WPI-CS-TR-92-5 August 1992

Relativizations of the P-Printable Sets and the Sets with Small Generalized Kolmogorov Complexity (Revised)

by

Roy S. Rubinstein

Relativizations of the P-Printable Sets and the Sets with Small Generalized Kolmogorov Complexity (Revised)

Roy S. Rubinstein

Worcester Polytechnic Institute[1]

August 1992

Abstract

Although tally sets are generally considered to be weak when used as oracles, it is shown here that in relativizing certain complexity classes, they are in fact no less powerful than any other class of sets and are more powerful than the class of recursive sets.

More specifically, relativizations of the classes of P-printable sets and sets with small Generalized Kolmogorov complexity (SGK) are studied. It is shown here that all sparse sets are $\mathrm{P}^{\mathrm{TALLY}}$-printable and are in $\mathrm{SGK}^{\mathrm{TALLY}}$, and that there are self-P-printable sets that are neither $\mathrm{P}^{\mathrm{REC}}$-printable nor in $\mathrm{SGK}^{\mathrm{REC}}$. There are also sets that are $\mathrm{P}^{\mathrm{REC}}$-printable and in $\mathrm{SGK}^{\mathrm{REC}}$ that are not self-P-printable. Relativizations to various subrecursive oracles are also presented.

A restriction on the number of oracle queries is also presented, with the result that relativizing SGK to *any* oracle with at most $O(\log n)$ queries results in a set that is still in SGK.

1 Introduction

Tally sets have generally been considered weak, often no more useful than the empty set, when used as oracles ([LS86, BBS86]), though in certain relativized worlds this is not the case ([HR92]). It is shown here that in relativizing certain structural complexity classes, the class of tally sets is in fact no less powerful than any other class of sets and is

[1]Some of these results were obtained while the author was at Northeastern University and was supported by National Security Agency grant MDA904-87-H-2020.

more powerful than the class of recursive sets. Specifically, relativizations of the classes of P-printable sets and sets with small Generalized Kolmogorov complexity (SGK) are studied. Definitions of these and other classes are presented in Section 2.

Although much work has been done studying relativizations of the computational complexity classes, including P and NP (starting with [BGS75]), relativizations of the *structural* complexity classes (see [Rub88]) have been neglected. This paper begins the study of relativizations of the structural complexity classes.

Any set that is $\mathrm{P}^{\mathcal{C}}$-printable or in $\mathrm{SGK}^{\mathcal{C}}$, for any class $\mathcal{C}$, must be sparse. How powerful must a class $\mathcal{C}$ be so that every sparse set is $\mathrm{P}^{\mathcal{C}}$-printable or in $\mathrm{SGK}^{\mathcal{C}}$? It is shown here that there exist sparse, even tally sets, that are not $\mathrm{P}^{\mathrm{REC}}$-printable, where REC is the class of recursive sets, and there are sparse sets (though clearly no tally sets) that are not in $\mathrm{SGK}^{\mathrm{REC}}$. It is also shown that there exist self-P-printable sets that are neither $\mathrm{P}^{\mathrm{REC}}$-printable nor in $\mathrm{SGK}^{\mathrm{REC}}$. Additionally, the existence of sets that are $\mathrm{P}^{\mathrm{REC}}$-printable and in $\mathrm{SGK}^{\mathrm{REC}}$ that are not self-P-printable is shown.

When structural complexity classes are used as oracles, however, not much power is needed to capture all the sparse sets. In particular, it is shown that every sparse set is in $\mathrm{SGK}^{\mathrm{TALLY}}$ and is $\mathrm{P}^{\mathrm{TALLY}}$-printable. This improves a result in [Sch86] that every sparse set is in $\mathrm{P}^{\mathrm{TALLY}}$ (as a corollary of his result that $\mathrm{P}^{\mathrm{TALLY}} = \mathrm{P}/poly$).

If the number of oracle calls is restricted, however, the sparse sets may not all be captured. Specifically, it is shown that for every oracle A and every constant k, $\mathrm{SGK}^{A[k \log n]} = \mathrm{SGK}$.

2 Preliminaries

It is assumed here that the reader has basic familiarity with the standard notions and classes in complexity theory.

We use the standard lexicographic ordering $\leq$ on strings, and $|w|$ denotes the length of the string w. All strings here are elements of $\{0,1\}^*$, and all sets are subsets of $\{0,1\}^*$. A tally language is a subset of $\{1\}^*$. The cardinality of a set A is denoted $\|A\|$. $A \subset B$ denotes $A \subseteq B$ and $A \neq B$. Whenever a number is used as a string, it is actually the binary representation of the number that is being used.

PSV is the class of functions computable deterministically in polynomial time. For any set A, PSV^A is the class of functions computable deterministically in polynomial time using A as an oracle. Similarly, NPSV is the class of single-valued functions computable nondeterministically in polynomial time, and NPSV^A is the relativization to A. EXPTIME and NEXPTIME are, respectively, $\bigcup_{c\geq 0} DTIME(2^{cn})$ and $\bigcup_{c\geq 0} NTIME(2^{cn})$. 2^{poly} is $\bigcup_{c\geq 0} DTIME(2^{n^c})$.

Standard polynomial time pairing functions are used, and the pairing of strings x and y, for example, is denoted $\langle x, y\rangle$. The pairing functions have the property that $|\langle x, y\rangle| \leq 2(|x|+|y|)$, that $\langle x, y\rangle$ can be determined from x and y in polynomial time, and that x and y can each be determined from $\langle x, y\rangle$ in polynomial time. Pairing functions may be applied to tally strings to produce another tally string by defining $\langle t_1, t_2\rangle$ to be $1^{\langle |t_1|,|t_2|\rangle}$, where t_1 and t_2 are tally strings. In this case $|\langle t_1, t_2\rangle| \leq 4(|t_1| + |t_2|)$. This same notation, when used for grouping more than two strings, actually denotes successive applications of the pairing function.

A ranking function determines the position of a string in a set. More precisely, for any set A and any string x, $r_A(x) = \|\{w \in A \mid w \leq x\}\|$. While this definition applies to strings not in the set as well as those in the set, this paper only uses ranking on strings in the set.

Definition 2.1 *A set A is* sparse *if there exists a polynomial p such that the number of strings in A of length less than or equal to n is less than or equal to $p(n)$.*

The *Kolmogorov complexity* of a finite binary string is the length of the shortest program that generates it ([Sol64, Kol65, Cha66]). A string is *Kolmogorov-random* if there is no program shorter than the string itself that can generate it. Since for every length n there are 2^n strings of length n and only $2^n - 1$ shorter programs, there are Kolmogorov-random strings of every length.

Generalized Kolmogorov complexity, a two-parameter version of Kolmogorov complexity that includes information about not only how far a string can be compressed, but how fast it can be restored, was introduced by Hartmanis ([Har83]), whose definition is presented here.

Definition 2.2 *For a (deterministic) Turing machine M and functions g and G mapping natural numbers to natural numbers, let*

$$K_M[g(n), G(n)] = \{x \mid (\exists y)[|y| \leq g(|x|) \text{ and } M(y) = x \text{ in } G(|x|) \text{ or fewer steps}]\}.$$

It was shown in [Har83] that there exists a universal Turing machine M_u such that for any other Turing machine M there exists a constant c such that $K_M[g(n), G(n)] \subseteq K_{M_u}[g(n) + c, cG(n) \log G(n) + c]$. Dropping the subscript, $K[g(n), G(n)]$ will actually denote $K_{M_u}[g(n), G(n)]$. This relativizes in a straightforward manner, where

$$K^A[g(n), G(n)] = \{x \mid (\exists y)[|y| \leq g(|x|) \text{ and } M_u^A(y) = x \text{ in } G(|x|) \text{ or fewer steps}]\}.$$

Definition 2.3 *A set is said to have* small generalized Kolmogorov complexity *if it is a subset of $K[c \log n, n^c]$ for some c.* SGK *denotes the class of sets with small generalized Kolmogorov complexity.*[2]

For any set A, SGK^A is the class of sets with small generalized Kolmogorov complexity relative to A. For any class of sets $\mathcal{C}$, SGK$^{\mathcal{C}}$ is the union over all A in $\mathcal{C}$ of SGK^A.

[2][BB86] refers to this class as **K[log, poly]**.

Definition 2.4 *A set S is* polynomial-time printable (P-printable) *if there exists a polynomial p such that all the elements of S of length less than or equal to n can be printed by a deterministic machine in time $p(n)$.*

P-printability relativizes (as in a set being P^A-printable) by allowing the printing machine to use an oracle. A set is *self*-P*-printable* if it is P-printable relative to itself. P-printability was introduced in [HY84] and was further explored in [HIS85], [AR88], and [Rub91].

Definition 2.5 *For any set A, $enum_A$ is the function that, for each n, on input 0^n produces a straightforward encoding of the set of strings in A of length less than or equal to n.*

Note that $enum_A \in \mathrm{PSV}$ is equivalent to A being P-printable, and that $enum_A \in \mathrm{PSV}^A$ is equivalent to A being self-P-printable. Other printabilities are defined in a like manner, so A is NP-printable if and only if $enum_A \in \mathrm{NPSV}$.

Allender ([AR88]) defined the complexity class FewP, a subclass of NP as follows:

Definition 2.6 *A language is in the class* FewP *if is accepted by a nondeterministic polynomial time Turing machine M for which there is a polynomial p such that for all inputs w, there are fewer than $p(w)$ accepting computations of M on w.*

This class relativizes (such as FewP^A) in a straightforward manner.

In [Lon85] and [LS86] it is shown that for every sparse set S, $enum_S \in \mathrm{PSV}^{prefix(S) \oplus S}$, where $prefix(S) = \{\langle y, 0^n \rangle \mid \exists z[yz \in S \text{ and } |yz| \leq n]\}$. ($\oplus$ denotes disjoint union, such as $A \oplus B = \{0x \mid x \in A\} \cup \{1y \mid y \in B\}$.) With the observation that $x \in S \iff \langle x, 0^{|x|} \rangle \in prefix(S)$, this is strengthened to $enum_S \in \mathrm{PSV}^{prefix(S)}$, or equivalently, the following proposition:

Proposition 2.1 *Every sparse set S is $\mathrm{P}^{prefix(S)}$-printable.*

A relativization with a numeric expression enclosed in brackets after the oracle denotes a bound on the number of queries to the oracle. For example, $\mathrm{P}^{A[k]}$ represents the class of sets that are recognizable by a deterministic polynomial time bounded oracle Turing machine making at most k queries to the set A.

For the remainder of this paper, unless otherwise stated, S will denote a sparse set and T will denote a tally set. TALLY is the class of tally sets, SPARSE is the class of sparse sets, and REC is the class of recursive sets.

3 Relativizations with Structural Complexity Classes

The following theorem shows that when using structural complexity class oracles, the class of sets with small generalized Kolmogorov complexity needs very little power in an oracle to encompass the same class of sets as a more powerful oracle.

Theorem 3.1 $\mathrm{SGK}^{\mathrm{TALLY}} = \mathrm{SGK}^{\mathrm{SGK}} = \mathrm{SGK}^{\mathrm{SPARSE}} = \mathrm{SGK}^{\mathrm{P}/poly}$.

Proof The inclusions from left to right are immediate, as the oracle classes are (properly) included from left to right. All that is needed to complete the proof is to show that $\mathrm{SGK}^{\mathrm{P}/poly} \subseteq \mathrm{SGK}^{\mathrm{TALLY}}$. Let B be a set in SGK^{A} for some set $A \in \mathrm{P}/poly$. Because every set in $\mathrm{P}/poly$ is polynomial time Turing reducible to some tally set ([Sch86]), let T be a tally set such that $A \leq_T^{\mathrm{P}} T$. Any query to A by a polynomial time machine can then be replaced by one or more queries to T using the reduction. Thus $B \in \mathrm{SGK}^{T}$. $\square$

Corollary 3.2 $\mathrm{SGK}^{\mathrm{SGK}^{\mathrm{SGK}}} = \mathrm{SGK}^{\mathrm{SGK}}$. *(There is no infinite* SGK *hierarchy.)*

Proof The inclusion of $\mathrm{SGK}^{\mathrm{SGK}}$ in $\mathrm{SGK}^{\mathrm{SGK}^{\mathrm{SGK}}}$ is immediate. For the other direction, because $\mathrm{SGK}^{\mathrm{SGK}}$ is sparse, $\mathrm{SGK}^{\mathrm{SGK}^{\mathrm{SGK}}} \subseteq \mathrm{SGK}^{\mathrm{SPARSE}} = \mathrm{SGK}^{\mathrm{SGK}}$. $\square$

Theorem 3.1 will now be strengthened to show that the class $\mathrm{SGK}^{\mathrm{TALLY}}$ is equal to the class of all sparse sets. One other result is needed first.

It was shown in [HH88, BB86, Rub86, AR88] that a set is P-printable if and only if it has small generalized Kolmogorov complexity and is in P. This result and its proof relativize to give the following proposition.

Proposition 3.3 *For all sets A and S, S is P^A-printable if and only if $S \in \mathrm{SGK}^A$ and $S \in \mathrm{P}^A$.*

The notation in this proposition generalizes to using a class of sets for the oracle, meaning that some set in that class is used.

We can now strengthen Theorem 3.1 to all sparse sets.

Theorem 3.4 $\mathrm{SGK}^{\mathrm{TALLY}} = \mathrm{SPARSE}$.

Proof The left to right inclusion is immediate. For the right to left inclusion, let S be a sparse set. By Proposition 2.1, S is $\mathrm{P}^{\mathit{prefix}(S)}$-printable. Because S is sparse, $\mathit{prefix}(S)$ is sparse, so S is $\mathrm{P}^{\mathrm{SPARSE}}$-printable. By Proposition 3.3, S is then in $\mathrm{SGK}^{\mathrm{SPARSE}} = \mathrm{SGK}^{\mathrm{TALLY}}$. $\square$

Corollary 3.5 SGK $\subset$ *the class of self*-P-*printable sets* $\subset \mathrm{SGK}^{\mathrm{TALLY}}$.

Proof These proper inclusions are already known and may be found in [BB86, Rub91]. $\square$

While every sparse set is $\mathrm{P}^{\mathrm{SPARSE}}$-printable, not every sparse set is self-P-printable. A sparse set may need a more powerful sparse set to enumerate itself in polynomial time. As the following theorem shows, all that is needed is a tally set.

Theorem 3.6 *Every sparse set is* $\mathrm{P}^{\mathrm{TALLY}}$*-printable.*

Proof Let S be a sparse set. By Theorem 3.4, S is in SGK^{T_1} for some tally set T_1. S is also in P^{T_2} for some tally set T_2 ([Sch86]). There is therefore a tally set $T = T_1 \oplus T_2$ such that $S \in \mathrm{SGK}^{T}$ and $S \in P^{T}$. By Proposition 3.3, S is P^{T}-printable. (Note that $T = T_1 \oplus T_2$ is a tally set by defining T to be $(\bigcup_{x \in T_1}\{\langle 1, x\rangle\}) \cup (\bigcup_{x \in T_2}\{\langle 11, x\rangle\}$.) $\square$

So every sparse set is P-printable relative to some tally set. If we restrict the sparse set, how much can we then restrict the tally set? For example could we obtain a result such as "Every sparse set in P is P-printable relative to some tally set in FewP"? If we had such a result, by noting that the existence of a sparse set in FewP $-$ P implies the existence of a sparse set in P that is not P-printable ([AR88]), we could easily prove "There exists a sparse set in FewP $-$ P if and only if there exists a tally set in FewP $-$ P." Unfortunately, the techniques usually used for such results (see [HIS85] and [All91]) do not appear to work in this setting, and the veracity of these statements is unknown.

The techniques of [HIS85] can, however, be extended to yield the following result.

Theorem 3.7 *Every sparse set in* FewP *is* P*-printable relative to some tally set in* NP.

Proof Let S be a sparse set in FewP witnessed by machine M, and let p be a polynomial such that for all n, the number of strings in S of length less than or equal to n is at most

$p(n)$. Define the tally set

$$T = \{1^{\langle n,i,j,k,d\rangle} \mid \exists x_1 < x_2 < ... < x_i \leq x < y_1 < y_2 < ... < y_j \in S,$$
$$|x_1| = |y_j| = n \text{ and the } k^{th} \text{ digit of } x \text{ is } d\}.$$

It will be shown that $T \in$ NP and S is P^T-printable.

The following algorithm shows that $T \in$ NP.

```
input $1^{\langle n,i,j,k,d\rangle}$ – reject if wrong format
if $i + j > p(n)$ then reject
nondeterministically guess $x_1, x_2, ..., x_i, x, y_1, y_2, ..., y_j$ and
        an accepting computation for machine $M$ for each
check that each of the above computations is correct,
        that $x_1 < x_2 < ... < x_i \leq x < y_1 < y_2 < ... < y_j$,
        that $|x_1| = |y_j| = n$, and that the $k^{th}$ digit of $x$ is $d$
accept if and only if all of these checks succeed
```

An accepting computation would require guessing $i+j+1$ strings each of which would be of length n, each with polynomial length accepting computations. As i and j are each less than or equal to $p(n)$ and the length of the input is $\Theta(n)$, this algorithm shows that $T \in$ NP.

Next it needs to be shown that S is P^T-printable. The following algorithm demonstrates this.

```
input $0^n$ – want to print strings in $S$ of length at most $n$
for each length $\ell$ from 0 to $n$
        find $c_\ell = \|S^{=\ell}\|$   (the number of strings in $S$ of length $\ell$) – see below
        for each $z$ between 1 and $c_\ell$   (print the $z^{th}$ string of length $\ell$ in $S$)
                $w := \lambda$
                for each position $k := 1$ to $\ell$
        if $1^{\langle \ell, z, c_\ell - z, k, 0\rangle} \in T$
                $w := w0$
        else
                $w := w1$
print $w$
```

To find c_ℓ, the number of strings in S of length ℓ, the following $p(n)$ pairs of strings are queried to T:

$$\begin{array}{rcl} 1^{\langle \ell,1,0,1,0\rangle} & \text{and} & 1^{\langle \ell,1,0,1,1\rangle} \\ 1^{\langle \ell,2,0,1,0\rangle} & \text{and} & 1^{\langle \ell,2,0,1,1\rangle} \\ 1^{\langle \ell,3,0,1,0\rangle} & \text{and} & 1^{\langle \ell,3,0,1,1\rangle} \\ & \ldots & \\ 1^{\langle \ell,p(n),0,1,0\rangle} & \text{and} & 1^{\langle \ell,p(n),0,1,1\rangle} \end{array}$$

c_ℓ will be equal to $i-1$, where i is the least integer such that both $1^{\langle \ell,i,0,1,0\rangle} \notin T$ and $1^{\langle \ell,i,0,1,1\rangle} \notin T$, signifying that there is no i^{th} string of length ℓ in S.

By checking $1^{\langle \ell,z,c_\ell - z,k,0\rangle} \in T$, the x whose k^{th} bit is checked must be the z^{th} string of length ℓ in S as follows. Having z as the second element in the quintuple ensures that there are at least z strings of length ℓ in S that are less than or equal to x, and having $c_\ell - z$ as the third element guarantees that there are at least $c_\ell - z$ strings of length ℓ in S that are greater than x. But there are exactly c_ℓ strings of length ℓ in S, so x must be z^{th} such string. □

4 Relativizations with Computational Complexity Classes

In the previous section we have seen that it takes only a simple *structural* complexity class, the class of tally sets, to be used as oracles to the class of P-printable sets and to SGK in order to capture all the sparse sets. What *computational* complexity class oracles are needed to these classes in order to capture the sparse sets? Also, what classes of sets are captured with other oracles?

Let us begin with relativizations of the P-printable sets. Proposition 2.1, that every sparse set S is $\mathrm{P}^{\mathit{prefix}(S)}$-printable, yields a number of interesting results. First, it must be observed that for any sparse set S, $\mathit{prefix}(S)$ is sparse and in FewP^S. With this we have the following, which may be considered corollaries of Proposition 2.1.

Corollary 4.1 *The following are equivalent:*

1. *S is a sparse set.*
2. *S is P-printable relative to some sparse set in FewP^S.*
3. *S is P-printable relative to some sparse set in NP^S.*
4. *S is $\Delta_2^{\mathrm{P},S}$-printable.*

Corollary 4.2 (1) $\Rightarrow$ (2) $\Rightarrow$ (3) $\Rightarrow$ (4):

1. *S is a sparse set in* P.
2. *S is P-printable relative to some sparse set in* FewP.
3. *S is P-printable relative to some sparse set in* NP.

4. *S is Δ_2^{P}-printable.*

Corollary 4.3 (1) $\Rightarrow$ (2) $\Rightarrow$ (3):

1. *S is a sparse set in Σ_k^{P}.*
2. *S is* P*-printable relative to some sparse set in $\Sigma_{k+1}^{\mathrm{P}}$.*
3. *S is $\Delta_{k+2}^{\mathrm{P}}$-printable.*

The following theorem is essentially a relativization of a result in [AR88]. The proof is slightly different, but the original proof relativizes as well.

Theorem 4.4 *For every class $\mathcal{C}$, there are no sparse sets in* $\mathrm{FewP}^{\mathcal{C}} - \mathrm{P}^{\mathcal{C}}$ *if and only if every sparse set in* $\mathrm{P}^{\mathcal{C}}$ *is* $\mathrm{P}^{\mathcal{C}}$*-printable.*

Proof For the left to right direction, assume there are no sparse sets in $\mathrm{FewP}^{\mathcal{C}} - \mathrm{P}^{\mathcal{C}}$, and let S be a sparse set in $\mathrm{P}^{\mathcal{C}}$. Then *prefix*(S) is a sparse set in $\mathrm{FewP}^{\mathrm{P}^{\mathcal{C}}}$, and because a polynomial time machine's using an oracle in $\mathrm{P}^{\mathcal{C}}$ is no more powerful than using an oracle in $\mathcal{C}$, *prefix*(S) is a sparse set in $\mathrm{FewP}^{\mathcal{C}}$. By assumption, *prefix*(S) is then in $\mathrm{P}^{\mathcal{C}}$. Because S is $\mathrm{P}^{\mathit{prefix}(S)}$-printable, S is $\mathrm{P}^{\mathrm{P}^{\mathcal{C}}}$-printable, and therefore is $\mathrm{P}^{\mathcal{C}}$-printable.

For the right to left direction, assume that every sparse set in $\mathrm{P}^{\mathcal{C}}$ is $\mathrm{P}^{\mathcal{C}}$-printable, and let S be a sparse set in $\mathrm{FewP}^{\mathcal{C}}$ via a nondeterministic polynomial time oracle machine M with oracle $A \in \mathcal{C}$. The set of accepting computations of all strings in S by machine M using oracle A, each of which contains the string in S being accepted, is a sparse set in P^{A}. By our assumption, this implies that this set of computations is $\mathrm{P}^{\mathcal{C}}$-printable, and therefore S is also $\mathrm{P}^{\mathcal{C}}$-printable and hence is in $\mathrm{P}^{\mathcal{C}}$.

□

Corollary 4.5 *The following are equivalent:*

1. *S is a sparse set in* PH.
2. *S is* P^{PH}*-printable.*
3. *S is* PH*-printable.*

Corollary 4.6 *The following are equivalent:*

1. *S is a sparse set in* PSPACE.
2. *S is* $\mathrm{P}^{\mathrm{PSPACE}}$*-printable.*
3. *S is* PSPACE*-printable.*

Similar results are easily obtained for nonsparse classes as well, but by a different technique.

Proposition 4.7 *Every set in 2^{poly} is 2^{poly}-printable.*

Proof To print all the strings of length less than or equal to n, simply run through each of the $2^{n+1} - 1$ possible strings, check each for membership in the set (time 2^{poly} each), and print if appropriate. □

This technique also yields the following result.

Proposition 4.8 *Every set in* EXPTIME *is* EXPTIME*-printable.*

These results do not, however, imply P-printability relative to an appropriate oracle. P-printability, regardless of the oracle used, implies sparseness. What about restrictions to the sparse sets? Is every sparse set in EXPTIME $\mathrm{P}^{\mathrm{EXPTIME}}$-printable? The method of testing each possible string does not work, as that still requires exponential time. Another technique does give us this result.

Theorem 4.9 *Every sparse set in* EXPTIME *is* $\mathrm{P}^{\mathrm{EXPTIME}}$*-printable.*

Proof Let S be a sparse set in EXPTIME witnessed by a machine with maximum running time 2^{cn}. By Proposition 2.1, S is $\mathrm{P}^{\mathit{prefix}(S)}$-printable, so all that needs to be shown is that $\mathit{prefix}(S) \in$ EXPTIME. The following algorithm determines membership in $\mathit{prefix}(x)$.

```
input $1^{\langle y,0^n \rangle}$ – reject if wrong format
for each $z$ of length 0 to $n - |y|$
        if $yz \in S$ halt and accept
halt and reject
```

The maximum number of z (if $|y| = 0$) is $2^{n+1} - 1$, and the maximum time to check $yz \in S$ (when $|yz| = n$) is 2^{cn}, so the maximum time for the loop is bounded by $(2^{n+1} - 1) * 2^{cn} < 2^{cn+(n+1)} = 2^{(c+1)n+1} < 2^{(c+2)n}$. So $\mathit{prefix}(S) \in$ EXPTIME and S is $\mathrm{P}^{\mathrm{EXPTIME}}$-printable. □

Is every sparse set in $\mathrm{P}^{\mathrm{EXPTIME}}$ $\mathrm{P}^{\mathrm{EXPTIME}}$-printable? The above method would require showing that for a sparse set $S \in \mathrm{P}^{\mathrm{EXPTIME}}$, $\mathit{prefix}(S) \in$ EXPTIME, but the above algorithm may run too long. There appears to be no obvious solution to this. We do have a similar result for NEXPTIME, though by a different technique.

Theorem 4.10 *Every sparse set in* $\mathrm{P}^{\mathrm{NEXPTIME}}$ *is* $\mathrm{P}^{\mathrm{NEXPTIME}}$*-printable.*

Proof It was shown by Hemachandra [Hem87] that the strong exponential hierarchy collapses to the $\mathrm{P}^{\mathrm{NEXPTIME}}$ level, i.e. $\mathrm{P}^{\mathrm{NEXPTIME}} = \mathrm{NP}^{\mathrm{NEXPTIME}}$, so $\mathrm{P}^{\mathrm{NEXPTIME}} = \mathrm{FewP}^{\mathrm{NEXPTIME}}$. There are thus no sparse sets in $\mathrm{FewP}^{\mathrm{NEXPTIME}} - \mathrm{P}^{\mathrm{NEXPTIME}}$, so by Theorem 4.4 every sparse set in $\mathrm{P}^{\mathrm{NEXPTIME}}$ is $\mathrm{P}^{\mathrm{NEXPTIME}}$-printable. $\square$

Via Proposition 3.3, the above give the following results regarding small generalized Kolmogorov complexity:

Corollary 4.11

1. *Every sparse set* S *is in* $\mathrm{SGK}^{\mathrm{FewP}^{S}}$.
2. *Every sparse set in* P *is in* $\mathrm{SGK}^{\mathrm{FewP}}$.
3. *Every sparse set in* Σ_k^{P} *is in* $\mathrm{SGK}^{\Sigma_{k+1}^{\mathrm{P}}}$.
4. *Every sparse set in* PH *is in* $\mathrm{SGK}^{\mathrm{PH}}$.
5. *Every sparse set in* PSPACE *is in* $\mathrm{SGK}^{\mathrm{PSPACE}}$.
6. *Every sparse set in* $\mathrm{P}^{\mathrm{NEXPTIME}}$ *is in* $\mathrm{SGK}^{\mathrm{NEXPTIME}}$.

Turning our attention to printability relative to the class of recursive sets, the following theorem shows that it is precisely the recursive sparse sets that are P-printable relative to a recursive oracle.

Theorem 4.12 *For any set* S*,* S *is* $\mathrm{P}^{\mathrm{REC}}$*-printable if and only if* S *is recursive and sparse.*

Proof Every set that is $\mathrm{P}^{\mathrm{REC}}$-printable must be in $\mathrm{P}^{\mathrm{REC}}$ and must therefore be recursive. It must also be sparse to be able to be printed in polynomial time, regardless of the oracle.

For the right to left direction, assume S is recursive and sparse. Because S is sparse, it is $\mathrm{P}^{\mathit{prefix}(S)}$-printable. Since S is recursive, so is *prefix*(S), therefore S is $\mathrm{P}^{\mathrm{REC}}$-printable. $\square$

This technique actually yields a more general result than that above. While this theorem deals with $\mathrm{P}^{\mathrm{REC}}$-printability and sparseness, other deterministic printabilities (relative to REC) and other densities work as well. The time needed for the printing is based on the length of the output (or the length of the input, if that is greater). All the "work" is done by the recursive oracle.

Corollary 4.13 *There exist sparse sets and tally sets that are not* $\mathrm{P}^{\mathrm{REC}}$*-printable.*

Corollary 4.14 *There exist self-*P*-printable sets that are not* $\mathrm{P}^{\mathrm{REC}}$*-printable.*

Corollary 4.15 *Every recursive sparse set is in* $\mathrm{SGK}^{\mathrm{REC}}$.

Note that the converse to the last corollary is false because there exist nonrecursive tally sets, all of which are in SGK. The question arises whether or not this corollary can be strengthened. Is every sparse set, including nonrecursive sparse sets, in $\mathrm{SGK}^{\mathrm{REC}}$? The answer to this question is "no", but it is not as immediate as for the $\mathrm{P}^{\mathrm{REC}}$-printable case, as there are nonrecursive sets, including all the tally sets, that are in $\mathrm{SGK}^{\mathrm{REC}}$. A more general result is presented.

Theorem 4.16 *For all fully time constructible functions $f(n) \in o(n)$ and $g(n)$, there exist sparse sets that are not subsets of $K^A[f(n), g(n)]$ for any recursive oracle A,*

Proof Let $f(n) \in o(n)$ and $g(n)$ be fully time constructible functions, and let B be an infinite set of Kolmogorov-random strings that is sparse enough so that for all n, the number of strings in B of length less than or equal to n is at most $f(n)$. Assume that B is a subset of $K^A[f(n), g(n)]$ for some recursive set A. Let l be the size of a Turing machine that determines membership in A, and let m be the size of the oracle Turing machine that restores the compressed strings. Thus, there is a constant k such that any string of length n in B can be generated by a program of length $k + l + m + f(n)$. This is so because this program could contain the program that determines membership in A, the program that restores the compressed strings, and the compressed string "hard-coded" in. But for any k, l, m, and $f(n) \in o(n)$ there exist n greater than $k + l + m + f(n)$, contradicting the Kolmogorov-randomness of the strings in B with long lengths. $\square$

Corollary 4.17 *There exist sparse sets that are not in* $\mathrm{SGK}^{\mathrm{REC}}$.

We can improve on this theorem a bit by showing that there are even self-P-printable sets that are not in $\mathrm{SGK}^{\mathrm{REC}}$. First, it should be noted (and is not shown here) that for any sparse set S, *prefix*(S) is self-P-printable ([Rub88]).

Theorem 4.18 *There exist self-*P*-printable sets that are not in* $\mathrm{SGK}^{\mathrm{REC}}$.

Proof Let S be a sparse set that is not in $\mathrm{SGK}^{\mathrm{REC}}$, as in the above corollary. Note that the set $S' = \{\langle y, 0^{|y|}\rangle \mid y \in S\}$ is also a sparse set that is not in $\mathrm{SGK}^{\mathrm{REC}}$. Let $R = \textit{prefix}(S)$, so R is self-P-printable. But $S' \subseteq R$, so R is not in $\mathrm{SGK}^{\mathrm{REC}}$. $\square$

The question then arises whether or not *every* set in $\mathrm{SGK}^{\mathrm{REC}}$ is self-P-printable. The answer to this is "no", as the following demonstrates.

Theorem 4.19 *There exists a set that is* $\mathrm{P}^{\mathrm{REC}}$*-printable that is not polynomial time Turing equivalent to any tally set.*

Proof Long ([Lon85]) shows that there exists a sparse recursive set that is not polynomial time Turing equivalent to any tally set. By Theorem 4.12, this set must be $\mathrm{P}^{\mathrm{REC}}$-printable. □

Corollary 4.20 *There exists a set in* $\mathrm{SGK}^{\mathrm{REC}}$ *that is not polynomial time Turing equivalent to any tally set.*

Corollary 4.21 *There exists a set that is* $\mathrm{P}^{\mathrm{REC}}$*-printable that is not self-*P*-printable.*

Proof Every self-P-printable set is polynomial time Turing equivalent to some tally set ([BB86, Rub91]), so the set in Theorem 4.19 that is $\mathrm{P}^{\mathrm{REC}}$-printable and is not polynomial time Turing equivalent to any tally set is not self-P-printable. □

Corollary 4.22 *There exists a set in* $\mathrm{SGK}^{\mathrm{REC}}$ *that is not self-*P*-printable.*

Thus, $\mathrm{SGK}^{\mathrm{REC}}$ and the class of self-P-printable sets are incomparable.

Using the class of recursive sets as the class to be relativized, once again the class of tally oracles is more powerful than the class of recursive oracles. $\mathrm{REC}^{\mathrm{REC}} = \mathrm{REC}$, which clearly does not contain all sparse sets. $\mathrm{REC}^{\mathrm{TALLY}}$, however, includes REC, all the sparse sets and more. In fact, *every* set is in $\mathrm{REC}^{\mathrm{TALLY}}$, as an arbitrary set A is in REC^{T}, where $T = \{1^x \mid x \in A\}$.

5 Query Restriction and Sparse Characterization

If the number of queries to any oracle is restricted to $O(\log n)$ for inputs of length n, however, we no longer get the entire class of sparse sets. In fact, all we get is SGK. A more general result of restricting the number of oracle queries to a generalized Kolmogorov complexity class is presented here.

Theorem 5.1 *For all fully time constructible functions $f(n)$, $g(n)$ and $h(n)$ and every oracle A, $K^{A[h(n)]}[f(n), g(n)] \subseteq K[2(f(n)+h(n)), k(g(n)+p(n))]$ for some polynomial p and some constant k.*

Proof Let B be a set in $K^{A[h(n)]}[f(n), g(n)]$, where $f(n)$, $g(n)$ and $h(n)$ are fully time constructible, and machine M is the oracle machine that restores an original string from its compressed string. Let x be a string in B of length n, let y be its compressed string, and let $a_1, a_2, ..., a_{h(n)}$ be the answers to the queries M makes to the oracle A – 0 for "no" and 1 for "yes" – in the restoration of x from y.

Consider now the pair $z = \langle y, a_1a_2...a_{h(n)}\rangle$. Because y has length at most $f(n)$ and $a_1a_2...a_{h(n)}$ has length $h(n)$, z has length at most $2(f(n)+h(n))$. z will be the compressed string for x for a machine that will restore x from z in time $k(g(n)+p(n))$ without an oracle, for some constant k and some polynomial p. This then demonstrates that $B \subseteq K[2(f(n)+h(n)), k(g(n)+p(n))]$.

To restore x from z, simulate M's restoration of x from y (which is contained in z) using the answers to the oracle calls contained in z instead of actually making the oracle calls. The polynomial p is needed for the overhead of the unpairing and machine simulation if g is too small, and the constant is necessary in case g and h are large.

$\square$

Corollary 5.2 *For every oracle A and every constant k,* $\mathrm{SGK}^{A[k\log n]} = \mathrm{SGK}$.

Corollary 5.3 *For every oracle A and every constant k, every set that is* $\mathrm{P}^{A[k\log n]}$*-printable is in* SGK.

Finally, a generalized Kolmogorov complexity characterization of the sparse sets is presented.

Theorem 5.4 *A set S is sparse if and only if $S \subseteq K^S[c\log n, c2^n]$ for some c.*

Proof The implication from right to left is clear. For the left to right direction, let S be a sparse set, and let x be a string in S of length n. The compression of x is defined to be $y = r_S(x)$, the ranking of x in S. To restore x, starting with λ check each string in lexicographic order for membership in S. The y^{th} string in S is x. Because there are fewer than $2 * 2^n$ strings lexicographically smaller than x, the restoration can be done within $c2^n$ steps for some c. $\square$

Corollary 5.5 *For any complexity class $\mathcal{C}$, every sparse set in $\mathcal{C}$ is a subset of $K^{\mathcal{C}}[c\log n, c2^n]$ for some c.*

Corollary 5.6 *Every sparse set in* P *is a subset of $K[c\log n, c2^n]$ for some c.*

References

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

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

[BB86] J. Balcázar and R. Book. Sets with small generalized Kolmogorov complexity. *Acta Informatica*, 23:679–688, 1986.

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

[BGS75] T. Baker, J. Gill, and R. Solovay. Relativizations of the P =? NP question. *SIAM J. Comput.*, 4(4):431–441, Dec. 1975.

[Cha66] G. Chaitin. On the length of programs for computing finite binary sequences. *J. Assoc. Comput. Mach.*, 13:547–569, 1966.

[Har83] J. Hartmanis. Generalized Kolmogorov complexity and the structure of feasible computations. In *Proc. 24th IEEE Symp. on Foundations of Computer Science, Tucson, Arizona*, pages 439–445, 1983.

[Hem87] L. Hemachandra. *Counting in Structural Complexity Theory.* PhD thesis, Cornell University, 1987.

[HH88] J. Hartmanis and L. Hemachandra. On sparse oracles separating feasible computation classes. *Info. Proc. Letters*, 28:291–296, 1988.

[HIS85] J. Hartmanis, N. Immerman, and V. Sewelson. Sparse sets in NP $-$ P: EXPTIME versus NEXPTIME. *Information and Control*, 65:158–181, 1985.

[HR92] L. Hemachandra and R. Rubinstein. Separating complexity classes with tally oracles. *Theoretical Computer Science*, 92:309–318, 1992.

[HY84] J. Hartmanis and Y. Yesha. Computation times of NP sets of different densities. *Theoretical Computer Science*, 34:17–32, 1984.

[Kol65] A. Kolmogorov. Three approaches to the quantitative definition of information. *Prob. Inform. Trans.*, 1:1–7, 1965.

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

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

[Rub86] R. Rubinstein. A note on sets with small generalized Kolmogorov complexity. Technical Report TR #86-4, Iowa State University, Ames, IA, March 1986.

[Rub88] R. Rubinstein. *Structural complexity classes of sparse sets: intractability, data compression and printability.* PhD thesis, College of Computer Science, Northeastern University, Boston, MA, 1988.

[Rub91] R. Rubinstein. Self-P-printability and polynomial time Turing equivalence to a tally set. *SIAM J. Comput.*, 20(6):1021–1033, December 1991.

[Sch86] U. Schöning. *Complexity and Structure.* Lecture Notes in Computer Science Vol. 211. Springer-Verlag, Berlin, 1986.

[Sol64] R. Solomonoff. A formal theory of inductive inference, Part 1 and Part 2. *Info. and Control*, 7:1–22, 224–254, 1964.