Information and Computation **158**, 29–52 (2000)
doi:10.1006/inco.1999.2810, available online at http://www.idealibrary.com on IDEAL®

Tally NP Sets and Easy Census Functions

Judy Goldsmith[1]

Department of Computer Science, *University of Kentucky*, *Lexington*, *Kentucky 40506*
E-mail: goldsmit@cs.uky.edu

Mitsunori Ogihara[2]

Department of Computer Science, *University of Rochester*, *Rochester*, *New York 14627*
E-mail: ogihara@cs.rochester.edu

and

Jörg Rothe[3]

Institut für Informatik, *Friedrich-Schiller-Universität Jena*, *07740 Jena*, *Germany*
E-mail: rothe@informatik.uni-jena.de

We study the question of whether every P set has an easy (i.e., polynomial-time computable) census function. We characterize this question in terms of unlikely collapses of language and function classes such as $\#\mathrm{P}_1 \subseteq \mathrm{FP}$, where $\#\mathrm{P}_1$ is the class of functions that count the witnesses for tally NP sets. We prove that every $\#\mathrm{P}_1{}^{\mathrm{PH}}$ function can be computed in $\mathrm{FP}^{\#\mathrm{P}_1{}^{\#\mathrm{P}_1}}$. Consequently, every P set has an easy census function if and only if every set in the polynomial hierarchy does. We show that the assumption $\#\mathrm{P}_1 \subseteq \mathrm{FP}$ implies $\mathrm{P} = \mathrm{BPP}$ and $\mathrm{PH} \subseteq \mathrm{MOD}_k\mathrm{P}$ for each $k \geqslant 2$. We also relate a set's property of having an easy census function to other well-studied properties of sets, such as rankability and scalability (the closure of the rankable sets under P-isomorphisms). Finally, we prove that it is no more likely that the census function of any set in P can be approximated (more precisely, can be n^{α}-enumerated in time n^{β} for fixed α and β) than that it can be precisely computed in polynomial time.

[1] Supported in part by NSF Grant CCR-9610348.
[2] Supported in part by NSF CAREER Award CCR-9701911.
[3] Supported in part by Grants NSF-INT-9513368/DAAD-315-PRO-fo-ab, NSF-CCR-9322513, and NSF-INT-9815095/DAAD-315-PPP-gü-ab, and by a NATO Postdoctoral Science Fellowship from the Deutscher Akademischer Austauschdienst ("Gemeinsames Hochschulsonderprogramm III von Bund und Ländern"). Work done in part while visiting the University of Rochester and the University of Kentucky.

0890-5401/00 $35.00

1. INTRODUCTION

Does every P set have an easy (i.e., polynomial-time computable) census function? Many important properties similar to this one were studied during the past decades to gain insight into the nature of feasible computation. Among the questions that were previously studied are the question of whether or not every P set has an easy-to-compute ranking function [GS91, HR90], whether every P set is P-isomorphic to some rankable set [GH96], whether every sparse set in P is P-printable [HY84, AR88, RRW94], whether every infinite set in P has an infinite P-printable subset [AR88, HRW97a], whether every P-printable set is P-isomorphic to some tally set in P [AR88], and whether every P set admits easy certificate schemes [HRW97a, HRW97b], to name just a few. Some of those questions arise in the field of data compression and are related to Kolmogorov complexity, some are linked to the question of whether one-way functions exist.

Extending this line of research, the present paper studies the complexity of computing the census functions of sets in P. Census functions have proven to be a particularly important and useful notion in complexity theory, and their use has had a profound impact upon almost every area of the field. In particular, consider the extensive literature related to the isomorphism conjecture of Berman and Hartmanis (e.g., [BH77, Mah82], and many other papers), the work on the existence of Turing-hard sparse sets (or of polynomial-size circuits) for various complexity classes (e.g., [KL80, KS85, BBS86, HR97]), the results relating the computation times for NP sets to their densities and the results on P-printability [HY84, AR88, RRW94, GH96], the upward separation technique (e.g., [Har83, HIS85, All91, RRW94, HJ95]; see [HHH99] for more recent advances that are not based on census functions), the results on positive relativization and relativization to sparse oracles (e.g., [Lon85, LS86, BBS86]), the unexpected collapse of the strong exponential-time hierarchy [Hem89], and applications to extended lowness [HJRW98].

Valiant, in his seminal papers [Val79a, Val79b], introduced $\#\mathrm{P}$, the class of functions that count the solutions of NP problems, and its tally version $\#\mathrm{P}_1$ for which the inputs are given in unary. Although $\#\mathrm{P}_1$ has not become as prominent as $\#\mathrm{P}$, it contains a number of quite interesting and important problems such as the problem `Self-Avoiding Walk` (see [Wel93]): Given an integer n in unary, compute the number of self-avoiding walks on the square lattice having length n and rooted at the origin. `Self-Avoiding Walk` is a well-known classical problem of statistical physics and polymer chemistry, and it is an intriguing open question whether `Self-Avoiding Walk` is $\#\mathrm{P}_1$-complete (see [Wel93]). Known problems complete for $\#\mathrm{P}_1$ [Val79b] have the form: Given an integer n in unary, compute the number of graphs having n vertices and satisfying a fixed graph property π.

In Section 3, we will characterize the question of whether every P set has an easy census function in terms of collapses of language and function classes that are considered to be unlikely. In particular, every P set has an easy census function if and only if $\#\mathrm{P}_1 \subseteq \mathrm{FP}$. The main technical contribution in Section 3 is Theorem 3.7: $\#\mathrm{P}_1{}^{\mathrm{PH}}$ is contained in $\mathrm{FP}^{\#\mathrm{P}_1{}^{\#\mathrm{P}_1}}$. An immediate consequence of this result are

upward collapse results of the form: The collapse $\#_1 \cdot \mathrm{P} \subseteq \mathrm{FP}$ implies the collapse $\#_1 \cdot \mathrm{PH} \subseteq \mathrm{FP}$. Thus, every P set has an easy census function if and only if every set in the polynomial hierarchy has an easy census function. Note that the corresponding upward collapse for the $\#$ operator applied to the levels of PH follows immediately from the upward collapse property of the polynomial hierarchy itself: $\# \cdot \mathrm{P} \subseteq \mathrm{FP}$ implies $\mathrm{NP} = \mathrm{P}$ and, thus, $\mathrm{PH} = \mathrm{P}$; so, $\# \cdot \mathrm{PH} = \# \cdot \mathrm{P} \subseteq \mathrm{FP}$. However, for the $\#_1$ operator this upward collapse property is not so clear, since the assumption $\#_1 \cdot \mathrm{P} \subseteq \mathrm{FP}$ merely implies that all *tally* NP sets are in P (equivalently, $\mathrm{NE} = \mathrm{E}$), from which one cannot immediately conclude that $\#_1 \cdot \mathrm{PH}$ or even $\#_1 \cdot \mathrm{NP}$ is contained in FP. In fact, Hartmanis, Immerman, and Sewelson [HIS85] show that in some relativized world, $\mathrm{NE} = \mathrm{E}$ and yet the (weak) exponential-time hierarchy does not collapse. In light of this result, it is quite possible that the assumption of all tally NP sets being in P does not force all tally sets from higher levels of the polynomial hierarchy into P.

We also show that the assumption $\#\mathrm{P}_1 \subseteq \mathrm{FP}$ implies both $\mathrm{P} = \mathrm{BPP}$ and $\mathrm{PH} \subseteq \mathrm{MOD}_k\mathrm{P}$ for each $k \geqslant 2$ (Theorem 3.6). We also relate a set's property of having an easy census function to other well-studied properties of sets, such as rankability [GS91] and scalability [GH96]. In particular, though every rankable set has an easy census function, we show in Theorem 3.2 that (even when restricted to the sets in P) the converse is not true unless $\mathrm{P} = \mathrm{PP}$. Thus, Theorem 3.2 expands the result of Hemaspaandra and Rudich that every P set is rankable if and only if $\mathrm{P} = \mathrm{PP}$ [HR90] by showing that $\mathrm{P} = \mathrm{PP}$ is already implied by the apparently weaker hypothesis that every P set *with an easy census function* is rankable.

Cai and Hemaspaandra [CH89] introduced the notion of enumerative counting as a way of approximating the value of a $\#\mathrm{P}$ function deterministically in polynomial time. Hemaspaandra and Rudich [HR90] show that every P set is k-enumeratively rankable for some fixed k in polynomial time if and only if $\#\mathrm{P} = \mathrm{FP}$. They conclude that it is no more likely that one can enumeratively rank all sets in P than that one can exactly compute their ranking functions in polynomial time. In Section 4, we similarly characterize the question of whether every P set has a census function that is n^α-enumerable in time n^β for fixed constants α and β (equivalently, whether every $\#\mathrm{P}_1$ function is n^α-enumerable in time n^β). We show that this hypothesis implies $\#\mathrm{P}_1 \subseteq \mathrm{FP}$, and we conclude that it is no more likely that one can n^α-enumerate the census function of every P set in time n^β than that one can precisely compute its census function in polynomial time.

Finally, Section 5 provides a number of relativization results.

2. NOTATION AND DEFINITIONS

Fix the alphabet $\Sigma = \{0, 1\}$. Σ^* denotes the set of all strings over Σ, and $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$, where ε denotes the empty string. For any string $x \in \Sigma^*$, we denote the length of x by $|x|$. For any set $L \subseteq \Sigma^*$, the number of strings in L is denoted $|L|$, and the complement of L in Σ^* is denoted $\bar{L}$. Let $L^{=n}$ (resp. $L^{\leqslant n}$) denote the set of strings in L of length n (resp. of length at most n). As a shorthand, we use Σ^n to denote $(\Sigma^*)^{=n}$. For any set L, the *census function of* L, $census_L: \Sigma^* \to \mathbb{N}$, is

defined by $census_L(1^n) \stackrel{\text{df}}{=} |L^{=n}|$,[4] and χ_L denotes the *characteristic function of* L; i.e., $\chi_L(x) = 1$ if $x \in L$ and $\chi_L(x) = 0$ if $x \notin L$. A set S is said to be *sparse* if there is a polynomial p such that for each length n, $census_S(1^n) \leqslant p(n)$. A set T is said to be *tally* if $T \subseteq \{1\}^*$. To encode pairs of strings, we use a one-to-one, onto pairing function $\langle \cdot, \cdot \rangle : \Sigma^* \times \Sigma^* \to \Sigma^*$ that is computable and invertible in polynomial time; this pairing function is extended to encode m-tuples of strings as is standard. For convenience, we will sometimes write m-tuples of strings $x_1, x_2, ..., x_m \in \Sigma^*$ explicitly as $x_1 \# x_2 \# \cdots \# x_m$, using a special separating symbol $\#$ not in Σ. We let $\leqslant$ denote the standard lexicographic order on Σ^*.

The definition of Turing machines and their languages, Turing transducers and the functions they compute, relativized (i.e., oracle) computations, (relativized) complexity classes, etc. is standard in the literature (see, e.g., the textbooks [HU79, BC93, Pap94]). We briefly recall the complexity classes most important in this paper. FP denotes the class of polynomial-time computable functions. FP_1 is the class of functions computable in polynomial time by deterministic transducers with a unary input alphabet. FE is the class of functions that can be computed by deterministic transducers running in time 2^{cn} for some constant c.

Let $\text{E} \stackrel{\text{df}}{=} \bigcup_{c>0} \text{DTIME}[2^{cn}]$ and $\text{NE} \stackrel{\text{df}}{=} = \bigcup_{c>0} \text{NTIME}[2^{cn}]$. An unambiguous Turing machine is a nondeterministic Turing machine that on each input has at most one accepting path. UP [Val76] (respectively, UE) is the class of all languages accepted by some unambiguous Turing machine running in polynomial time (respectively, in time 2^{cn} for some constant c).

For any nondeterministic Turing machine M and any input $x \in \Sigma^*$, let $\text{acc}_M(x)$ denote the number of accepting computation paths of $M(x)$. A *spanP machine* [KST89] is an NP machine that has a special output device on which some output is printed for each accepting path. For any spanP machine M and any input $x \in \Sigma^*$, $\text{span}_M(x)$ is defined to be the number of distinct outputs of $M(x)$ if $M(x)$ has at least one accepting path, and 0 otherwise. A *tally NP machine* (respectively, a *tally spanP machine*) is an NP (respectively, a spanP) machine with a unary input alphabet.

DEFINITION 2.1. 1. [Val79a, Val79b]. $\#\text{P} \stackrel{\text{df}}{=} \{\text{acc}_M \mid M \text{ is an NP machine}\}$.

2. [Val79b]. $\#\text{P}_1 \stackrel{\text{df}}{=} \{\text{acc}_M \mid M \text{ is a tally NP machine}\}$.

3. [KST89]. $\text{spanP} \stackrel{\text{df}}{=} \{\text{span}_M \mid M \text{ is a spanP machine}\}$.

4. $\text{spanP}_1 \stackrel{\text{df}}{=} \{\text{span}_M \mid M \text{ is a tally spanP machine}\}$.

5. $\#\text{E} \stackrel{\text{df}}{=} \{\text{acc}_M \mid M \text{ is an NE machine}\}$.

6. [MS72, Sto77]. The polynomial hierarchy is inductively defined by $\Sigma_0^p \stackrel{\text{df}}{=} \text{P}$, $\Sigma_k^p \stackrel{\text{df}}{=} \text{NP}^{\Sigma_{k-1}^p}$ for $k \geqslant 1$, and $\text{PH} \stackrel{\text{df}}{=} \bigcup_{i \geqslant 0} \Sigma_i^p$.

7. [Gil77]. PP is the class of languages L for which there exist a set A in P and a polynomial p such that for all strings $x \in \Sigma^*$,

$$x \in L \Leftrightarrow |\{y \mid |y| = p(|x|) \text{ and } \langle x, y \rangle \in A\}| \geqslant 2^{p(|x|)-1}.$$

[4] The census function of L at n is often defined as the number of elements in L of length up to n in the literature. This definition and our definition are compatible, as long as our computability admits subtraction. We also note that we let $census_L$ map strings 1^n (as opposed to numbers n in binary notation) to $|L^{=n}|$ to emphasize that the input to the transducer computing $census_L$ is given in unary.

8. [Gil77]. BPP is the class of languages L for which there exist a set A in P and a polynomial p such that for all strings $x \in \Sigma^*$,

$$x \in L \Rightarrow |\{ y \mid |y| = p(|x|) \text{ and } \langle x, y \rangle \notin A\}| \leqslant 2^{p(|x|)-2};$$

$$x \notin L \Rightarrow |\{ y \mid |y| = p(|x|) \text{ and } \langle x, y \rangle \in A\}| \leqslant 2^{p(|x|)-2}.$$

9. [CH90, Her90, BG92]. For any fixed $k \geqslant 2$, $\mathrm{MOD}_k\mathrm{P}$ is the class of languages L for which there exist a set A in P and a polynomial p such that for all strings $x \in \Sigma^*$,

$$x \in L \Leftrightarrow |\{ y \mid |y| = p(|x|) \text{ and } \langle x, y \rangle \in A\}| \not\equiv 0 \bmod k.$$

If $k = 2$, we write $\oplus\mathrm{P}$ (introduced in [PZ83, GP86]), instead of $\mathrm{MOD}_2\mathrm{P}$.

10. [OH93, FFK94]. SPP is the class of languages L for which there exist a set A in P and a polynomial p such that for all strings $x \in \Sigma^*$,

$$x \in L \Rightarrow |\{ y \mid |y| = p(|x|) \text{ and } \langle x, y \rangle \in A\}| = 2^{p(|x|)-1} + 1;$$

$$x \notin L \Rightarrow |\{ y \mid |y| = p(|x|) \text{ and } \langle x, y \rangle \in A\}| = 2^{p(|x|)-1}.$$

11. [KL80]. For any language class $\mathscr{C}$, let $\mathscr{C}/\mathrm{poly}$ be the class of all languages L for which there exist a set $A \in \mathscr{C}$, a polynomial p, and an advice function $h: \Sigma^* \to \Sigma^*$ such that for each length n, $|h(1^n)| = p(n)$, and for every $x \in \Sigma^*$, $x \in L$ if and only if $\langle x, h(1^{|x|}) \rangle \in A$. For any function class $\mathscr{F}$, let $\mathscr{F}/\mathrm{poly}$ be the class of all functions g for which there exist a function $f \in \mathscr{F}$, a polynomial p, and an advice function $h: \Sigma^* \to \Sigma^*$ such that for each length n, $|h(1^n)| = p(n)$, and for every $x \in \Sigma^*$, $g(x) = f(\langle x, h(1^{|x|}) \rangle)$.

We will use the common operator notation at times in order to generalize function classes such as $\#\mathrm{P}$ and $\#\mathrm{P}_1$.

DEFINITION 2.2. For any language class $\mathscr{C}$, define the function classes $\# \cdot \mathscr{C}$ and $\#_1 \cdot \mathscr{C}$ as follows:

1. $\# \cdot \mathscr{C}$ is the class of functions $f: \Sigma^* \to \mathbb{N}$ for which there exist a set $A \in \mathscr{C}$ and a polynomial p such that for each $x \in \Sigma^*$,

$$f(x) = |\{ y \mid |y| = p(|x|) \text{ and } \langle x, y \rangle \in A\}|.$$

2. $\#_1 \cdot \mathscr{C}$ is the class of functions $f: \Sigma^* \to \mathbb{N}$ for which there exist a set $A \in \mathscr{C}$ and a polynomial p such that for each $n \in \mathbb{N}$,

$$f(1^n) = |\{ y \mid |y| = p(n) \text{ and } \langle 1^n, y \in A \rangle\}|.$$

As stated below, both operators, $\#$ and $\#_1$, are monotonic. Since this property follows immediately from the definitions, we omit the proof of the following proposition.

PROPOSITION 2.3. *Let $\mathscr{C}$ and $\mathscr{D}$ be any classes of sets.*

1. *If $\mathscr{C} \subseteq \mathscr{D}$, then $\# \cdot \mathscr{C} \subseteq \# \cdot \mathscr{D}$.*
2. *If $\mathscr{C} \subseteq \mathscr{D}$, then $\#_1 \cdot \mathscr{C} \subseteq \#_1 \cdot \mathscr{D}$.*

Next, we gather some easy observations regarding equivalent formulations of the classes $\# \mathrm{P}^{\mathrm{PH}}$ and $\# \mathrm{P}_1{}^{\mathrm{PH}}$ to be used in Section 3. Analogs of Proposition 2.4 for classes other than PH as the oracle class could be stated as well; we focus here on the class of interest to us.

For any language class $\mathscr{C}$, we write $\mathrm{P}^{\mathscr{C}[1]}$ to indicate that on every input in the $\mathrm{P}^{\mathscr{C}}$ computation at most one call to the $\mathscr{C}$ oracle is allowed. Similarly, for any function class $\mathscr{F}$, we write $\mathrm{P}^{\mathscr{F}[1]}$ to indicate that on every input in the $\mathrm{P}^{\mathscr{F}}$ computation at most one call to the function oracle from $\mathscr{F}$ is allowed.

PROPOSITION 2.4. *Three statements are true:*

1. $\mathrm{P}^{\mathrm{PH}} = \mathrm{P}^{\mathrm{PH}[1]}$.
2. $\#_1 \cdot \mathrm{P}^{\mathrm{PH}} = \#_1 \cdot \mathrm{P}^{\mathrm{PH}[1]} = \#_1 \cdot \mathrm{PH} = \# \mathrm{P}_1{}^{\mathrm{PH}[1]} = \# \mathrm{P}_1{}^{\mathrm{PH}}$.
3. $\# \cdot \mathrm{P}^{\mathrm{PH}} = \# \cdot \mathrm{P}^{\mathrm{PH}[1]} = \# \cdot \mathrm{PH} = \# \mathrm{P}^{\mathrm{PH}[1]} = \# \mathrm{P}^{\mathrm{PH}}$.

Proof. Part 1 follows immediately from the fact that PH is closed under polynomial-time Turing reductions. That is,

$$\mathrm{P}^{\mathrm{PH}} \subseteq \mathrm{PH} \subseteq \mathrm{P}^{\mathrm{PH}[1]} \subseteq \mathrm{P}^{\mathrm{PH}};$$

so, the above inclusions are equalities.

Part 2. From the proof of part 1 and the monotonicity of the $\#_1$ operator (see Proposition 2.3), we have the first two equalities: $\#_1 \cdot \mathrm{P}^{\mathrm{PH}} = \#_1 \cdot \mathrm{P}^{\mathrm{PH}[1]} = \#_1 \cdot \mathrm{PH}$.

To see that $\#_1 \cdot \mathrm{PH} \subseteq \# \mathrm{P}_1{}^{\mathrm{PH}[1]}$, let $f \in \#_1 \cdot \mathrm{PH}$ be witnessed by a set $A \in \mathrm{PH}$ and a polynomial p; i.e., for each $n \in \mathbb{N}$,

$$f(1^n) = |\{ y \mid |y| = p(n) \text{ and } \langle 1^n, y \in A \rangle \}|.$$

Consider the following tally NP oracle machine M. On input 1^n, M with oracle A guesses a string y of length $p(n)$, and for each y guessed, M accepts if and only if $\langle 1^n, y \rangle \in A$. Hence, $f \in \# \mathrm{P}_1{}^{\mathrm{PH}[1]}$.

Since $\# \mathrm{P}_1{}^{\mathrm{PH}[1]} \subseteq \# \mathrm{P}_1{}^{\mathrm{PH}}$, it remains to show that $\# \mathrm{P}_1{}^{\mathrm{PH}} \subseteq \#_1 \cdot \mathrm{PH}$. Let $f \in \# \mathrm{P}_1{}^{\mathrm{PH}}$ be witnessed by a tally NP oracle machine M with oracle $A \in \mathrm{PH}$; i.e., $f = \mathrm{acc}_{M^A}$. We assume that all computation paths of M on input 1^n are encoded as strings in $\{0, 1\}^{p(n)}$ for some polynomial p, where the oracle queries that are asked on such a path and the corresponding answers are part of the encoding string. Define B to be the set of all strings $\langle 1^n, y \rangle$ such that

- $n \in \mathbb{N}$,
- $y \in \{0, 1\}^{p(n)}$ encodes an accepting computation path of $M^A(1^n)$ with oracle queries $q_1, q_2, \ldots, q_k$, and

• for each i with $1 \leqslant i \leqslant k$, $M^A(1^n)$ on path y proceeds in the "yes" state if and only if $q_i \in A$.

It follows that $B \in \mathrm{PH}$ and that for each $n \in \mathbb{N}$,

$$f(1^n) = |\{ y \mid |y| = p(n) \text{ and } \langle 1^n, y \in A \rangle \}|.$$

Hence, $f \in \#_1 \cdot \mathrm{PH}$, completing the proof of part 2.

The proof of part 3 is analogous to the proof of part 2. ▌

DEFINITION 2.5. 1. A bijection $\phi: \Sigma^* \to \Sigma^*$ is a *P-isomorphism* if ϕ is computable and invertible in polynomial time.

2. A P-isomorphism ϕ is *length-preserving* if for all $x \in \Sigma^*$, $|\phi(x)| = |x|$.

3. A P-isomorphism ϕ mapping set $A \subseteq \Sigma^*$ to set $B \subseteq \Sigma^*$ is *order-preserving* if for any two strings x and y satisfying either $x, y \in A$ or $x, y \notin A$, if $x \leqslant y$ then $\phi(x) \leqslant \phi(y)$.

DEFINITION 2.6 [GS91]. The *ranking function* of a language $A \subseteq \Sigma^*$ is the function $r: \Sigma^* \to \mathbb{N}$ that maps each $x \in \Sigma^*$ to $|\{ y \leqslant x \mid y \in A \}|$. A language A is *rankable* if its ranking function is computable in polynomial time.

Goldsmith and Homer [GH96] introduced the property of scalability, a more flexible notion than rankability in which the rank of some given element within the set is not necessarily determined with respect to the lexicographic order of Σ^*, but rather with respect to *any* well-ordering of Σ^* that can be "scaled" by a polynomial-time computable and polynomial-time invertible bijection between $\mathbb{N}$ and Σ^*. Equivalently, the scalable sets are precisely those that are P-isomorphic to some rankable set [GH96]. The definition below is based on this characterization.

DEFINITION 2.7 [GH96]. A language A is *scalable* if it is P-isomorphic to a rankable set. For any oracle X, the *X-scalable* sets are those that are P^X-isomorphic to some set rankable in FP^X.

3. DOES P HAVE EASY CENSUS FUNCTIONS?

We start by exploring the relationships between the properties of a set being rankable, being scalable, and having an easy census function. Let A be any set (not necessarily in P). Consider the following five conditions:

(i) A is rankable.

(ii) A has an easy census function.

(iii) A is P-isomorphic to some rankable set (i.e., A is scalable).

(iv) A is P-isomorphic to some rankable set via some length-preserving isomorphism.

(v) A is P-isomorphic to some rankable set via some order-preserving isomorphism.

It is immediately clear that for any set A, (i) implies each of (ii), (iv), and (v), and each of (iv) and (v) implies (iii). The next proposition shows that the rankable sets are closed under order-preserving P-isomorphisms (thus, conditions (i) and (v), in fact, are equivalent) and that the class of sets having an easy census function is closed under length-preserving P-isomorphisms. The latter fact immediately gives that (iv) implies (ii), since every rankable set has an easy census function. The inclusion structure of the sets in P satisfying properties (i) through (iv) is given in Fig. 1.

PROPOSITION 3.1. *Two statements are true:*

1. *The class of all rankable sets is closed under order-preserving* P-*isomorphisms.*

2. *The class of sets having an* FP-*computable census function is closed under length-preserving* P-*isomorphisms.*

Proof. (1) Let A be P-isomorphic to a rankable set, B, via some order-preserving isomorphism, ϕ. Since B is rankable, $\bar{B}$ is rankable. Let respectively r and $\bar{r}$ be the ranking functions for B and $\bar{B}$. For any string $x \in \Sigma^*$, let $\mathrm{lex}(x)$ denote the lexicographic order of x, i.e., the number of strings $w \in \Sigma^*$ with $w \leqslant x$. Define the function

$$r'(x) \stackrel{\mathrm{df}}{=} \begin{cases} r(\phi(x)) & \text{if } x \in A, \\ \mathrm{lex}(x) - \bar{r}(\phi(x)) & \text{if } x \notin A. \end{cases}$$

Clearly, r' is computable in polynomial time and r' is the ranking function for A.

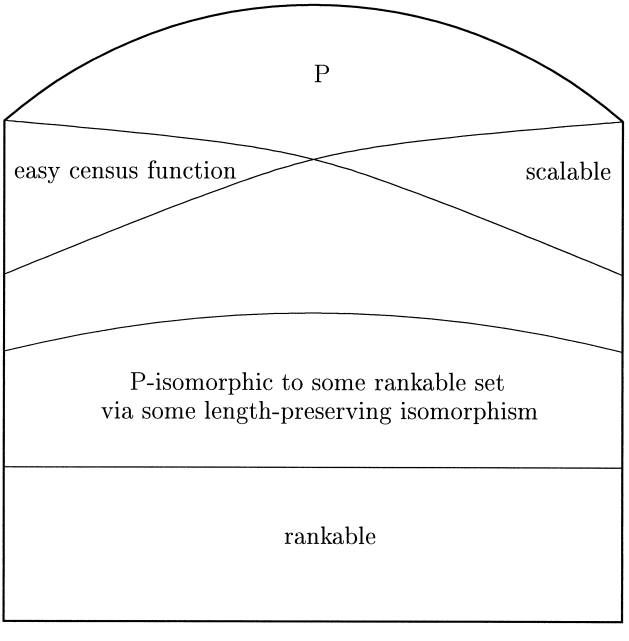

FIG. 1. Inclusion structure of the sets in P satisfying properties (i) through (iv).

(2) Let A be P-isomorphic to a set B via some length-preserving isomorphism ϕ, let $census_B \in \mathrm{FP}$. Then, for each n, $\phi(A^{=n}) = B^{=n}$. Thus, $census_A = census_B$, which implies $census_A \in \mathrm{FP}$. ∎

So we are left with only the four conditions (i) to (iv). Since there are nonrecursive sets with an FP-computable census function, but any set satisfying one of (i), (iii), or (iv) is in P, condition (ii) in general cannot imply any of the other three conditions. On the other hand, when we restrict our attention to the sets in P having easy census functions, we can show that (ii) implies (i) if and only if $\mathrm{P} = \mathrm{PP}$. Thus, even when restricted to P sets, it is unlikely that (ii) is equivalent to (i).

THEOREM 3.2. *All* P *sets with an easy census function are rankable if and only if* $\mathrm{P} = \mathrm{PP}$.

Proof. Hemaspaandra and Rudich [HR90] show that $\mathrm{P}^{\#\mathrm{P}} = \mathrm{P}$ if and only if every P set is rankable. Noticing that $\mathrm{P} = \mathrm{PP}$ is equivalent to $\mathrm{P}^{\#\mathrm{P}} = \mathrm{P}$, this result in particular implies that every P set with an easy census function is rankable if $\mathrm{P} = \mathrm{PP}$.

Conversely, assume that every P set with an easy census function is rankable. We show that this assumption implies $\mathrm{P} = \mathrm{PP}$. Let L be any set in PP, and let A be a set in P and let p be a polynomial such that for all $x \in \Sigma^*$:

$$x \in L \Leftrightarrow |\{ y \mid |y| = p(|x|) \text{ and } x \# y \in A\}| \geqslant 2^{p(|x|)-1}.$$

Define

$$T \stackrel{\mathrm{df}}{=} \{b \# x \# y \mid x, y \in \Sigma^*, |y| = p(|x|), b \in \{0, 1\}, \text{ and } \chi_A(x \# y) = b\}.$$

Clearly, $T \in \mathrm{P}$. Also, the census function of T is easy to compute: Given n in unary, compute the largest integer i such that $i + p(i) + 3 \leqslant n$. Then,

$$census_T(1^n) = \begin{cases} 2^{i+p(i)} & \text{if } \ i + p(i) + 3 = n \\ 0 & \text{if } \ i + p(i) + 3 < n. \end{cases}$$

Since $T \in \mathrm{P}$ and $census_T \in \mathrm{FP}$, our hypothesis implies that T is rankable. Let r be the ranking function for T. For each $x \in \Sigma^+$, let $\hat{x}$ denote the lexicographic predecessor of x. Note that, for each $x \in \Sigma^+$, $r(0 \# x \# 1^{p(|x|)}) - r(1 \# \hat{x} \# 1^{p(|\hat{x}|)})$ gives the number of strings y of length $p(|x|)$ such that $x \# y \notin A$. Hence, for each $x \in \Sigma^+$,

$$x \in L \Leftrightarrow r(0 \# x \# 1^{p(|x|)}) - r(1 \# \hat{x} \# 1^{p(|\hat{x}|)}) \leqslant 2^{p(|x|)-1}.$$

Since the predicate on the right-hand side of the above equivalence can be decided in polynomial time, it follows that $L \in \mathrm{P}$. ∎

COROLLARY 3.3. *All* P *sets are rankable if and only if all sets in* P *with an easy census function are rankable.*

One might ask whether or not all P sets outright have an easy census function (which, if true, would make Corollary 3.3 trivial). The following characterization of this question in terms of unlikely collapses of certain function and language classes

suggests that this probably is not true. Thus, Corollary 3.3 is nontrivial with the same certainty with which we believe that for instance not all $\#\mathrm{P}_1$ functions are in FP.[5]

THEOREM 3.4. *Five statements are equivalent:*

1. *Every* P *set has an* FP*-computable census function.*
2. $\#\mathrm{P}_1 \subseteq \mathrm{FP}$.
3. $\#\mathrm{E} = \mathrm{FE}$.
4. $\mathrm{P}^{\#\mathrm{P}_1} = \mathrm{P}$.
5. *For every language* L *accepted by a logspace-uniform depth* 2 *AND-OR circuit family of bottom fan-in* 2, $census_L$ *is in* FP.

Proof. To show that (1) implies (2), let f be any function in $\#\mathrm{P}_1$. Let M be some tally NP machine with $\mathrm{acc}_M = f$. Assume that M runs in time n^k, for some constant k. Define

$$A \stackrel{\mathrm{df}}{=} \{x \mid |x| = n^k \text{ for some } n \text{ and } x \text{ encodes an accepting path of } M(1^n)\}.$$

Clearly, A is in P (note that n can be found in polynomial time, since computing the kth root of some integer can be done in polynomial time). Now, from our hypothesis it follows that $census_A$ is in FP, and since $census_A = \mathrm{acc}_M$, we have $f \in \mathrm{FP}$.

Conversely, let A be an arbitrary set in P. Define M to be the tally NP machine that, on input 1^n, guesses an $x \in \{0, 1\}^n$, and for each x guessed, accepts along the path for x if and only if $x \in A$. Then, $\mathrm{acc}_M = census_A$. Since by hypothesis $\mathrm{acc}_M \in \mathrm{FP}$, it follows that $census_A \in \mathrm{FP}$.

The equivalence of (2) and (3) can be proven by means of standard translation—this is essentially the function analog of Book's result that every tally NP set is in P if and only if $\mathrm{NE} = \mathrm{E}$ [Boo74]; see [Har83, HIS85] for the extension of this result to sparse sets.

The equivalence of (2) and (4) is straightforward.

It is easy to see that (2) implies (5). In order to prove that (5) implies (2), note that computing the number of satisfying assignments for monotone 2CNF formulas is complete for $\#\mathrm{P}$ [Val79b] under logspace reductions. Now, given a function f in $\#\mathrm{P}_1$, there exist logspace computable functions R, S, ρ such that for all n, $R(1^n)$ is a monotone 2CNF formula with $\rho(1^n)$ variables, and $f(1^n)$ equals the number of satisfying assignments for $R(1^n)$ divided by $S(1^n)$. The reduction R can be modified so that for every n, $\rho(1^{n+1}) > \rho(1^n)$. Now let C_m be the circuit defined as follows: (a) if $m = \rho(1^n)$ for some n, then C_m is a depth 2 AND-OR circuit that tests whether an assignment, given as the input, satisfies $R(1^n)$; (b) if not, C_m is a depth 1 AND circuit that rejects all inputs. This circuit family $F = \{C_m\}_{m \geqslant 1}$ is logspace-uniform. Now

[5] It is not difficult to construct—by standard techniques—an oracle relative to which $\#\mathrm{P}_1 \not\subseteq \mathrm{FP}$. On the other hand, we will show in Section 5 that, relative to some oracle, $\#\mathrm{P}_1 \subseteq \mathrm{FP}$, yet $\#\mathrm{P} \neq \mathrm{FP}$ (and thus, $\mathrm{PP} \neq \mathrm{P}$).

let A be the language accepted by F. Then, for every n, $f(1^n) = census_A(1^{\rho(1^n)})/S(1^n)$. Thus, (5) implies that $f \in \mathrm{FP}$. ∎

Theorem 3.4 can as well be stated for more general classes than $\#\mathrm{P}_1 = \#_1 \cdot \mathrm{P}$. In particular, this comment applies to $\#_1 \cdot \mathscr{C}$, where for instance $\mathscr{C} = \mathrm{NP}$ or $\mathscr{C} = \mathrm{PH}$. Noticing that $\mathrm{span}\mathrm{P}_1 = \#_1 \cdot \mathrm{NP}$ and focusing on the first two conditions of Theorem 3.4, this observation is exemplified as follows.

THEOREM 3.5. *Two statements are true:*

1. *Every* NP *set has an* FP*-computable census function if and only if* $\mathrm{span}\mathrm{P}_1 \subseteq \mathrm{FP}$.

2. *Every set in* PH *has an* FP*-computable census function if and only if* $\#_1 \cdot \mathrm{PH} \subseteq \mathrm{FP}$.

We will show later that the conditions of Theorem 3.4 in fact are equivalent to the two conditions stated in either part of Theorem 3.5.

We now give two more consequences of the assumption $\#\mathrm{P}_1 \subseteq \mathrm{FP}$.

THEOREM 3.6. *If* $\#\mathrm{P}_1 \subseteq \mathrm{FP}$, *then two statements are true:*

1. *For any fixed* $k \geqslant 2$, $\mathrm{PH} \subseteq \mathrm{MOD}_k\mathrm{P}$.
2. $\mathrm{P} = \mathrm{BPP}$.

Proof. Suppose $\#\mathrm{P}_1 \subseteq \mathrm{FP}$. In order to prove the first part, note that if a natural number $k \geqslant 2$ has prime factorization of the form $p_1^{e_1} \cdots p_t^{e_t}$, then $\mathrm{MOD}_k\mathrm{P} = \{L_1 \cap \cdots \cap L_t \mid L_1 \in \mathrm{MOD}_{p_1}\mathrm{P}, ..., L_t \in \mathrm{MOD}_{p_t}\mathrm{P}\}$ [Her90, BG92]. Thus it suffices to show that for every prime $k \geqslant 2$, $\mathrm{PH} \subseteq \mathrm{MOD}_k\mathrm{P}$.

We claim that for every prime $k \geqslant 2$, each language in PH belongs to $\mathrm{MOD}_k\mathrm{P}/\mathrm{poly}$ with an advice function in $\mathrm{FP}^{\#\mathrm{P}[1]}$. To prove the claim, let L be any language in PH and $k \geqslant 2$ be any prime number. Toda and Ogihara [TO92] prove that some $A \in \mathrm{MOD}_k\mathrm{P}$ witnesses that $L \in \mathrm{MOD}_k\mathrm{P}/\mathrm{poly}$, together with some polynomially length-bounded advice function. Fix such an A and define

$$B \stackrel{\mathrm{df}}{=} \{\langle 1^n, w\rangle \mid n \geqslant 1 \text{ and for every } x,\ |x| = n,\ x \in L \text{ if and only if } \langle x, w\rangle \in A\}.$$

Define f to be the function that for each n, maps 1^n to the lexicographically smallest string w such that $\langle 1^n, w\rangle \in B$. Since $L \in \mathrm{PH}$ and $A \in \mathrm{MOD}_k\mathrm{P}$, B belongs to $\mathrm{coNP}^{\mathrm{PH} \cup \mathrm{MOD}_k\mathrm{P}}$, which is included in $\mathrm{PH}^{\mathrm{MOD}_k\mathrm{P}}$. Then f is total and $f \in \mathrm{FP}^{\mathrm{NP}^B} \subseteq \mathrm{FP}^{\mathrm{PH}^{\mathrm{MOD}_k\mathrm{P}}}$. Toda and Ogihara show that $\mathrm{PH}^{\mathrm{MOD}_k\mathrm{P}} \subseteq \mathrm{BPP}^{\mathrm{MOD}_k\mathrm{P}}$. A part of the proof of Toda's theorem [Tod91] shows $\mathrm{BPP}^{\oplus\mathrm{P}} \subseteq \mathrm{P}^{\#\mathrm{P}[1]}$. By following the same argument one can show that for every prime $k \geqslant 2$, $\mathrm{BPP}^{\mathrm{MOD}_k\mathrm{P}} \subseteq \mathrm{P}^{\#\mathrm{P}[1]}$, completing the proof of the claim.

Since f can be computed in $\mathrm{FP}^{\#\mathrm{P}[1]}$, the set

$$\{\langle 1^n, i, b\rangle \mid n \geqslant 1, b \in \{0, 1\}, \text{ and the } i\text{th bit of } f(1^n) \text{ is } b\}$$

can be decided in polynomial time with one query to a suitable function h in $\#\mathrm{P}$. Let $q_{n,i}$ be the string that is queried on input $\langle 1^n, i, b\rangle$. Define a $\#\mathrm{P}_1$ function g by

$$g(1^n) \stackrel{\mathrm{df}}{=} \langle h(q_{n,1}), h(q_{n,2}), \ldots, h(q_{n,p(n)})\rangle,$$

where p is a polynomial bounding the length of the advice and the value of $g(1^n)$ is viewed as a number written in binary. One query to g will allow us to compute f; i.e., $f \in \mathrm{FP}_1^{\#\mathrm{P}_1[1]}$.

Applying our supposition $\#\mathrm{P}_1 \subseteq \mathrm{FP}$, we conclude that f can be computed in polynomial time. Since L is in $\mathrm{MOD}_k\mathrm{P}/\mathrm{poly}$ with polynomial-time computable advice, it follows that $L \in \mathrm{MOD}_k\mathrm{P}$. Hence, $\mathrm{PH} \subseteq \mathrm{MOD}_k\mathrm{P}$.

In order to prove the second part, note that $\mathrm{BPP} \subseteq \mathrm{P}/\mathrm{poly}$ [Adl78] and $\mathrm{BPP} \subseteq \mathrm{PH}$ [Sip83, Lau83]. By following the proof of the first part with P in place of $\mathrm{Mod}_k\mathrm{P}$ we obtain that $\mathrm{BPP} \subseteq \mathrm{P}$. ∎

Now we show that the conditions of Theorem 3.4 in fact are equivalent to the two conditions stated in either part of Theorem 3.5. To this end, we establish the following theorem, which is interesting in its own right. Theorem 3.7 is the main technical contribution in this section.

THEOREM 3.7. $\#\mathrm{P}_1^{\mathrm{PH}} \subseteq \mathrm{FP}^{\#\mathrm{P}_1^{\#\mathrm{P}_1}}$.

Before we turn to the proof of Theorem 3.7, we discuss some issues related to this result.

First, we stress that Theorem 3.7 is a novel insight and does not trivially follow from known results. In particular, Toda's result that $\mathrm{PH} \subseteq \mathrm{P}^{\#\mathrm{P}[1]}$ does not imply Theorem 3.7 in any obvious way. Note that Toda's theorem does imply the following two inclusions[6]:

$$\#\mathrm{P}_1^{\mathrm{PH}} \subseteq \#\mathrm{P}_1^{\#\mathrm{P}[1]}; \tag{1}$$

$$\#\mathrm{P}^{\mathrm{PH}} \subseteq \#\mathrm{P}^{\#\mathrm{P}[1]}. \tag{2}$$

Observe, however, that the oracles on the right-hand sides of the inclusions (1) and (2) are $\#\mathrm{P}$ functions. In contrast, Theorem 3.7 establishes containment of $\#\mathrm{P}_1^{\mathrm{PH}}$ in a class in which only $\#\mathrm{P}_1$ oracles occur. Although our proof also applies the techniques of Toda [Tod91] and Toda and Ogihara [TO92], our *result* seems to be incomparable with the above consequence (1) of Toda's theorem.

Second, can Theorem 3.7 be strengthened to $\mathrm{FP}^{\mathrm{PH}}$ or even $\#\mathrm{P}^{\mathrm{PH}}$ being contained in $\mathrm{FP}^{\#\mathrm{P}_1^{\#\mathrm{P}_1}}$? We note that the containment $\mathrm{FP}^{\mathrm{PH}} \subseteq \mathrm{FP}^{\#\mathrm{P}_1^{\#\mathrm{P}_1}}$ appears to be unlikely, since it would imply that $\mathrm{FP}^{\mathrm{PH}} \subseteq \mathrm{FP}/\mathrm{poly}$. In turn, the assumption $\mathrm{FP}^{\mathrm{PH}} \subseteq \mathrm{FP}/\mathrm{poly}$ implies that the polynomial hierarchy has polynomial-size circuits

[6] From part 1 of Proposition 2.4 and from Toda's Theorem, we have $\mathrm{P}^{\mathrm{PH}} = \mathrm{P}^{\mathrm{PH}[1]} \subseteq \mathrm{P}^{\#\mathrm{P}[1][1]} = \mathrm{P}^{\#\mathrm{P}[1]}$. The inclusions (1) and (2) now follow from Proposition 2.3, the equalities $\#_1 \cdot \mathrm{P}^{\mathrm{PH}} = \#\mathrm{P}_1^{\mathrm{PH}}$ and $\# \cdot \mathrm{P}^{\mathrm{PH}} = \#\mathrm{P}^{\mathrm{PH}}$ from parts 2 and 3 of Proposition 2.4, and the similar observations that $\#_1 \cdot \mathrm{P}^{\#\mathrm{P}[1]} = \#\mathrm{P}_1^{\#\mathrm{P}[1]}$ and $\# \cdot \mathrm{P}^{\#\mathrm{P}[1]} = \#\mathrm{P}^{\#\mathrm{P}[1]}$.

and, thus, collapses by the result of Karp and Lipton [KL80]. In contrast, the inclusion $\mathrm{FP}_1{}^{\mathrm{PH}} \subseteq \mathrm{FP}_1/\mathrm{poly}$, which indeed does follow from Theorem 3.7, merely implies that all *tally* sets in PH have polynomial-size circuits, a true statement that has no unlikely consequences. Indeed, P/poly is known to contain *all* tally sets and even the Turing closure of the sparse sets.

Third, we mention that Theorem 3.7, nonetheless, can be strengthened. Note that our proof below will make use of Toda and Ogihara's [TO92] result that $\mathrm{PH} \subseteq \bigoplus \mathrm{P}/\mathrm{poly}$. Since Toda and Ogihara [TO92] also showed that $\bigoplus \mathrm{P}^{\mathrm{PH}}/\mathrm{poly} = \bigoplus \mathrm{P}/\mathrm{poly}$, and so $\bigoplus \mathrm{P}^{\mathrm{PH}} \subseteq \bigoplus \mathrm{P}/\mathrm{poly}$, Theorem 3.7 and its corollaries could be stated even with PH replaced by $\bigoplus \mathrm{P}^{\mathrm{PH}}$. However, we focus on the PH case, as this is a more natural and more central class.

Now, we turn to the proof of Theorem 3.7.

Proof of Theorem 3.7. Let f be any function in $\#\mathrm{P}_1{}^{\mathrm{PH}}$. By part 2 of Proposition 2.4, $\#\mathrm{P}_1{}^{\mathrm{PH}} = \#_1 \cdot \mathrm{PH}$. Thus, there exist a set $\hat{L} \in \mathrm{PH}$ and a polynomial $\hat{p}$ such that for each length n, $f(1^n) = |\{ y \in \{0,1\}^{\hat{p}(n)} \mid 1^n \# y \in \hat{L}\}|$.

Before we proceed with the proof, a technical point needs to be discussed. For the calculations in the final paragraph of this proof, it would be useful to have a polynomial $\hat{p}$ satisfying that for each n, $\log \hat{p}(n)$ is an integer; i.e., $\hat{p}(n)$ is a power of two. Since that cannot be assumed in general, we define a function $p: \mathbb{N} \to \mathbb{N}$ as follows. For each n, $p(n)$ is the smallest power of 2 such that $\hat{p}(n) \leqslant p(n)$. Since for every integer there is a power of 2 that is at most double that integer, we have $p(n) \leqslant 2\hat{p}(n)$; so, p still is polynomially bounded in n. Define a padded version L of $\hat{L}$ by

$$L \stackrel{\mathrm{df}}{=} \{1^n \# yw \mid n \in \mathbb{N} \wedge |y| = \hat{p}(n) \wedge 1^n \# y \in \hat{L} \wedge w = 0^{p(n) - \hat{p}(n)}\}.$$

It follows that $L \in \mathrm{PH}$ and for each length n, $f(1^n) = |\{ y \in \{0,1\}^{p(n)} \mid 1^n \# y \in L\}|$.

By Toda and Ogihara's result that $\mathrm{PH} \subseteq \bigoplus \mathrm{P}/\mathrm{poly}$ [TO92], there exist a set $A \in \bigoplus \mathrm{P}$, an advice function $h: \Sigma^* \to \Sigma^*$, and a polynomial q such that for each length m and each x of length m, it holds that $|h(1^m)| = q(m)$, and $x \in L$ if and only if $\langle x, h(1^m)\rangle \in A$. By the argument given in the proof of Theorem 3.6, h is computable in $\mathrm{FP}_1{}^{\#\mathrm{P}_1[1]}$. Let M be a machine witnessing that $A \in \bigoplus \mathrm{P}$, i.e., for every string z, $z \in A$ if and only if $\mathrm{acc}_M(z)$ is odd.

Toda [Tod91] defined inductively the sequence of polynomials: For each $j \in \mathbb{N}$, define $s_0(j) \stackrel{\mathrm{df}}{=} j$, and for each $j \in \mathbb{N}$ and $i > 0$, define

$$s_i(j) \stackrel{\mathrm{df}}{=} 3(s_{i-1}(j))^4 + 4(s_{i-1}(j))^3.$$

One very useful property of this sequence of polynomials is that for all $i, j \in \mathbb{N}$, it holds that $s_i(j) = c_j \cdot 2^{2^i}$ for some $c_j \in \mathbb{N}$ if j is even, and $s_i(j) = d_j \cdot 2^{2^i} - 1$ for some $d_j \in \mathbb{N}$ if j is odd; see Toda [Tod91] for the induction proof.

We describe a polynomial-time oracle transducer T that, on input 1^n, invokes its $\#\mathrm{P}_1{}^{\#\mathrm{P}_1}$ function oracle g on 1^n, receives the number $g(1^n)$ written in binary, and then prints in binary the number $f(1^n)$. Formally, function g is defined by

$$g(1^n) \stackrel{\mathrm{df}}{=} \sum_{y \in \{0,1\}^{p(n)}} (s_{\ell_n}(\mathrm{acc}_M(\langle 1^n \# y, h(1^{n+1+p(n)})\rangle)))^2,$$

where $\ell_n \stackrel{\mathrm{df}}{=} \log p(n)$.

Intuitively, the fact that g is in $\#\mathrm{P}_1{}^{\#\mathrm{P}_1}$ follows from the properties of the Toda polynomials, from the closure of $\#\mathrm{P}$ under strong sum and product,[7] and from the fact that advice function h is computable in $\mathrm{FP}_1{}^{\#\mathrm{P}_1[1]}$.

More formally, to show that $g \in \#\mathrm{P}_1{}^{\#\mathrm{P}_1}$, we describe a tally NP oracle machine G and a $\#\mathrm{P}_1$ oracle $\hat{g}$ for G such that, for every n, the number of accepting paths of G on input 1^n with oracle $\hat{g}$ equals $g(1^n)$. On input 1^n, G first gets the advice string $a_n = h(1^{n+1+p(n)})$ of length $q(n+1+p(n))$ via one call to some appropriate $\#\mathrm{P}_1$ oracle, call it $\hat{g}$. This is possible by the argument given in the proof of Theorem 3.6, which shows how to construct $\hat{g}$. Then, G guesses a string y of length $p(n)$, and for each y guessed it proceeds as follows. For fixed n and y of length $p(n)$, let $j_{n,y}$ be a shorthand for $\mathrm{acc}_M(\langle 1^n \# y, a_n \rangle)$. Note that, for given n and y of length $p(n)$, $(s_{\ell_n}(j_{n,y}))^2$ is a polynomial in $j_{n,y}$ of degree $2^{2\ell_n+1}$, which is polynomial in n. Also, the coefficients of the polynomial $(s_{\ell_n}(j_{n,y}))^2$ are deterministically computable in time polynomial in n; see Toda [Tod91]. Since $\mathrm{acc}_M \in \#\mathrm{P}$ and $\#\mathrm{P}$ is closed under strong sum and product (see Footnote 7), the function mapping $\langle 1^n \# y, a_n \rangle$ to $(s_{\ell_n}(j_{n,y}))^2$ is in $\#\mathrm{P}$. Let $\tilde{G}$ be an NP machine witnessing that this function is in $\#\mathrm{P}$. Then, G on input 1^n can for each guessed y produce exactly $(s_{\ell_n}(j_{n,y}))^2$ accepting paths by simulating $\tilde{G}$ on input $\langle 1^n \# y, a_n \rangle$. Again using the closure of $\#\mathrm{P}$ under strong sum, it follows that $g \in \#\mathrm{P}_1{}^{\#\mathrm{P}_1}$, as claimed.

By the above properties of the Toda polynomials, it follows that for each n and for each y of length $p(n)$, if $j_{n,y}$ is even then $s_{\ell_n}(j_{n,y}) = c_{j_{n,y}} \cdot 2^{2^{\ell_n}}$ for some $c_{j_{n,y}} \in \mathbb{N}$, and if $j_{n,y}$ is odd then $s_{\ell_n}(j_{n,y}) = d_{j_{n,y}} \cdot 2^{2^{\ell_n}} - 1$ for some $d_{j_{n,y}} \in \mathbb{N}$.

Thus, recalling that $2^{\ell_n} = p(n)$, we have

$$j_{n,y} \text{ is even} \Rightarrow (s_{\ell_n}(j_{n,y}))^2 = (c_{j_{n,y}}{}^2 \cdot 2^{p(n)-1})\, 2^{p(n)+1},$$

$$j_{n,y} \text{ is odd} \Rightarrow (s_{\ell_n}(j_{n,y}))^2 = (d_{j_{n,y}}{}^2 \cdot 2^{p(n)-1} - d_{j_{n,y}})\, 2^{p(n)+1} + 1.$$

Defining the integer-valued functions

$$\hat{c}(n, y) \stackrel{\mathrm{df}}{=} c_{j_{n,y}}{}^2 \cdot 2^{p(n)-1} \text{ and}$$

$$\hat{d}(n, y) \stackrel{\mathrm{df}}{=} d_{j_{n,y}}{}^2 \cdot 2^{p(n)-1} - d_{j_{n,y}},$$

we obtain

$$(s_{\ell_n}(j_{n,y}))^2 = \begin{cases} \hat{c}(n, y) \cdot 2^{p(n)+1} & \text{if } j_{n,y} \text{ is even} \\ \hat{d}(n, y) \cdot 2^{p(n)+1} + 1 & \text{if } j_{n,y} \text{ is odd;} \end{cases}$$

that is, the value of $(s_{\ell_n}(j_{n,y}))^2$ is a multiple of either $2^{p(n)+1}$ or $2^{p(n)+1}+1$, depending on the parity of $j_{n,y}$. Since $f(1^n) \leqslant 2^{p(n)}$ and since $j_{n,y}$ is odd if and only if $1^n \# y \in L$, the rightmost $p(n)+1$ bits of the binary representation of $g(1^n)$ represent the value of $f(1^n)$. Hence, after the value $g(1^n)$ has been returned by the oracle, T

[7] That $\#\mathrm{P}$ is closed under strong sum and product means the following: If $f \in \#\mathrm{P}$ and q is a polynomial, then the functions $\mathrm{sum}(x) \stackrel{\mathrm{df}}{=} \sum_{|y| \leqslant q(|x|)} f(\langle x, y \rangle)$ and $\mathrm{prod}(x) \stackrel{\mathrm{df}}{=} \prod_{0 \leqslant y \leqslant q(|x|)} f(\langle x, y \rangle)$ both are in $\#\mathrm{P}$. We refer to the work of Fenner *et al.* [FFK94] for a proof of this claim.

can output $f(1^n)$ by printing the $p(n)+1$ rightmost bits of $g(1^n)$. This completes the proof. ∎

Since $\#\mathrm{P}_1 \subseteq \mathrm{FP}$ implies $\mathrm{FP}^{\#\mathrm{P}_1^{\#\mathrm{P}_1}} \subseteq \mathrm{FP}$, we have from Theorem 3.7 the following corollary.

COROLLARY 3.8. *$\#\mathrm{P}_1 \subseteq \mathrm{FP}$ if and only if $\#\mathrm{P}_1^{\mathrm{PH}} \subseteq \mathrm{FP}$, and in particular, $\#\mathrm{P}_1 \subseteq \mathrm{FP}$ if and only if $\mathrm{span}\mathrm{P}_1 \subseteq \mathrm{FP}$.*

Corollary 3.8, together with the equivalences of Theorems 3.4 and 3.5, gives the following corollary.

COROLLARY 3.9. *Every* P *set has an easy census function if and only if every set in* PH *has an easy census function.*

Köbler *et al.* [KST89] proved that $\mathrm{span}\mathrm{P} = \#\mathrm{P}$ if and only if NP = UP. Their proof also establishes the analogous result for tally sets:

LEMMA 3.10 (Implicit in [KST89]). *Every tally* NP *set is in* UP *if and only if* $\mathrm{span}\mathrm{P}_1 = \#\mathrm{P}_1$.

Using Lemma 3.10, we now show that $\mathrm{span}\mathrm{P}_1$ and $\#\mathrm{P}_1$ are different classes unless NE = UE, or unless every sparse set in NP is low for SPP. A set S is said to be *$\mathscr{C}$-low* for some class $\mathscr{C}$ if $\mathscr{C}^S = \mathscr{C}$; see, e.g., [Sch83, KS85, Sch87, KSTT92] for a number of important lowness results. In particular, it is known that every sparse NP set is low for P^{NP} [KS85] and for PP [KSTT92], but it is not known whether all sparse NP sets are low for SPP. Torán's result that in some relativized world there exists some sparse NP set that is not contained in $\oplus\mathrm{P}$ [Tor88], and thus not in SPP, may be taken as evidence that not all sparse NP sets are SPP-low. Since Corollary 3.11 relativizes, $\mathrm{span}\mathrm{P}_1 \neq \#\mathrm{P}_1$ holds relative to the same oracle.

COROLLARY 3.11. *If $\mathrm{span}\mathrm{P}_1 = \#\mathrm{P}_1$, then two statements are true:*

1. NE = UE.
2. *Every sparse* NP *set is low for* SPP.

Proof. The first part follows from a standard upward translation argument (as mentioned in the proof of Theorem 3.4).

For the second part, assume $\mathrm{span}\mathrm{P}_1 = \#\mathrm{P}_1$, and let S be any sparse set in NP. By the result of Hartmanis, Immerman, and Sewelson [Har83, HIS85], S polynomial-time truth-table reduces to some tally NP set T. By Lemma 3.10, our assumption implies that $T \in \mathrm{UP}$, and, thus, $T \in \mathrm{SPP}$. Since $\mathrm{P}^{\mathrm{SPP}} = \mathrm{SPP}$, we have $S \in \mathrm{SPP}$. The result now follows from the self-lowness of SPP [FFK94], i.e., from the equality $\mathrm{SPP}^{\mathrm{SPP}} = \mathrm{SPP}$. ∎

4. ENUMERATIVE APPROXIMATION OF CENSUS FUNCTIONS

Cai and Hemaspaandra [CH89] introduced the notion of enumerative counting as a way of approximating the value of a #P function deterministically in polynomial time.

DEFINITION 4.1 [CH89]. Let $f: \Sigma^* \to \Sigma^*$ and $g: \mathbb{N} \to \mathbb{N}$ be two functions. A Turing transducer E is a $g(n)$-*enumerator* of f if for all $n \in \mathbb{N}$ and $x \in \Sigma^n$,

1. E on input x prints a list $\mathscr{L}_x$ with at most $g(n)$ elements, and
2. $f(x)$ is a member of list $\mathscr{L}_x$.

A function f is $g(n)$-*enumerable* in time $t(n)$ if there exists a $g(n)$-enumerator of f that runs in time $t(n)$. A set is $g(n)$-*enumeratively rankable* in time $t(n)$ if its ranking function is $g(n)$-enumerable in time $t(n)$.

Recall from the introduction Hemaspaandra and Rudich's [HR90] result that every P set is k-enumeratively rankable for some fixed k (and indeed, even $\mathcal{O}(n^{1/2-\varepsilon})$-enumeratively rankable for some $\varepsilon > 0$) in polynomial time if and only if $\#\mathrm{P} = \mathrm{FP}$. They conclude that it is no more likely that one can enumeratively rank all sets in P than that one can exactly compute their ranking functions in polynomial time. We similarly characterize the question of whether every P set has a census function that is n^α-enumerable in time n^β for fixed constants α and β. By the argument given in the proof of Theorem 3.4, this question is equivalent to asking whether every $\#\mathrm{P}_1$ function is n^α-enumerable in time n^β. We show that this assumption implies $\#\mathrm{P}_1 \subseteq \mathrm{FP}$, and we conclude that it is no more likely that one can n^α-enumerate the census function of every P set in time n^β than that one can precisely compute its census function in polynomial time. It would be interesting to know if this result can be improved to hold for polynomial time instead of time t for some fixed polynomial $t(n) = n^\beta$.

THEOREM 4.2. *Let $\alpha, \beta > 0$ be constants. If every $\#\mathrm{P}_1$ function is n^α-enumerable in time n^β, then $\#\mathrm{P}_1 \subseteq \mathrm{FP}$.*

Proof. Cai and Hemaspaandra [CH91] show that for any fixed k, if $\#\mathrm{SAT}$ (the function mapping any boolean formula f to the number of satisfying assignments of f) is n^k-enumerable, then $\#\mathrm{P} \subseteq \mathrm{FP}$. In order to prove this, they develop the following protocol for computing the permanent of an $m \times m$ matrix A,[8] given as parameters (the encoding of) a polynomial-time transducer E (the enumerator for $\#\mathrm{SAT}$), and a prime number p: Set $A_0 = A$ to the input matrix and repeat the following steps for $i = 1, ..., m-1$:

1. Construct from A_{i-1} an $(m-i) \times (m-i)$ matrix $B_i(X)$ over an indeterminate X, defined by

$$B_i(X) \stackrel{\mathrm{df}}{=} \sum_{k=1}^{m-i} e_k(X)\, a_{1k} A_{i-1}^{(1,k)},$$

where $e_k(X)$ is a degree $(m-i)$ polynomial in X such that $e_k(X) \equiv 1$ if $X = k$ and 0 otherwise, a_{1k} is the $(1, k)$ entry of A_{i-1}, and $A_{i-1}^{(1,k)}$ is the $(1, k)$-minor of A_{i-1}. Each matrix is viewed as a matrix over $\mathbb{Z}/p\mathbb{Z}[X]$; that is, the matrix entries

[8] Denoting the (i, j) entry of an $m \times m$ integer matrix A by a_{ij}, the *permanent* of A is defined by $\mathrm{perm}(A) \stackrel{\mathrm{df}}{=} \sum_\sigma \prod_{i=1}^m a_{i\sigma(i)}$, where the sum is over all permutations σ on $\{1, 2, ..., m\}$.

are polynomials in X whose integer coefficients are reduced modulo p. Then the following conditions hold.

- Each entry of $B_i(X)$ is a degree $(m-i)$ polynomial in X with coefficients in $\{0, ..., p-1\}$, so $\text{perm}(B_i(X))$ is a degree $(m-i)^2$ polynomial in X.
- $\sum_{k=1}^{m-i} \text{perm}(B_i(k)) = \text{perm}(A_{i-1})$.

2. Encode $B_i(X)$ into a binary string specifying in binary p, m, and the coefficients of $B_i(X)$. There is some fixed constant $c > 0$ such that the encoding length is at most $c(m-i)^3 \log p$. Define $Q_i(X) \stackrel{\text{df}}{=} \text{perm}(B_i(X))$. Then, Q_i is a polynomial of degree at most $(m-i)^2$, whose coefficients are each length-bounded by a fixed polynomial in p and m. Thus, there is a $\#\text{P}$ function G that maps $B_i(X)$ to a number from which the coefficients of Q_i can be decoded in polynomial time.

3. Use E as an enumerator for G to obtain candidates $g_1, ..., g_t$. These are all degree $(m-i)^2$ polynomials that are pairwise distinct. Since two distinct degree $(m-i)^2$ polynomials can agree at no more than $(m-i)^2 - 1$ points, there are fewer than $t^2(m-i)^2 \leqslant t^2m^2 - 1$ points X at which any two candidate polynomials agree. Thus, if $p \geqslant t^2m^2$, then there is an $r \in \{0, ..., p-1\}$ such that $g_j(r) \neq g_k(r)$ for all $j \neq k$. Take the smallest such r and set A_i to $B_i(r)$ with the entries reduced modulo p. Now, $\text{perm}(A_i)$ modulo p specifies which g_j is correct, so we can recover $\text{perm}(A_{i-1})$ modulo p in polynomial time.

At the end of this loop, A_m is a 1×1 matrix, so its permanent is easy to compute. Now working backwards again, we can recover $\text{perm}(A)$ modulo p. If we do this for polynomially (in the encoding length of A) many distinct primes, then by the Chinese Remainder Theorem, we can recover the exact value of $\text{perm}(A)$.

Valiant [Val79a] showed that the permanent of matrices whose entries are from the set $\{-1, 0, 1, 2\}$ is complete for $\#\text{P}$.[9] Analogously, we can show that there exists an infinite sequence of matrices $[M_1, M_2, ...]$ such that (i) the mapping $1^n \to \text{perm}(M_n)$ is complete for $\#\text{P}_1$, (ii) the mapping $1^n \to M_n$ is polynomial-time computable, and (iii) for every n, M_n is an $n \times n$ matrix whose entries are from $\{-1, 0, 1, 2\}$. Because of (iii), $\text{perm}(M_n) \leqslant 2^{2n}$ for all n. So, by the Chinese Remainder Theorem, for every n, the exact value of $\text{perm}(M_n)$ can be computed from $\text{perm}(M_n)$ modulo p for $2n$ arbitrary distinct primes p. Define polynomials q and s by $q(n) = \langle n, n, n, 2n \rangle$ and $s(n) = q(n)^{2\alpha} n^2$. Define the function f from the tally strings to the set of natural numbers as follows:

- If $m = \langle H, n, i, j \rangle$ for some $H, i \leqslant n$, and $j \leqslant 2n$, then $f(1^m)$ is $G(B_i(X))$, where the function G and the matrix $B_i(X)$ are defined as in the above protocol, except that now we simulate the protocol subject to the constraints:

  — The jth smallest prime $> s(n)$ is used in place of p.

  — M_n is used in place of the input matrix A_0.

  — H is viewed as (the encoding of) a Turing transducer and is used in place of the enumerator E. Here, for each k with $1 \leqslant k \leqslant i-1$, the input given to H in the

[9] That is, Valiant [Val79a] showed that $\text{perm} \in \#\text{P}$ and $\#\text{P} \subseteq \text{FP}^{\text{perm}}$, where perm is used as a function oracle.

kth round of the protocol is $\langle H, n, k, j\rangle$, not the matrix A_k. Also, H is supposed to run in $q(n)^\beta$ steps and to generate at most $q(n)^\alpha$ candidates in each round. If H does not halt in $q(n)^\beta$ steps or generates more than $q(n)^\alpha$ candidates at any point of the simulation, then the simulation is immediately aborted and the value $f(1^m)$ is set to 0.

- If m is not of the above form, $f(1^m)$ is 0.

This function f is in $\#\mathrm{P}_1$. First, there are only $i \leqslant m$ rounds to be simulated and each round requires m^α steps for candidate generation and some polynomial (in n) number of steps for other computations. Second, by the Prime Number Theorem, the first $2n$ smallest primes $>n$ are in $\mathcal{O}(n)$. (Remember that m is the *length* of the input and m is bounded by some polynomial in n; so, in time polynomial in n one can find these primes using simple methods such as the Sieve of Erastosthenes.) Since j and $s(n)$ are polynomially bounded, finding the jth smallest prime $>s(n)$ requires only a polynomial number of steps.

Now, by our assumption, there is an m^α-enumerator $\hat{E}$ for f that runs in time m^β. Since the number of candidates that $\hat{E}$ generates is at most m^α and the dimension of the matrix M_n is n, we have a prime $>m^{2\alpha}n^2$. This implies that with $\hat{E}$ as the enumerator, for every $n \geqslant \hat{E}$, every j, $1 \leqslant j \leqslant 2n$, and every i, $1 \leqslant i \leqslant n$, we successfully find an r for distinguishing the candidates. So, with $\hat{E}$ as the enumerator, for all $n \geqslant \hat{E}$, $\mathrm{perm}(M_n)$ is polynomial-time computable. Hence $\#\mathrm{P}_1 \subseteq \mathrm{FP}$. ▌

5. ORACLE RESULTS

In this section, we provide a number of relativized results on the existence or nonexistence of P sets simultaneously satisfying pairs of conditions chosen among the properties (i), (ii), and (iii) from Section 3. For instance, Theorem 5.1 and its Corollary 5.2 below exhibit a relativized world in which every P set has an easy census function (Property (ii)), yet there exists some set in P that is not rankable (Property (i)).

THEOREM 5.1. *There exists an oracle D such that $\#\mathrm{P}_1{}^D \subseteq \mathrm{FP}^D \neq \#\mathrm{P}^D$.*

From the relativized versions of Theorem 3.4 and of Hemaspaandra and Rudich's result in [HR90] that every P set is rankable if and only if $\mathrm{P}^{\#\mathrm{P}} = \mathrm{P}$ (which is equivalent to $\mathrm{FP} = \#\mathrm{P}$, and this equivalence itself also relativizes), we immediately obtain the following corollary.

COROLLARY 5.2. *There exists an oracle D such that all sets in P^D have a census function computable in FP^D, yet there exists some set in P^D that is not rankable by any function in FP^D.*

Proof of Theorem 5.1. Balcázar *et al.* [BBS86] and Long and Selman [LS86] proved that the polynomial hierarchy does not collapse if and only if it does not collapse relative to every sparse oracle. Since their proof relativizes (i.e., it applies to the relativized polynomial hierarchy as well), we have the following claim.

CLAIM 5.3 [BBS86, LS86]. *For every set B, PH^B does not collapse if and only if for every sparse oracle S, $(\mathrm{PH}^B)^S$ does not collapse.*

Note that $(\mathrm{PH}^B)^S = \mathrm{PH}^{B \oplus S}$, where $X \oplus Y \stackrel{\mathrm{df}}{=} \{0x \mid x \in X\} \cup \{1y \mid y \in Y\}$ denotes the join of any two sets X and Y. Fix an oracle A such that PH^A does not collapse (such oracles were constructed by Yao [Yao85], Håstad [Hås89], and Ko [Ko89] who built on the work of Furst *et al.* [FSS84]). Then, by Claim 5.3 above, for every sparse set S, $\mathrm{PH}^{A \oplus S}$ does not collapse. So, in particular, $\mathrm{P}^{A \oplus S} \neq \mathrm{NP}^{A \oplus S}$ for every sparse set S. Since for every oracle B, $\#\mathrm{P}^B = \mathrm{FP}^B$ implies $\mathrm{NP}^B = \mathrm{P}^B$, we have that $\#\mathrm{P}^{A \oplus S} \neq \mathrm{FP}^{A \oplus S}$ for every sparse set S.

So it remains to prove that there exists a sparse set T such that $\#\mathrm{P}_1{}^{A \oplus T} \subseteq \mathrm{FP}^{A \oplus T}$. Then, setting $D = A \oplus T$ completes the proof.

Assume that our pairing function $\langle \cdot, \cdot, \cdot \rangle$ is nondecreasing in each argument (when the other arguments are fixed), polynomial-time computable and invertible, and is one-to-one and onto. Let $N_1^{(\cdot)}, N_2^{(\cdot)}, \ldots$ be a standard enumeration of all tally NP oracle machines. For each $i \geqslant 1$, let p_i be the polynomial time bound of $N_i^{(\cdot)}$. Then, the function $f^{(\cdot)}$ defined by

$$f^{(\cdot)}(1^{\langle i, n, j \rangle}) \stackrel{\mathrm{df}}{=} \begin{cases} \mathrm{acc}_{N_i^{(\cdot)}}(1^n) & \text{if } p_i(n) < j \\ 0 & \text{otherwise} \end{cases}$$

is a canonical function complete for the class $\#\mathrm{P}_1{}^{(\cdot)}$.[10] In particular, for every fixed set S, $f^{(A \oplus S)}$ is complete for $\#\mathrm{P}_1{}^{A \oplus S}$.

The oracle set T is defined in such a way that, for any given $m = \langle i, n, j \rangle$ in unary, some polynomial-time oracle transducer can retrieve the value of $f^{(A \oplus T)}(1^m)$ from its oracle $A \oplus T$ by asking at most m queries. More formally, we construct T in stages such that for each $m = \langle i, n, j \rangle$:

$$1^k 0^{m-k} \# b \in T \Leftrightarrow 1 \leqslant k \leqslant |f^{(A \oplus T)}(1^m)| \text{ and the } k\text{th bit of } f^{(A \oplus T)}(1^m) \text{ is } b.$$

By the above definition, $|f^{(A \oplus T)}(1^m)| < m$ and coding information into the oracle is unnecessary when $N_i^{A \oplus T}(1^n)$ queries strings of length $\geqslant m$. So there is no interference between the stages of the construction of T. It is easy to see that T is a sparse set satisfying $\#\mathrm{P}_1{}^{A \oplus T} \subseteq \mathrm{FP}^{A \oplus T}$. ∎

Now we construct an oracle relative to which there exists some scalable set in P whose census function is not easy to compute.

THEOREM 5.4. *There exists an oracle A such that there exists an A-scalable set B whose census function is not in FP^A.*

Proof. We will construct A and B in such a way that B is P^A-isomorphic to the set $R \stackrel{\mathrm{df}}{=} \{0x \mid x \in \Sigma^*\}$, which is rankable in FP (and thus in FP^A). For each $n \geqslant 1$, we have $census_R(1^n) = 2^{n-1}$. So $census_R$ is easy to compute, but we want B to have a hard census function. In light of Proposition 3.1.2, we thus need the isomorphism, f, between B and R to be non-length-preserving. In particular, we will define f so as to satisfy $|f(x)| \leqslant |x| + 1$ and $|f^{-1}(y)| \leqslant |y|$ for all $x, y \in \Sigma^*$. When f is defined, we

[10] See [Val79b] for *natural* $\#\mathrm{P}_1$-complete functions.

let B be the set $f^{-1}(R)$. To have f and its inverse computable in FP^A, we encode f and f^{-1} into $A \stackrel{\mathrm{df}}{=} A_f \oplus A_{f^{-1}}$ as follows. For all $x \in \Sigma^*$, $i \geqslant 1$, and $b \in \{0, 1\}$, we ensure that

$$\langle x, i, b \rangle \in A_{f^*} \Leftrightarrow \text{the } i\text{th bit of } f^*(x) \text{ is } b, \tag{3}$$

where f^* stands for either f or f^{-1}. At the same time we diagonalize against FP^A so as to ensure $census_B \notin \mathrm{FP}^A$.

Let $T_1^{(\cdot)}, T_2^{(\cdot)}, \ldots$ be a standard enumeration of all deterministic polynomial-time oracle transducers, and let $p_1, p_2, \ldots$ be a sequence of strictly increasing polynomials such that p_i bounds the running time of T_i (independent of the oracle used). By (3) above, implicit in the definition of f and f^{-1} is the definition of A, so it suffices to construct the isomorphism. The construction of f and f^{-1} is in stages. By the end of stage i, f will have been defined for all strings of length up to $r(i)$, where r will be determined below. Initially, we start with $r(0) = 0$, and we define $f(\varepsilon) = \varepsilon$. Stage $i > 0$ of the construction is as follows.

Stage i. Choose n_i to be the smallest integer such that $n_i > r(i-1)$ and $p_i(n_i) < 2^{n_i - 2}$. Let A' be the subset of A that has been decided by now. We want to define f so that, eventually, $T_i^A(1^{n_i}) \neq census_B(1^{n_i})$. Simulate $T_i^{A'}$ on input 1^{n_i}. Whenever in this simulation a string of the form $0\langle x, i, b \rangle$ whose membership in A has not yet been decided is queried, we add this string to A' and set the ith bit of $f(x)$ to b unless we have already put $0\langle x, i, 1-b \rangle$ into A (and thus, have set this bit to $1-b$), or unless $i > |x| + 1$. The same comment applies to query strings $1\langle y, j, b \rangle$ whose membership in A has not been decided yet and which may fix the jth bit of $f^{-1}(y)$. If we added the queried string to A', we continue the simulation in the "yes" state; otherwise, in the "no" state. In this way, the simulation of $T_i^{A'}(1^{n_i})$ may determine f (and f^{-1}) on at most $p_i(n_i) < 2^{n_i - 2}$ bits of the strings of length n_i. Thus, for no $m \geqslant n_i$ is f^{-1} determined on all strings of length m in R or $\bar{R}$. Once the value $T_i^{A'}(1^{n_i})$ is computed, there is room to decide $f(x)$ and $f^{-1}(y)$ for all strings x and y of lengths between $r(i-1)$ and $p_i(n_i)$ so that f is an isomorphism mapping to $\bigcup_{\ell = r(i-1)}^{p_i(n_i)} R^{=\ell}$ and such that $census_B(1^{n_i}) \neq T_i^{A'}(1^{n_i})$, without changing the output value of $T_i^{A'}(1^{n_i})$. Finally, define $r(i) = p_i(n_i)$. ∎

Next, we provide an oracle relative to which there exists some set in P that is neither scalable nor has an easy census function.

THEOREM 5.5. *There exists an oracle D such that $D \in \mathrm{P}^D$ is not D-scalable and its census function is not in* FP^D.

Proof. It is known from the work of Goldsmith and Homer [GH96] that any sparse set is scalable if and only if it is rankable, and this holds if and only if it is P-printable.[11] D will be sparse, with at most two strings at each length. We assume that $(T_i^{(\cdot)})_{i \geqslant 1}$ enumerates $\mathrm{FP}^{(\cdot)}$, and that $T_i^{(\cdot)}$ runs in time n^i. A simple diagonalization guarantees that no P^D function computes the census of D. Note that this guarantees that no P^D function computes the rank of 1^n in D for all n, since $census_D(1^n) = \mathrm{rank}_D(1^n) - \mathrm{rank}_D(1^{n-1})$ would then be in P^D.

[11] A set is P-*printable* [HY84] if there exists a polynomial-time transducer T such that for each length n, T on input 1^n prints a list of all elements of the set up to length n.

At stage i we guarantee that $T_i^D(1^n)$ does not compute the census function of D, where n is chosen large enough that $n^i < 2^n$. Compute $T_i^D(1^n)$, restraining any oracle strings of length $\geqslant n$ that it queries. By our choice of n, this does not decide $D^{=m}$ for any $m \geqslant n$, so we can then put in the appropriate number of strings of length n for the diagonalization. ∎

Finally, we show that relative to an oracle, there exists some non-scalable set in P having an easy census function.

THEOREM 5.6. *There exists an oracle A such that $A \in \mathrm{P}^A$ is not A-scalable and its census function is in* FP^A.

Proof. We construct the oracle A so that A has one string of each length. For those lengths for which nothing else is decided, we put in 1^n. Otherwise, we do the following.

To make the oracle A non-A-scalable, we actually make it non-P^A-printable. At stage i, choose an appropriate length n and, then, compute $T_i^A(1^n)$. Whenever it queries a string of length $\geqslant n$, restrain the string from the oracle. If it does anything except print out $A^{\leqslant n}$, then put in the first unrestrained string of each length. If it correctly prints A up to length n, then choose an x of each relevant length to include that neither is restrained nor printed. ∎

We conclude this section with an open question: Can one construct an oracle E such that all sets in P^E have a census function computable in FP^E, but $E \in \mathrm{P}^E$ is not E-scalable? Note that scalability, rankability, and P-printability are equivalent properties on the sparse sets [GH96]; so, one way of solving the above question would be to prove the following stronger version of Theorem 5.1: There exists a *sparse* set E such that $\#\mathrm{P}_1{}^E \subseteq \mathrm{FP}^E \neq \#\mathrm{P}^E$. We note that a solution to this issue seems to require new techniqes, since the oracle $D = A \oplus T$ constructed in the proof of Theorem 5.1 inherently is a *non*sparse set due to its A part, and it cannot be made sparse unless one could separate the unrelativized polynomial hierarchy [LS86, BBS86].

ACKNOWLEDGMENTS

We are deeply indebted to Lance Fortnow, Lane Hemaspaandra, and Gabriel Istrate for interesting discussions and for helpful comments and suggestions, and we thank Eric Allender and Lane Hemaspaandra for pointers to the literature. We also thank Eric Allender and an anonymous referee for discovering an error in an earlier draft of this paper and for pointing out that the second part of Theorem 3.6 should not be interpreted as evidence against P having easy census functions, in light of the results of Impagliazzo and Wigderson [IW97], showing that *very* reasonable hypotheses imply P = BPP.

Received March 30, 1998; final manuscript received June 14, 1999

REFERENCES

[Adl78] Adleman, L. (1978), Two theorems on random polynomial time, *in* "Proceedings of the 19th IEEE Symposium on Foundations of Computer Science," pp. 75–83.

[All91] Allender, E. (1991), Limitations of the upward separation technique, *Math. Systems Theory* **24**(1), 53–67.

[AR88] Allender, E., and Rubinstein, R. (1988), P-printable sets, *SIAM J. Comput.* **17**(6), 1193–1202.

[BBS86] Balcázar, J., Book, R., and Schöning, U. (1986), The polynomial-time hierarchy and sparse oracles, *J. Assoc. Comput. Mach.* **33**(3), 603–617.

[BC93] Bovet, D., and Crescenzi, P. (1993), "Introduction to the Theory of Complexity," Prentice–Hall, Englewood Cliffs, NJ.

[BG92] Beigel, R., and Gill, J. (1992), Counting classes: Thresholds, parity, mods, and fewness, *Theoret. Comput. Sci.* **103**(1), 3–23.

[BH77] Berman, L., and Hartmanis, J. (1977), On isomorphisms and density of NP and other complete sets, *SIAM J. Comput.* **6**(2), 305–322.

[Boo74] Book, R. (1974), Tally languages and complexity classes, *Inform. and Control* **26**(2), 186–193.

[CH89] Cai, J., and Hemachandra, L. (1989), Enumerative counting is hard, *Inform. and Comput.* **82**(1), 34–44.

[CH90] Cai, J., and Hemachandra, L. (1990), On the power of parity polynomial time, *Math. Systems Theory* **23**(2), 95–106.

[CH91] Cai, J., and Hemachandra, L. (1991), A note on enumerative counting, *Inform. Process. Lett.* **38**(4), 215–219.

[FFK94] Fenner, S., Fortnow, L., and Kurtz, S. (1994), Gap-definable counting classes, *J. Comput. System Sci.* **48**(1), 116–148.

[FSS84] Furst, M., Saxe, J., and Sipser, M. (1984), Parity, circuits, and the polynomial-time hierarchy, *Math. Systems Theory* **17**(1), 13–27.

[GH96] Goldsmith, J., and Homer, S. (1996), Scalability and the isomorphism problem, *Inform. Process. Lett.* **57**(3), 137–143.

[Gil77] Gill, J. (1977), Computational complexity of probabilistic Turing machines, *SIAM J. Comput.* **6**(4), 675–695.

[GP86] Goldschlager, L., and Parberry, I. (1986), On the construction of parallel computers from various bases of boolean functions, *Theoret. Comput. Sci.* **43**(1), 43–58.

[GS91] Goldberg, A., and Sipser, M. (1991), Compression and ranking, *SIAM J. Comput.* **20**(3), 524–536.

[Har83] Hartmanis, J. (1983), On sparse sets in NP − P, *Inform. Process. Lett.* **16**(2), 55–60.

[Hås89] Håstad, J. (1989), Almost optimal lower bounds for small depth circuits, *in* "Randomness and Computation" (S. Micali, Ed.), Advances in Computing Research, Vol. 5, pp. 143–170, JAI Press, Greenwich, CT.

[Hem89] Hemachandra, L. (1989), The strong exponential hierarchy collapses, *J. Comput. System Sci.* **39**(3), 299–322.

[Her90] Hertrampf, U. (1990), Relations among MOD-classes, *Theoret. Comput. Sci.* **74**(3), 325–328.

[HHH99] Hemaspaandra, E., Hemaspaandra, L., and Hempel, H. (1999), A downward collapse within the polynomial hierarchy, *SIAM J. Comput.* **28**(2), 383–393.

[HIS85] Hartmanis, J., Immerman, N., and Sewelson, V. (1985), Sparse sets in NP − P: EXPTIME versus NEXPTIME, *Inform. and Control* **65**(2/3), 159–181.

[HJ95] Hemaspaandra, L., and Jha, S. (1995), Defying upward and downward separation, *Inform. and Comput.* **121**, 1–13.

[HJRW98] Hemaspaandra, L., Jiang, Z., Rothe, J., and Watanabe, O. (1998), Boolean operations, joins, and the extended low hierarchy, *Theoret. Comput. Sci.* **205**(1–2), 317–327.

[HR90] Hemachandra, L., and Rudich, S. (1990), On the complexity of ranking, *J. Comput. System Sci.* **41**(2), 251–271.

[HR97] Hemaspaandra, L., and Rothe, J. (1997), Unambiguous computation: Boolean hierarchies and sparse Turing-complete sets, *SIAM J. Comput.* **26**(3), 634–653.

[HRW97a] Hemaspaandra, L., Rothe, J., and Wechsung, G. (1997), Easy sets and hard certificate schemes, *Acta Inform.* **34**(11), 859–879.

[HRW97b] Hemaspaandra, L., Rothe, J., and Wechsung, G. (1997), On sets with easy certificates and the existence of one-way permutations, *in* "Proceedings of the Third Italian Conference on Algorithms and Complexity," pp. 264–275, Lecture Notes in Computer Science, Vol. 1203, Springer-Verlag, New York/Berlin.

[HU79] Hopcroft, J., and Ullman, J. (1979), "Introduction to Automata Theory, Languages, and Computation," Addison–Wesley, Reading, MA.

[HY84] Hartmanis, J., and Yesha, Y. (1984), Computation times of NP sets of different densities, *Theoret. Comput. Sci.* **34**(1/2), 17–32.

[IW97] Impagliazzo, R., and Wigderson, A. (1997), P = BPP unless E has sub-exponential circuits: Derandomizing the XOR lemma, *in* "Proceedings of the 29th ACM Symposium on Theory of Computing," pp. 220–229, ACM Press, New York.

[KL80] Karp, R., and Lipton, R. (1980), Some connections between nonuniform and uniform complexity classes, *in* "Proceedings of the 12th ACM Symposium on Theory of Computing," pp. 302–309. [An extended version is: Turing machines that take advice, *Enseign. Math. Series 2* **28**, 1982, 191–209]

[Ko89] Ko, K. (1989), Relativized polynomial time hierarchies having exactly k levels, *SIAM J. Comput.* **18**(2), 392–408.

[KS85] Ko, K., and Schöning, U. (1985), On circuit-size complexity and the low hierarchy in NP, *SIAM J. Comput.* **14**(1), 41–51.

[KST89] Köbler, J., Schöning, U., and Torán, J. (1989), On counting and approximation, *Acta Inform.* **26**(4), 363–379.

[KSTT92] Köbler, J., Schöning, U., Toda, S., and Torán, J. (1992), Turing machines with few accepting computations and low sets for PP, *J. Comput. System Sci.* **44**(2), 272–286.

[Lau83] Lautemann, C. (1983), BPP and the polynomial hierarchy, *Inform. Process. Lett.* **17**(4), 215–217.

[Lon85] Long, T. (1985), On restricting the size of oracles compared with restricting access to oracles, *SIAM J. Comput.* **14**(3), 585–597. [Erratum, *SIAM J. Comput.* **17**(3), 628]

[LS86] Long, T., and Selman, A. (1986), Relativizing complexity classes with sparse oracles, *J. Assoc. Comput. Mach.* **33**(3), 618–627.

[Mah82] Mahaney, S. (1982), Sparse complete sets for NP: Solution of a conjecture of Berman and Hartmanis, *J. Comput. System Sci.* **25**(2), 130–143.

[MS72] Meyer, A., and Stockmeyer, L. (1972), The equivalence problem for regular expressions with squaring requires exponential space, *in* "Proceedings of the 13th IEEE Symposium on Switching and Automata Theory, pp. 125–129.

[OH93] Ogiwara, M., and Hemachandra, L. (1993), A complexity theory for feasible closure properties, *J. Comput. System Sci.* **46**(3), 295–325.

[Pap94] Papadimitriou, C. (1994), "Computational Complexity," Addison–Wesley, Reading, MA.

[PZ83] Papadimitriou, C., and Zachos, S. (1983), Two remarks on the power of counting, *in* "Proceedings of the 6th GI Conference on Theoretical Computer Science," pp. 269–276, Lecture Notes in Computer Science, Vol. 145, Springer-Verlag, New York/Berlin.

[RRW94] Rao, R., Rothe, J., and Watanabe, O. (1994), Upward separation for FewP and related classes, *Inform. Process. Lett.* **52**, 175–180.

[Sch83] Schöning, U. (1983), A low and a high hierarchy within NP, *J. Comput. System Sci.* **27**, 14–28.

[Sch87] Schöning, U. (1987), Graph isomorphism is in the low hierarchy, *J. Comput. System Sci.* **37**, 312–323.

[Sip83] Sipser, M. (1983), A complexity theoretic approach to randomness, *in* "Proceedings of the 15th ACM Symposium on Theory of Computing," pp. 330–335.

[Sto77] Stockmeyer, L. (1977), The polynomial-time hierarchy, *Theoret. Comput. Sci.* **3**(1), 1–22.

[TO92] Toda, S., and Ogiwara, M. (1992), Counting classes are at least as hard as the polynomial-time hierarchy, *SIAM J. Comput.* **21**(2), 316–328.

[Tod91] Toda, S. (1991), PP is as hard as the polynomial-time hierarchy, *SIAM J. Comput.* **20**(5), 865–877.

[Tor88] Torán, J. (1988), "Structural Properties of the Counting Hierarchies," Ph.D. thesis, Universitat Politècnica de Catalunya, Barcelona, Spain.

[Val76] Valiant, L. (1976), The relative complexity of checking and evaluating, *Inform. Process. Lett.* **5**(1), 20–23.

[Val79a] Valiant, L. (1979), The complexity of computing the permanent, *Theoret. Comput. Sci.* **8**(2), 189–201.

[Val79b] Valiant, L. (1979), The complexity of enumeration and reliability problems, *SIAM J. Comput.* **8**(3), 410–421.

[Wel93] Welsh, D. (1993), "Complexity: Knots, Colourings and Counting," Cambridge Univ. Press, Cambridge.

[Yao85] Yao, A. (1985), Separating the polynomial-time hierarchy by oracles, *in* "Proceedings of the 26th IEEE Symposium on Foundations of Computer Science," pp. 1–10, IEEE Comput. Soc., Los Alamitos, CA.