeness of $\hat{L}$. In the intervals I_4, I_5, I_6 this process will be reversed, that is, from the string t the initial ID is reconstructed, and the chain V reaches the string $[x, m_7, C_0(x)]_3$, $\dot{m}_7 = 12K$—independent of whether $x \in L$ holds or not. Thus, by defining $h([x, m_7, C_0(x)]_3) = [x^+, 0^{p(|x^+|)}, C_0(x^+)]_3$ we can "glue the different x-pieces together" in a one-one fashion, and the same will be repeated for the lexicographical successor of x.

For each configuration C by C_s we denote its successing configuration (if it exists) and by C_p its predecessing configuration (if it exists).

In I_1 the computation of $M(x)$ is simulated: h maps each $[x, n, C]_3$ with $\dot{n} < K$ to $[x, n^+, C_s]_3$ until $C = C_A(x)$. Inputs $[x, n, C]_3$ for which $\dot{n} = K$ holds or for which $C = C_A(x)$ and $\dot{n} \leq K$ holds are mapped to $[x, m_2, C]_3$ with $\dot{m}_2 = 4K - 2\dot{n}$, i.e., into I_2. Inputs that are not mappable by the above rules (e.g., C does not represent an ID with a successing one, or $\dot{n} > K$) are not in $\hat{L}$ and will be mapped in a padding fashion:

$$h([x, n, C]_3) = [[x, n, C]_3, 0 \ldots 0, 0 \ldots 0]_3.$$

Note that at this point the chain V reaches some string $v_2 = [x, m_2, C]_3$ in I_2, and it holds $C = C_A(x)$ if $x \in L$ and $\dot{m}_2 = 2K$ and $C \neq C_A(x)$ if $x \notin L$.

In I_2 we will rewind the computation of $M(x)$ and increase the counter: $h([x, n, C]_3) = [x, n^+, C_p]_3$ until $C = C_0(x)$. $[x, n, C_0(x)]_3$ is mapped to $[x, m_3, C_0(x)]_3$ with $\dot{m}_3 = \dot{n} + 2K$, that is into I_3. Strings in I_2 that can not be mapped by this rules will be padded as above. This way the chain V arrives at the string $v_3 = [x, m_3, C_0(x)]_3$, which still carries the information on the length of the computation of $M(x)$: it holds that $6K - N = \dot{m}_3$, where N is the length of the computation if $x \in L$ and $N = K$ if $x \notin L$.

In I_3 we repeat the computation and increase the counter: $h([x, n, C]_3)$ is defined to be $[x, n^+, C_s]_3$. Again, inputs that are not mappable by this rule are padded. From the construction it follows that the string $v_4 \in I_4$ that is reached at this point by the chain V is $[x, m_4, C_A(x)]_3$ if $x \in L$ and $[x, m_4, C]_3$, $C \neq C_A(x)$ otherwise.

In I_4, I_5, and I_6 we reverse this process by changing the directions of the computations—thus deriving the initial configuration again. Since the algorithm works similar to the intervals $I_1 - I_3$, for this intervals we give a more compact description of h. As before, all strings in the intervals I_4, I_5, and I_6 that are not mappable by the following rules will be padded.

I_4: For $\dot{n} \leq 7K$ define $h([x, n, C]_3) = [x, n^+, C_p]_3$ if $C \neq C_0(x)$, and $h([x, n, C]_3) = [x, m_5, C]_3$ with $\dot{m}_5 = 10K - 2(\dot{n} - 6K)$ if $C = C_0(x)$.

I_5: Define $h([x,n,C]_3) = [x,n^+,C_s]_3$ if $C \neq C_A(x)$. If $C = C_A(x)$ define $h([x,n,C_A(x)]_3) = [x,m_6,C]_3$ with $\dot{m}_6 = \dot{n} + 2K$.

I_6: Define $h([x,n,C]_3) = [x,n^+,C_p]_3$.

Finally, all strings apart from $[x,m_7,C_0(x)]_3$ that do not belong to one of the intervals $I_1,\ldots,I_6$ will be padded.

h is polynomial-time computable, and, looking at the single cases, it is not difficult to check that for each string $[x,n,C]_3$ either h^{-1} is undefined or there is exactly one string that is mapped to $[x,n,C]_3$ by h. As an example, consider a string $w = [x,n,C]_3 \in I_5$. In the two following cases $h^{-1}(w)$ is not defined: $C \neq C_0(x)$ and C_p does not exist; $C = C_0(x)$ and $\dot{n}$ is odd. If $C = C_0(x)$ and $\dot{n}$ is even then $h^{-1}(w)$ is the unique string $[x,m,C_0] \in I_4$ that satisfies $\dot{n} = 10K - 2(\dot{m} - 6K)$.

Note that the use of the padding function does not violate the one-oneness of the h, since strings $0^{p(n)}$ do not represent valid IDs of M, and thus have no successing of predecessing IDs. Note also that for each $[x,n,C]_3$ it can be easily determined, in which interval it is, and thus which of the above rules has to be applied. Since the padding function is polynomial-time invertible as well, h is polynomial-time invertible. □

Before giving the characterization of the class $\oplus$OptP let's have a look at monotonic *bi*-enumerations. If a set L is monotonically bi-enumerable via an enumeration function h then there exist strings v and w such that $L = \{v,h(v),\ldots\}$ and $\overline{L} = \{w,h(w),\ldots\}$, that is, Σ^* is partitioned into two chains. In this case h has the following two properties:

(A) For all strings $x \geq_{lex} \max\{v,w\}$: $\exists y\,[y <_{lex} x <_{lex} h(y)]$

(B) For all strings x: $\forall y_1, y_2\,[y_1 <_{lex} x <_{lex} h(y_1), y_2 <_{lex} x <_{lex} h(y_2) \Rightarrow y_1 = y_2]$,

which intuitively say that all but finite number of strings are passed by exactly one chain. In fact, properties (A) and (B) imply that for all $x \in \{z : z >_{lex} \max\{v,w\}\}$ it holds that $x \in range(h)$, i.e., $h^{-1}(x)$ is defined, and h is one-to-one on this set (see the proof of Theorem 4.4). Clearly, such a function can easily be modified to be a monotonic bi-enumeration function. Thus having an enumeration function with properties (A) and (B) is a necessary and sufficient condition for a set to be monotonically bi-enumerable.

Now, relaxing requirement (A) we come to the following definition:

Definition 4.3 *A set L is (polynomially)* piecewise monotonically bi-enumerable by iteration *(short:* piecewise monotonically bi-enumerable*) if L is finite or cofinite or there exists a polynomial-time computable monotonic function f, which enumerates L, and for all x it holds*

$$\forall y_1, y_2\,[y_1 <_{lex} x <_{lex} f(y_1), y_2 <_{lex} x <_{lex} f(y_2) \Rightarrow y_1 = y_2].$$

The justification for the term "piecewise" will be given by the proof of Theorem 4.4, which shows that such an enumeration function partitions Σ^* into intervals $[x_0, x_1], [x_1, x_2], [x_2, x_3], \ldots$, and the enumeration function bi-enumerates the set within these intervals and fails to do so at the edges.

Note that these enumeration functions do not have to be one-one any more. Clearly, any monotonically bi-enumerable set is piecewise monotonically bi-enumerable.

Theorem 4.4 *For any set L the following are equivalent:*

1. $L \in \oplus\mathrm{OptP}$
2. *L is reducible to a piecewise monotonically bi-enumerable set.*

Proof The proof involves some ideas of [11]. Let L and $\overline{L}$ be infinte.

"2⇒1": Let L be piecewise monotonically bi-enumerable via an enumeration function h having property (B). We show that there exists a function $f \in \mathrm{FP}^{\mathrm{NP}}$ and a nondeterministic polynomial-time Turing machine N such that $x \in L$ if and only if N accepts $f(x)$ on an odd number of computation paths. Then by Proposition 2.5 it follows that $L \in \oplus\mathrm{OptP}$.

Let $v_0 = \min L$ and $w_0 = \min \overline{L}$. Without loss of generality we assume that $v_0 <_{lex} w_0$. Fix a string x with $w_0 <_{lex} x$. If we know a string $y \in \overline{L}$ such that $w_0 \leq_{lex} y \leq_{lex} x$ and h bi-enumerates L on the interval $I_{yx} = \{z : y <_{lex} z \leq_{lex} x\}$, that is, $\{h^i(y) | i \geq 0\} \cap I_{yx} = \overline{L} \cap I_{yx}$, then we can use the $\oplus$P-algorithm of [13, Theorem 6.5] to decide membership in L: on input $y\#x$, $y <_{lex} x$ the nondeterministic polynomial-time Turing machine N guesses a string $z : y \leq_{lex} z <_{lex} x$ and accepts if and only if $h(z) \neq z^+$. Since $y \in \overline{L}$, it holds that $x \in L$ if and only if for an odd number of strings z, $y \leq_{lex} z <_{lex} x$, it holds that $z \in L \iff z^+ \notin L$ if and only if N accepts on an odd number of computation paths.

We will now use the $\mathrm{FP}^{\mathrm{NP}}$-precomputation to find such a string y for a given x. In some cases the algorithm computing f as given below will detect that $x \in L$ holds. Then it will output some fixed string that is accepted by N on an odd number of paths. The function f on inputs x with $y_0 <_{lex} x$ is specified by the following algorithm:

- Let $A(x) = \{z | \, w_0 <_{lex} z \leq_{lex} x, \exists y_1, y_2 : y_1 \neq y_2, h(y_1) = h(y_2) = z\}$.
- If $A(x) = \emptyset$ then $f(x) = w_0\#x$.
- If $A(x) \neq \emptyset$ then let $v_m = \max A(x)$.
  - (a) If $(\forall y : v_m \leq_{lex} y <_{lex} x)\ [h(y) = y^+]$ then "$x \in L$"
  - (b) else let $w_m = \min I_{v_m x} \cap \{z | f(z^-) \neq z\}$ and output $f(x) = w_m\#x$

It is easily verified that f can be computed in polynomial-time relative to an NP-oracle (v_m and w_m are determined using binary search).

To prove the correctness of the algorithm first assume that $A(x)$ is empty. In this case h bi-enumerates L on I_{w_0x}: if there is some $z \in \overline{L} \cap I_{w_0x}$ for which $z \notin \{h^i(w_0)|i \geq 0\}$ then there exist two distinct strings $v_1 \in L$ and $w_1 \in \{h^i(w_0)|i \geq 0\}$ such that $v_1 <_{lex} z <_{lex} h(v_1)$ and $w_1 <_{lex} z <_{lex} h(w_1)$, since L and $\overline{L}$ are infinte and h is lexicographically increasing. This contradicts (B).

For the case that $A(x)$ is nonemtpy Lemma 4.5 shows that all elements of $A(x)$ belong to L. It follows that if for all y with $v_m \leq_{lex} y <_{lex} x$ it holds that $h(y) = y^+$ then $x \in L$, since h is an enumeration function for L. In the other case w_m is defined. By Lemma 4.5 it belongs to $\overline{L}$, and h bi-enumerates L on the interval I_{w_mx}. Thus the output $f(x) = w_m\#x$ is correct.

Lemma 4.5 *Let $w_0 <_{lex} x$, $A(x) \neq \emptyset$, and w_m be defined.*

1. $A(x) \subset L$
2. $w_m \in \overline{L}$ *and* $\{h^i(w_m)|i \geq 0\} \cap I_{w_mx} = \overline{L} \cap I_{w_mx}$

Proof of Lemma 4.5

1. Suppose there is some $z \in A(x)$ and $z \notin L$. Let $y_1 <_{lex} y_2 <_{lex} z$ such that $h(y_1) = h(y_2) = z$. Since L is infinte and h enumerates L there is some $w \in L$ such that $w <_{lex} z <_{lex} h(w)$. Then $y_2 <_{lex} w <_{lex} z$ cannot hold, since property (B) would not hold for w. Also $y_1 <_{lex} w <_{lex} y_2$ is false, since property (B) would not hold for y_2. For the same reason $w <_{lex} y_1$ cannot hold. Hence we have a contradiction, which proves $z \in L$.

2. Let $v_m = \max A(x) <_{lex} x$ and $y_1 <_{lex} y_2 <_{lex} v_m$ such that $h(y_1) = h(y_2) = v_m$. First note that $w_m \notin range(h)$, i.e., $h^{-1}(w_m)$ is not defined. Otherwise by the minimality of w_m it would follow that $h^{-1}(w_m) <_{lex} v_m$ and a contradiction to the assumption that h has property (B) can be derived in the same way as in 1. Hence $w_m \notin L$. Also there cannot exist a string $z \in \overline{L} \cap I_{w_mx}$, such that $h^{-1}(z)$ is not defined, since it would contradict (B) in the same way as for $A(x) = \emptyset$.

□

End of the proof of Lemma 4.5

"2⇒1": Let $L \in \oplus$OptP. We will construct a language $\hat{L}$ such that L is reducible to $\hat{L}$ and $\hat{L}$ is piecewise bi-enumerable. By Proposition 2.5, part 2 we know that there is a nondeterministic polynomial-time bounded Turing machine N, a predicate $R(x,y) \in$ P, and a polynomial p such that for all x and $z = \max\{y : R(x,y), |y| = p(|x|)\}$

$$x \in L \iff N \text{ accepts } [x,z]_2 \text{ on an odd number of computation paths.}$$

Without loss of generality we can assume that $R(x, 0^{p(|x|)})$ holds and that N has the following features: on input $[x,y]_2$ all computation paths w have exactly length $p(|x|)$; for all w such that

$|w| = p(|x|) - 1$ the machine N accepts on the path $w0$; and for all w such that $|w| = p(|x|) - 1$ the machine N does not accept on the path $w1$ if $R(x,y)$ does not hold. Then in particular it holds that for all y of length $p(|x|)$ such that $y >_{lex} \max\{v : R(x,v), |v| = p(|x|)\}$ the machine N accepts $[x,y]_2$ on an even number of paths.

We now define the language $\hat{L}$ by the equivalence

$$\begin{aligned} [x,y,w]_3 \in \hat{L} \iff & \text{there is an odd number of strings } [x,y',w']_3 \leq_{lex} [x,y,w]_3 \\ & \text{such that } N \text{accepts } [x,y']_2 \text{ on path } w' \text{ and} \\ & y' \geq_{lex} \max\{v \leq_{lex} y \mid R(x,v),\ |v| = p(|x|)\} \end{aligned}$$

The idea behind this definition is the following: since N accepts all inputs $[x,y]_2$ such that $y >_{lex} \max\{v : R(x,v), |v| = p(|x|)\}$ on an even number of paths, the string $[x,z,1^{p(|x|)}]_3$ with $z = \max\{v : R(x,v), |v| = p(|x|)\}$ is in $\hat{L}$ if and only if $[x,1^{p(|x|)},1^{p(|x|)}]_3 \in \hat{L}$. More explicitly, it holds

$$\begin{aligned} x \in L \iff & N \text{ accepts } [x,z]_2 \text{ on an odd number of paths} \\ \iff & \text{there is an odd number of strings } [x,y',w']_3 \text{ such that } N \text{ accepts } [x,y']_2 \\ & \text{on path } w' \text{ and } [x,z,0^{p(|x|)}]_3 \leq_{lex} [x,y',w']_3 \leq_{lex} [x,1^{p(|x|)},1^{p(|x|)}]_3 \\ \iff & [x,1^{p(|x|)},1^{p(|x|)}]_3 \in \hat{L} \end{aligned}$$

Thus $g(x) = [x,1^{p(|x|)},1^{p(|x|)}]_3$ reduces L to $\hat{L}$.

It remains to show that $\hat{L}$ is indeed piecewise monotonically bi-enumerable. The enumeration function h we define will bi-enumerate $\hat{L}$ on the intervals I_{ab} where $a = [x,y,0^{p(|x|)-1}1]_3$ (or $a = [x,y,0^{p(|x|)-2}10]_3$, depending on whether N accepts $[x,y]_2$ on the path $0^{p(|x|)}1$), $b = [x,y',0^{p(|x|)}]_3$ such that $R(x,y)$ and $R(x,y')$ holds, but there is no value y'' between y and y' such that $R(x,y'')$ holds. Strings of the form $a = [x,y,0^{p(|x|)}]_3$, for which $R(x,y)$ holds will be the points, for which h^{-1} is not uniquely defined.

More concretely, for all inputs $[x,y,w]_3$, such that there is a string $[x,y',w']_3 >_{lex} [x,y,w]_3$, for which the value $\max\{v \leq_{lex} y \mid R(x,v), |v| = p(|x|)\}$ has not changed (i.e., $\max\{v \leq_{lex} y \mid R(x,v), |v| = p(|x|)\} = \max\{v \leq_{lex} y' \mid R(x,v), |v| = p(|x|)\}$), we define $h([x,y,w]_3) = [x,y',w']_3$, which is the lexicographical smallest string $>_{lex} [x,y,w]_3$ such that $[x,y,w]_3 \in \hat{L}$ if and only if $[x,y',w']_3 \in \hat{L}$. Since N accepts on each path of the form $v0$, i.e., on each second path in the lexicographical ordering, this can be done in polynomial-time.

If we cannot find such a string $[x,y',w']_3$ before the value of $\max\{v \leq_{lex} y \mid R(x,v), |v| = p(|x|)\}$ changes then from the definition of $\hat{L}$ it follows that $w \in \{1^{p(|x|)-1}0, 1^{p(|x|)}\}$. In this case we define $h([x,y,w]_3) = [x,y^+,0^{p(|x|)}]_3$ if $y \neq 1^{p(|x|)}$ and $h([x,y,w]_3) = [x^+,0^{p(|x^+|)},0^{p(|x^+|)}]_3$ otherwise. Since for all y, for which $R(x,y)$ holds, $[x,y,0^{p(|x|)}]_3$ belongs to $\hat{L}$, in this case h

maps all elements into $\hat{L}$. Hence h enumerates $\hat{L}$, and it is not hard to see that this function has property (B) as required.

End of the proof of Theorem 4.4 □

It follows that the class of nearly near-testable sets (for a definition see [11]) essentially coincides with the class of piecewise monotonically bi-enumerable sets.

Corollary 4.6 *For any language L the following are equivalent:*

1. *L is reducible to some nearly near-testable set.*
2. *L is reducible to some piecewise monotonically bi-enumerable set.*

5 Conclusion and Open Problems

This paper has studied monotonic enumerability. It has been shown that various complexity classes can be completely characterized in terms of monotonic enumeration functions, and that many complexity classes between P and PSPACE in fact do have monotonically enumerable complete sets. For quantifier-defined complexity classes monotonic enumerability reveals surprising differences between classes that are defined by a leading existential quantifier on the one side and by a leading universal or counting quantifier on the other side.

Many problems remain open. What we would like to know is, which classes have complete sets with one-one monotonic enumerations functions. We have seen that PSPACE is such a class, but the enumeration functions constructed for the complete sets in Theorem 3.1 are certainly not one-one. It seems even hard to construct a complete set for coNP that is one-one monotonically enumerable. Can we get any evidence that this is impossible? Note that in [14] it was shown that there exist GP-complete sets that are monotonically enumerable via one-one polynomial-time invertible enumeration functions. On the other hand we would like to know more about the relations to other kinds of monotonic enumerability. Which classes have enumerable complete sets via enumeration functions that are not length-decreasing? It is also interesting to consider honest enumerations, which means the enumeration chain is only allowed to reach strings that are at most polynomially shorter than the greatest string reached so far.

Acknowledgments

I am grateful to Paul Young for helpful discussions, reading an earlier version of this paper, and suggesting many improvements.

References

[1] J. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I.* Springer Verlag, 1988.

[2] C. Bennett. Time/space trade-offs for reversible Turing machines. *SIAM Journal on Computing*, 18(4):766–776, August 1989.

[3] L. Berman and J. Hartmanis. On isomorphisms and density of NP and other complete sets. *SIAM Journal on Computing*, 6(2):305–322, 1977.

[4] S. Fenner, S. Kurtz, and J. Royer. Every polynomial every 1-degree collapses iff P=PSPACE. In *Proceedings 30th IEEE Symposium on Foundations of Computer Science*, pages 624–629. IEEE Computer Society Press, 1989.

[5] J. Gill. Computational complexity of probabilistic Turing machines. *SIAM Journal on Computing*, 6(4):675–695, December 1977.

[6] J. Goldsmith, L. Hemachandra, and K. Kunen. On the structure and complexity of infinite sets with minimal perfect hash functions. Technical Report TR-339, University of Rochester, Department of Computer Science, Rochester, NY, 14627, USA, May 1990.

[7] J. Goldsmith, L. Hemachdra, D. Joseph, and P. Young. Near-testable sets. *SIAM Journal on Computing.* To appear.

[8] J. Goldsmith, D. Joseph, and P. Young. Self-reducible, P-selective, near-testable, and P-cheatable sets: The effect of internal structure on the complexity of a set. In *Proceedings 2nd Structure in Complexity Theory Conference*, pages 50–59. IEEE Computer Society Press, 1987.

[9] T. Gundermann, N.A. Nasser, and G. Wechsung. A survey on counting classes. In *Proceedings 5th Structure in Complexity Theory Conference*, pages 140–153. IEEE Computer Society Press, 1990.

[10] L. Hemachandra. Algorithms from complexity theory: Polynomial-time operations for complex sets. In *Proceedings SIGAL International Symposium on Algorithms*, pages 221–231. Springer-Verlag Lecture Notes in Computer Science #450, 1990.

[11] L. Hemachandra and A. Hoene. On sets with efficient implicit membership tests. *SIAM Journal on Computing.* To appear.

[12] L. Hemachandra, A. Hoene, and D. Siefkes. Polynomial-time functions generate SAT: On P-splinters. In *Mathematical Foundations of Computer Science*, pages 259–269. Springer-Verlag Lecture Notes in Computer Science #379, 1989.

[13] L. Hemachandra, A. Hoene, D. Siefkes, and P. Young. On sets polynomially enumerable by iteration. *Theoretical Computer Science*. To appear.

[14] A. Hoene. *Polynomielle Splinter*. PhD thesis, Technische Universität Berlin, 1990. In German.

[15] J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.

[16] M. Krentel. The complexity of optimization problems. *Journal of Computer and System Sciences*, 36:490–509, 1988.

[17] C. Papadimitriou and S. Zachos. Two remarks on the power of counting. In *Proceedings 6th GI Conference on Theoretical Computer Science*, pages 269–276. Springer-Verlag Lecture Notes in Computer Science #145, 1983.

[18] J. Simon. On the difference between one and many. In *Proceedings 14th International Colloquium on Automata, Languages, and Programming*, pages 480–491. Springer-Verlag Lecture Notes in Computer Science #52, 1977.

[19] L. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3:1–22, 1977.

[20] J. Torán. An oracle characterization of the counting hierarchy. In *Proceedings 3th Structure in Complexity Theory Conference*, pages 213–223. IEEE Computer Society Press, 1988.

[21] K. Wagner. The complexity of combinatorial problems with succinct input representation. *Acta Informatica*, 23:325–356, 1986.