SIAM J. COMPUT.
Vol. 17, No. 6, December 1988

P-PRINTABLE SETS*

ERIC W. ALLENDER† AND ROY S. RUBINSTEIN‡

Abstract. P-printable sets arise naturally in the studies of generalized Kolmogorov complexity and data compression, as well as in other areas. We present new characterizations of the P-printable sets and present necessary and sufficient conditions for the existence of sparse sets in P that are not P-printable. As a corollary to one of our results, we show that the class of sets of small generalized Kolmogorov complexity is exactly the class of sets which are P-isomorphic to a tally language.

Key words. Kolmogorov complexity, sparse sets, P-isomorphisms, AuxPDAs, computational complexity

AMS(MOS) subject classifications. 68Q15, 68Q30, 68Q05, 03D15, 03D30

1. Introduction. Sparse sets have the useful property that for every n there is a table of size polynomial in n that lists the elements of the set of size less than or equal to n. Letting S be an arbitrary sparse set, it is conceivable that by storing a polynomial size table for sufficiently large n, we might have immediate access to all the information about S that we might ever need, despite the fact that the uniform complexity of S may be very high. Unfortunately, the complexity of producing such a table may be much greater than the complexity of recognizing the set S. If the complexity of producing a table for S is not much greater than the complexity of recognizing S, i.e., the complexity of producing a table for S is polynomial-time Turing reducible to S, then S is said to be *self* P-*printable*. If the complexity of producing a table for S is easy, i.e., a table for S can be produced in polynomial time without using S as an oracle, then S is P-*printable*. (Formal definitions for this and other concepts are given in § 2.) Obviously, P-printable sets belong to P. The notion of P-printability was introduced in [HY84] and was further explored in [HIS85].

The idea of generalized Kolmogorov complexity was introduced in [Har83] and [Sip83], and the connection between it and P-printability has been studied independently in [BB86], [HH86], and [Rub86b]. Generalized Kolmogorov complexity is a measure of how far a string can be compressed and how fast it can be restored. The relativized version of generalized Kolmogorov complexity allows the use of an oracle in the restoration. Sets containing only strings that can be greatly compressed and quickly restored (to be defined more precisely in the next section) are said to have *small generalized Kolmogorov complexity.* The papers [BB86] and [HH86] show that a set is self-P-printable if and only if it has small generalized Kolmogorov complexity relative to itself. This has the corollary (independently proved in [Rub86b]) that a set is P-printable if and only if it is in P and has small generalized Kolmogorov complexity.

P-printable sets are also shown here to have close connections with tally sets. Section 3 presents the result that a set is P-printable if and only if it is P-isomorphic to a tally language in P, if and only if it is in P and has small generalized Kolmogorov complexity. This has the important corollary that sets of small generalized Kolmogorov complexity are precisely those that are P-isomorphic to a tally language. This improves

* Received by the editors November 18, 1986; accepted for publication (in revised form) February 3, 1988.

† Department of Computer Science, Rutgers University, New Brunswick, New Jersey 08903. The work of this author was supported in part by the National Science Foundation under grant MCS 81-03608 and by a 1986 Rutgers University Research Council Summer Fellowship.

‡ College of Computer Science, Northeastern University, Boston, Massachusetts 02115. The work of this author was supported in part by the National Science Foundation under grant DCR84-02033 and by the National Security Agency under grant MDA904-87-H-2020.

upon a result in [BB86] that the sets of small generalized Kolmogorov complexity are those that are "semi-isomorphic" to a tally language. Since most of the properties of sets studied in complexity theory are invariant under P-isomorphisms, the significance of this corollary is that the sets of small generalized Kolmogorov complexity are essentially identical to the tally languages from the point of view of complexity theory.

It is also shown that a set is P-printable if and only if it is sparse and has a ranking function computable in polynomial time. A ranking function is a function that maps an element of a set to its index in the lexicographic ordering of the set. They are presented in the context of data compression in [GS85] and are of independent interest [All85] and [Hem87].

Section 4 presents machine-based characterizations of the P-printable sets. It is shown there that a set is P-printable if and only if it is sparse and accepted by a deterministic one-way logspace-bounded AuxPDA, if and only if it is sparse and accepted by a nondeterministic one-way logspace-bounded AuxPDA. This presents a parallel to the result of Cook [Coo71] that a set is in P if and only if it is accepted by a (two-way) logspace-bounded AuxPDA. It is surprising that restricting AuxPDAs to have a one-way input head leads to a characterization of P-printable sets.

One-way AuxPDAs are not very powerful machines, as it was shown in [Bra77b] that some relatively "natural" languages in P are not accepted by one-way AuxPDAs of sublinear space complexity. With this result, any argument showing the existence of a sparse set in P (or even PSPACE) that is not accepted by any one-way logspace-bounded AuxPDA is strong enough to settle several outstanding problems in complexity theory, since, for example, P = PSPACE implies all sparse sets in PSPACE are P-printable [HY84].

Section 5 addresses some structural questions concerning P-printable sets. While it is clear that every P-printable set is sparse and in P, it is not known whether or not there is a sparse set in P that is not P-printable. Here we present the result that there is a sparse set in P that is not P-printable if and only if there is a sparse set in DLOG that is not P-printable, if and only if there is a sparse set in FewP − P. FewP was introduced in [All86b] as a class of sets between UP and NP (see § 2 for more details). This section concludes with the results that there are infinite sparse sets of time complexity arbitrarily close to polynomial that have finite intersection with every P-printable set, and that every infinite set in P has an infinite P-printable subset if and only if every infinite set in NP has an infinite P-printable subset.

2. Preliminaries. While it is assumed that the reader is familiar with the basic concepts and structures from complexity theory (such as P, NP, and DLOG), some of the more important and not universally known ones are presented here, along with notation.

We use the standard lexicographic ordering $\leqq$ 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 $\{0, 1\}^*$. Although a language may be referred to as a subset of $\{0, 1, \#\}^*$, this is merely a notational convenience; such a language should be thought of as a subset of $\{00, 11, 01\}^*$. A tally language is a subset of $\{0\}^*$.

Strings are sometimes used to denote numbers (and vice versa) by letting the string w denote the number whose binary representation is $1w$. This preserves the ordering and allows us to write, for example, $|w| = \lfloor \log w \rfloor$. All logarithms are base two. We use EXPTIME to denote DTIME$(2^{O(n)})$ and NEXPTIME to denote NTIME$(2^{O(n)})$.

DEFINITION 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 finite strings was introduced independently by Kolmogorov [Kol65] and Chaitin [Cha66], [Cha75] as a way to measure the randomness of a finite string (or equivalently, the amount of information contained in a string). The Kolmogorov complexity of a finite binary string is the length of the shortest program that generates it. Intuitively it can be seen that if a string can be generated by a program shorter than itself (i.e., it can be *compressed*), it must contain some redundant information. A string is *random* if it cannot be compressed.

One limitation in using this as the definition of randomness is that it provides no limits or restrictions on the computation time to generate the original string from its smallest program. Time-bounded versions of Kolmogorov complexity have been considered by Ko [Ko86], Sipser [Sip83], and more recently by Hartmanis [Har83].

Hartmanis introduced a two-parameter version of Kolmogorov complexity (now called *generalized Kolmogorov complexity*) that includes information about not only how far a string can be compressed, but how fast it can be restored. This generalized Kolmogorov complexity is further explored in [All87], [BB86], [Huy86], [KOSW86], [Lon86], [Rub86b], and [Rub86a]. It is Hartmanis's definition of generalized Kolmogorov complexity that is presented here.

DEFINITION 2. For a Turing machine M_u and functions g and G mapping natural numbers to natural numbers, let

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

We will refer to y as the *compressed string*, x as the *restored string*, $g(n)$ as the *compression*, and $G(n)$ as the *restoration time*. It was shown in [Har83] that there exists a Turing machine M_u (called a *universal* Turing machine) such that for any other Turing machine M_v there exists a constant c such that $K_v[g(n), G(n)] \subseteq K_u[g(n)+c, cG(n) \log G(n)+c]$. Dropping the subscript, $K[g(n), G(n)]$ will actually denote $K_u[g(n), G(n)]$.

DEFINITION 3. A set is said to have *small generalized Kolmogorov complexity* if it is a subset of $K[k \log n, n^k]$ for some k.

Clearly every set with small generalized Kolmogorov complexity is sparse. Note that the definition of small generalized Kolmogorov complexity is robust enough to handle the small difference between the universal machine mentioned above and any other machine.

We now give the definition of P-printability.

DEFINITION 4. A set S is *polynomial-time printable* (P-*printable*) if there exists a k such that all the elements of S up to size n can be printed by a deterministic machine in time $n^k + k$.

Clearly every P-printable set is necessarily sparse and in P.

Goldberg and Sipser [GS85] discuss compression of languages and ranking and present the following definitions.

DEFINITION 5. A function $f:\Sigma^* \to \Sigma^*$ is a *compression* of language L if f is one-to-one on L and for all except finitely many $x \in L$, $|f(x)| < |x|$.

DEFINITION 6. A language L is *compressible in time T* if there is a compression function f for L that can be computed in time T, and the "inverse" of f, $f^{-1}: f(L) \to L$, such that for any $x \in L$, $f^{-1}(f(x)) = x$, can be computed in time T.

This clearly relates to generalized Kolmogorov complexity in that compressible languages do not contain more than finitely many random strings, and the compression time (in this sense) relates to the second parameter of the generalized Kolmogorov complexity. Note that in this version, the compression time is a bound on both the compression and restoration, whereas the time parameter of generalized Kolmogorov complexity refers only to the restoration time.

Reference [GS85] additionally presents the following definitions.

DEFINITION 7. A function f *optimally compresses* a language L if for any $x \in L$ of length n, $|f(x)| \leqq \lceil \log (\sum_{i=0}^{n} |L^i|) \rceil$, where L^i is the set of strings in L of length i.

DEFINITION 8. For any set $L \subseteq \Sigma^*$, the *ranking function* for L, $r_L : \Sigma^* \to N$, is given by $r_L(x) = |\{w \in L \mid w \leqq x\}|$.

The ranking is a special kind of optimal compression.

Allender [All86b] defined the complexity class FewP to be between UP and NP as follows.

DEFINITION 9. FewP is the class of languages that are accepted by nondeterministic polynomial-time Turing machines M for which there is a polynomial p such that for all inputs w, if M accepts w, there are fewer than $p(|w|)$ accepting computations of M on w.

FewP is related to the class UP [Ber77], [Val76], [GS84] of languages in NP that are accepted by nondeterministic polynomial-time bounded Turing machines with unique accepting computations. Both UP and FewP are subclasses of NP defined by restricting the number of accepting computations. Densities of accepting computations were previously considered in [Mor82], but no class equivalent to FewP was introduced.

DEFINITION 10. A function f is a P-*isomorphism* if it is a bijection such that both f and f^{-1} are computable in polynomial time. Two sets A and B are P-*isomorphic* if there is some P-isomorphism f such that $A = f(B)$.

Auxiliary pushdown automata (AuxPDAs) are due to Cook [Coo71]. An AuxPDA is a Turing machine with a pushdown store in addition to a worktape. When we bound the space used by an AuxPDA, we bound only the space used on the worktape; the space used on the pushdown store is "free." Useful results about AuxPDAs are summarized in [HU79]. The fact that the languages accepted in time $T(n)^{O(1)}$ are precisely the sets accepted by log $T(n)$ space-bounded deterministic and nondeterministic AuxPDAs [Coo71] will be used.

A one-way AuxPDA is an AuxPDA with a one-way input head. One-way AuxPDAs have been studied before in [Bra77b], [Bra77a], [Chy77], [WB79], and [Wec80]. In [BDG85] and [Huy85] one-way AuxPDAs were investigated in connection with restricted forms of nonuniform complexity. In most studies of one-way AuxPDAs, the machine starts its computation with log n space marked off on its worktape; that is the model used here.

3. Structural characterizations. In this section the non-machine-based characterizations of the class of P-printable sets presented in the Introduction are proved.

THEOREM 1. *The following are equivalent*:

(1) *S is* P-*printable.*

(2) *S is sparse and has a ranking function computable in polynomial time.*

(3) *S is* P-*isomorphic to some tally set in* P.

(4) $S \subseteq K[k \log n, n^k]$ *and* $S \in$ P.

Note. The equivalence of (1) and (4) also appears in [BB86], [HH86], and [Rub86b].

Proof. [(1) $\Rightarrow$ (2)]. The proof is immediate.

[(2) $\Rightarrow$ (3)]. Let S have a ranking function r_1 computable in polynomial time, and let there be fewer than $p(n)$ strings of length n in S. Thus the function r_2 given by $r_2(w) = w - r_1(w)$ is a ranking function for the complement of S. (Recall that strings can represent numbers, as explained in § 2.) Also, the set $T = \{0^{np(n)+i} \mid r_1(1^{n-1}) <$

$i \leq r_1(1^n)\}$ is a tally set in P; let r_3 be a ranking function for the complement of T. As was noted in [GS85], all these ranking functions have inverses that are computable in time polynomial in the length of their output. It is now easy to show that the function that takes x of length n to $0^{np(n)+r_1(x)}$ if $x \in S$ and to $r_3^{-1}(r_2(x))$ if $x \notin S$ is a P-isomorphism mapping S onto T.

[(3) $\Rightarrow$ (4)]. Let S be P-isomorphic to $T \subseteq 0^*$ via some isomorphism f such that both f and f^{-1} are computable in time n^c. The compressed form of a string $x \in S$ of length n is z, the binary representation of $|f(x)|$. Since f is computable in time at most n^c, it follows that $|f(x)| \leq n^c$, and thus $|z| \leq c \log n$. To get x back from z, simply compute $f^{-1}(0^z)$. Since $|0^z| = |f(x)| \leq n^c$, computing 0^z from z takes time at most polynomial in n, call it n^l. Computing $f^{-1}(0^z)$ takes time at most $|0^z|^c \leq (n^c)^c = n^{c^2}$. The computation time of x from z is thus $\leq n^{c^2} + n^l \leq n^k$ for some $k \geq c$. Thus $S \subseteq K[k \log n, n^k]$. If T is in P, then S will also be in P since they are P-isomorphic.

[(4) $\Rightarrow$ (1)]. Assume that $S \in \mathrm{P}$ and that for some k, $S \subseteq K[k \log n, n^k]$. On input n, for each of the $n^{k+1}-1$ strings of length $\leq k \log n$, run M_u for at most n^k steps and, if the computation has completed and the result is in S, print it. This process can clearly be done in time polynomial in n. □

It should be pointed out that results similar to those of Theorem 1 were presented in [BB86] as part of an investigation of sets of small generalized Kolmogorov complexity. In [BB86], Balcázar and Book define "semi-isomorphisms" and show that a set has small generalized Kolmogorov complexity if and only if it is semi-isomorphic to a tally set. Using Theorem 1 we can improve upon their result. First, however, we need the following easy result.

PROPOSITION 2. *For all M_v and k, $K_v[k \log n, n^k] \in \mathrm{P}$.*

This proposition is readily seen to be true by the following procedure: to determine if a string x of length n is in $K_v[k \log n, n^k]$, run machine M_v on all strings of length less than or equal to $k \log n$, and accept if and only if M_v outputs x on one of these strings.

COROLLARY 3. *There exists a k such that $A \subseteq K[k \log n, n^k]$ if and only if A is P-isomorphic to a tally set.*

Proof. The proof from right to left follows from the argument given in the proof of [(3) $\Rightarrow$ (4)] of Theorem 1. For the forward direction, let $A \subseteq K[k \log n, n^k]$. By Theorem 1 and Proposition 2, $K[k \log n, n^k]$ is P-isomorphic to some tally set T in P via some P-isomorphism f. It is now clear that A is P-isomorphic to $f(A) \subseteq T \subseteq \{0^*\}$. □

While no good upper bound is known for the generalized Kolmogorov complexity of sparse sets in P without assuming P-printability, it is not hard to show that every sparse set in P is a subset of $K[O(\log n), 2^{O(n)}]$. However, that is not very informative since every sparse set in EXPTIME is contained in $K[O(\log n), 2^{O(n)}]$.

We remark that, while it is not known whether or not there are sparse sets in P that are not subsets of $K[k \log n, n^k]$ for any k, it is not hard to show that, for any time-constructible function $T(n)$ that is greater than every polynomial, there is a sparse in DTIME $(T(n))$ that is not a subset of $K[k \log n, n^k]$ for any k. Also, there is a nonrecursive sparse set that is not a subset of $K[S(n), T(n)]$ for any $S(n) = o(n)$ and any recursive $T(n)$. As an example of such a set, consider a set consisting of exactly one Kolmogorov-random string of each length n. Sparse sets such as these are not P-isomorphic to any tally set.

4. Machine-based characterizations. We now present machine-based characterizations of the P-printable sets. Cook [Coo71] showed that a set is in P if and only if it is accepted by a (two-way) logspace-bounded AuxPDA. By restricting this machine to be one way, we obtain a characterization of the P-printable sets.

THEOREM 4. *The following are equivalent*:

(1) *S is* P-*printable.*

(2) *S is sparse and is accepted by a deterministic one-way logspace-bounded* AuxPDA.

(3) *S is sparse and is accepted by a nondeterministic one-way logspace-bounded* AuxPDA.

Proof. [(1) $\Rightarrow$ (2)]. If S is P-printable, then there is a function f computable in polynomial time such that $f(n)$ encodes the elements of S of length less than or equal to n. Thus the set $A=\{n\#i \mid \text{the } i\text{th bit of } f(n)=1\}$ is in EXPTIME and can thus be recognized by a deterministic AuxPDA that uses linear space. Equivalently, an AuxPDA with $n\#i$ written on its worktape can determine if $n\#i\in A$ without referencing its input tape (i.e., without moving its input head).

Using this trick, a one-way AuxPDA can easily check if the word on its input tape agrees with the ith word in the enumeration of S. If the ith word disagrees with the input in the jth position, it is not necessary to move the input head back to the start of the tape in order to compare the input with the $(i+1)$st word. Instead, find the next word in the enumeration which agrees with the ith word up to (and not including) the jth position. The following algorithm makes this precise. Let p be a polynomial such that for all n there are fewer than $p(n)$ elements of S of length less than or equal to n. The following algorithm can be carried out by a deterministic logspace-bounded AuxPDA.

```
begin
    At the start of the computation, $w = w_1 \cdots w_n$ is on the input tape and $\lfloor \log n \rfloor$
                space is marked off on the worktape.
    $r := 1^{\lfloor \log n \rfloor}$      //Note that the AuxPDA cannot know what $n$ is until it has read
                  the entire input. Clearly, however, $n \leqq r \leqq 2n$.
    $i := 1$
    $j := 0$          //The pair $(i, j)$ will be maintained so that the word on the input
                  tape will match the $i$th word in the enumeration of $S$ up through
                  position $j$.
    step one
        Increment $j$ until a $j$ is found (by repeated calls of the form $r\#1 \in A$?, $r\#2 \in$
          $A$?, $\cdots$) such that the $j$th input symbol differs from the $j$th symbol of the
          $i$th word of the enumeration of $S$, or until $j = n$.
        If $j = n$ and the first $n$ symbols of the input match the first $n$ symbols of the
          $i$th word of the enumeration and the $i$th word has length $n$, then halt and
          accept.
        Otherwise go on to step two.
    step two
        By making repeated calls of the form $r\#1 \in A$?, $r\#2 \in A$?, $\cdots$, find the least
          $i'$ such that $i < i' < p(r)$, and such that the first $j-1$ symbols of the $i$th and
          the $i'$th words of the enumeration of $S$ agree.
        If no such $i'$ exists, halt and reject.
        Otherwise set $i := i'$ and go to step one.
end
```

The AuxPDA described above simply searches through the enumeration of S until some word is found that matches the input, and accepts if such a word is found. It is easy to see that the algorithm is correct. Note that the only time the input head is moved is in step one, and that the input need only be read in one scan from left to right.

$[(2) \Rightarrow (3)]$. The proof is immediate.

$[(3) \Rightarrow (1)]$. Let S be sparse and accepted by a nondeterministic one-way logspace-bounded AuxPDA M. It is easy to construct another nondeterministic one-way AuxPDA accepting the set $S' = \{0^n \# w \mid \text{there is a string } x \text{ such that } |wx| = n \text{ and } wx \in S\}$. (This machine will store n in binary on its worktape and then simulate M on w. Then it will continue the simulation by guessing the $n - |w|$ characters of x. Since M is one-way, only one bit of x needs to be stored at any time.) Thus $S' \in \mathrm{P}$. By using S' it is now easy to construct the elements of S, bit by bit, and thus it follows that S is P-printable.

(This proof was pointed out to the first author by Osamu Watanabe, and simplifies the proof of the same result, which was presented in [All86b]). □

An interesting result somewhat related to Theorem 4 has recently been proved by Ibarra and Ravikumar; in [IR86] they show that all sparse CFLs are bounded.

The study of P-uniform circuit complexity is also related to the study of P-printable sets, and was part of the motivation for the present investigation of P-printable sets. A P-*uniform family of circuits* is a set of circuits $\{C_n \mid n \geqq 1\}$ such that the function $n \rightarrow C_n$ is computable in time polynomial in n. P-uniform circuits are studied in [All85], [All86a], [BCH84], and [vzG84].

When studying circuit complexity classes, the class of functions that can be computed by very fast circuits (i.e., circuits of small depth) is of special interest. Thus, we are led to consider the class P-uniform NC (PUNC), the class of languages accepted by P-uniform circuits of depth $\log^{O(1)} n$ [All86a].

Note that if $\{C_n \mid n \geqq 1\}$ is a P-uniform family of circuits, then it is a P-printable set. Thus in some sense, the sets in PUNC are those sets that can be computed very easily, relative to some P-printable set. For example, it is fairly easy to see that all P-printable sets are in PUNC.

Given this connection between P-printable sets and P-uniform circuit complexity, it is natural to consider the question of whether or not all sparse sets in PUNC are P-printable. It was shown in [All85], [All86a] that a set is in PUNC if and only if it is accepted by a logspace-bounded AuxPDA that moves its input head $2^{\log^{O(1)} n}$ times. Thus both PUNC and the class of P-printable sets have characterizations in terms of AuxPDAs with restricted access to the input. On the other hand, we show in the next section that every sparse set in PUNC is P-printable if and only if every sparse set in P is P-printable.

5. Some structural questions. Questions relating to the existence and structure of non-P-printable sets are now presented.

While it is clear that all P-printable sets are sparse and in P, it is not known whether there are sparse sets in P that are not P-printable. The following theorem shows some equivalent conditions.

THEOREM 5. *The following are equivalent*:

(1) *There is a sparse set in* P *that is not* P-*printable.*

(2) *There is a sparse set in* DLOG *that is not* P-*printable.*

(3) *There is a sparse set in* FewP − P.

Proof. If S is a sparse set in P that is not P-printable, then the set $\{x \# 0^n \mid x \text{ is a prefix of a string of length } n \text{ in } S\}$ is a sparse set in FewP − P, thus proving $[(1) \Rightarrow (3)]$. $[(3) \Rightarrow (2)]$ is proved by the observation that if S is sparse and in FewP − P, then the set of accepting computations of a machine accepting S is a sparse set in DLOG that is not P-printable. Because DLOG $\subseteq$ P, the implication of (1) by (2) is clear. □

As a consequence of the fact that DLOG $\subseteq$ PUNC $\subseteq$ P, the above is also equivalent to the existence of a sparse set in PUNC that is not P-printable. This result provides a nice parallel to the result in [HY84] that there is a sparse set in NP that is not P-printable if and only if there is a sparse set in NP-P. While NP is the complexity class most closely related to the existence of non-P-printable sparse sets in NP, FewP is the complexity class most closely related to the existence of non-P-printable sets in P and DLOG. As a side note, it is not known whether all sparse sets in P are in PUNC.

Turning now to the structure of non-P-printable sets, we are interested in determining what they "look like." Can they be immune to the P-printable sets or must they have a P-printable subset? The following two theorems partially answer these questions.

THEOREM 6. *Let S be a set in* P *that is not* P-*printable, and let $T(n)$ be a time-constructible function that grows faster than any polynomial. Then there is an infinite set $A \in$ DTIME $(T(n))$ such that $A \subseteq S$ and A has finite intersection with every* P-*printable set.*

Proof. This is proved using techniques similar to those used in, e.g., [Orp86], where a result with a similar flavor concerning complexity cores was proved. Let $M_1, M_2, \cdots$ be an indexing of polynomial-time machines taking input from 0^*; each such machine can be viewed as taking 0^n as input and producing a list of strings of length n. Thus $\{M_i(0^*) \mid i \geqq 1\}$ is a representation of all P-printable sets. Since the intersection of a set in P with a P-printable set is P-printable, S is not contained in $M_i(0^*)$ for any i.

The idea of the proof is to find, for all i, a string x_i in S that is not in $M_j(0^*)$ for any $j \leqq i$. Such a string x_i must exist, or else $S \subseteq M_1(0^*) \cup M_2(0^*) \cup \cdots \cup M_i(0^*)$, and thus S is P-printable.

Thus the function $r(i) = \min\{n \mid \exists x \in S, |x| = n \text{ and } x \notin M_j(0^n) \text{ for all } j \leqq i\}$ is total, monotone, and recursive. Let s be a total, monotone, recursive function that is greater than r and has the property that the function $i \to s(i)$ can be computed in time linear in $s(i)$; taking s to be the time complexity function for some machine computing r in unary will suffice.

We also require that our indexing satisfy the condition that machine M_i has running time bounded by $T(n)/n$ on inputs of length $n \geqq i$. Since T grows faster than any polynomial, this condition is easy to satisfy (see [Orp86]).

The following routine will run in time $O(T(n))$ and will accept a subset A of S that has finite intersection with every P-printable set.

```
begin on input $x$ of length $n$
    Compute $s(1), s(2), \cdots$ until some $i$ is found such that $s(i) \leqq n < s(i+1)$.
        //This step takes $O(n^2)$ time.
    Accept $x$ iff $x \in S - M_j(0^n)$ for all $j \leqq i$.
        //This step takes $\leqq T(n)$ time, since (assuming without loss of generality
          that $i < n$) each of the $i$ simulations performed in this step requires at
          most $T(n)/n$ steps.
end
```

It is clear that $A \subseteq S$, and that this routine runs in time $O(T(n))$. If $x \in A$ then for some i such that $s(i) \leqq n < s(i+1)$, x is in $S - M_j(0^n)$ for all $j \leqq i$. Thus $M_i(0^*)$ contains no elements in A of length $\geqq s(i)$. It only remains to show that A is infinite.

Let i and i' be numbers such that $s(i') \leqq r(i) < s(i'+1)$. By the definition of r, there is a string $x_i \in S$ of length $r(i)$ such that $x_i \notin M_j(0^{|x_i|})$ for all $j \leqq i$. In particular, $x_i \notin M_j(0^{|x_i|})$ for all $j \leqq i'$, since $i' \leqq i$. Thus $x_i \in A$ for all i, so A is infinite. □

It is natural to wonder whether the set A constructed in Theorem 6 can be made to be in P. A weaker form of this question asks whether every infinite set in P has an infinite P-printable subset. While this is still undetermined, the following theorem, pointed out to the first author by David Russo, shows that the sets in NP have similar structure to the sets in P in this regard.

THEOREM 7. *Every infinite set in* P *has an infinite* P-*printable subset if and only if every infinite set in* NP *has an infinite* P-*printable subset.*

Proof. Assume that every infinite set in P has an infinite P-printable subset, and let L be an infinite set in NP. Thus there is an infinite set A in P such that $L = \{x \mid \exists y, x\#y \in A\}$. Without loss of generality, assume that there is some k such that $x\#y \in A \Rightarrow |x\#y| = |x|^k$. By assumption, A has an infinite P-printable subset S, so the set $\{x \mid \exists y, x\#y \in S\}$ is an infinite P-printable subset of L. $\square$

Acknowledgments. The first author acknowledges the influence of stimulating dialogue with his thesis advisor Kim King, Klaus Ambos-Spies, Ronald Book, Jin-Yi Cai, Juris Hartmanis, Steve Mahaney, Larry Ruzzo, David Russo, Alan Selman, and Osamu Watanabe.

The second author acknowledges the invaluable assistance of his advisor Alan Selman, and also Peter van Emde Boas.

REFERENCES

[All85] E. ALLENDER, *Invertible functions*, Ph.D. thesis, Georgia Institute of Technology, Atlanta, GA, September 1985.

[All86a] ———, *Characterizations of* PUNC *and precomputation*, in Proc. 13th International Colloquium on Automata, Languages, and Programming, Springer-Verlag, Berlin, New York, 1986, pp. 1-10.

[All86b] ———, *The complexity of sparse sets in* P, in Proc. Conference on Structure in Complexity Theory, A. Selman, ed., Springer-Verlag, Berlin, New York, 1986, pp. 1-11.

[All87] ———, *Some consequences of the existence of pseudorandom generators*, in Proc. 19th Annual ACM Symposium on Theory of Computing, 1987, pp. 151-159.

[BB86] J. BALCÁZAR AND R. BOOK, *On generalized Kolmogorov complexity*, Acta Inform., 23 (1986), pp. 679-688.

[BCH84] P. BEAME, S. COOK, AND H. HOOVER, *Log depth circuits for division and related problems*, in Proc. 25th IEEE Symposium on Foundations of Computer Science, 1984, pp. 1-11.

[BDG85] J. BALCÁZAR, J. DIAZ, AND J. GABARRÓ, *Uniform characterizations of non-uniform complexity measures*, Information and Control, 67 (1985), pp. 53-69.

[Ber77] L. BERMAN, *Polynomial reducibilities and complete sets*, Ph.D. thesis, Cornell University, Ithaca, NY, 1977.

[Bra77a] F.-J. BRANDENBERG, *The contextsensitivity bounds of contextsensitive grammars and languages*, in Proc. 4th International Colloquium on Automata, Languages, and Programming, Springer-Verlag, Berlin, New York, 1977, pp. 272-281.

[Bra77b] ———, *On one-way auxiliary pushdown automata*, in Proc. 3rd GI Conference, Springer-Verlag, Berlin, New York, pp. 133-144.

[Cha66] G. CHAITIN, *On the length of programs for computing finite binary sequences*, J. Assoc. Comput. Mach., 13 (1966), pp. 547-569.

[Cha75] ———, *A theory of program size formally identical to information theory*, J. Assoc. Comput. Mach., 22 (1975), pp. 329-340.

[Chy77] M. CHYTIL, *Comparison of the active visiting and the crossing complexities*, in Proc. 6th Conference on Mathematical Foundations of Computer Science, Springer-Verlag, Berlin, New York, 1977, pp. 272-281.

[Coo71] S. COOK, *Characterizations of pushdown machines in terms of time-bounded computers*, J. Assoc. Comput. Mach., 19 (1971), pp. 175-183.

[GS84] J. GROLLMANN AND A. SELMAN, *Complexity measures for public-key cryptosystems*, in Proc. 25th IEEE Symposium on Foundations of Computer Science, 1984, pp. 495-503.

[GS85] A. GOLDBERG AND M. SIPSER, *Compression and ranking*, in Proc. 17th Annual ACM Symposium on Theory of Computing, 1985, pp. 440-448.

[Har83] J. HARTMANIS, *Generalized Kolmogorov complexity and the structure of feasible computations*, in Proc. 24th IEEE Symposium on Foundations of Computer Science, 1983, pp. 439-445.

[Hem87] L. HEMACHANDRA, *Counting in structural complexity theory*, Ph.D. thesis, Cornell University, Ithaca, NY, 1987.

[HH86] J. HARTMANIS AND L. HEMACHANDRA, *On sparse oracles separating feasible complexity classes*, in Proc. 3rd Annual Symposium on Theoretical Aspects of Computer Science, Springer-Verlag, Berlin, New York, 1986, pp. 321-333.

[HIS85] J. HARTMANIS, N. IMMERMAN, AND V. SEWELSON, *Sparse sets in* NP-P: EXPTIME *versus* NEXPTIME, Inform. Control, 65 (1985), pp. 158-181.

[HU79] J. HOPCROFT AND J. ULLMAN, *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, Reading, MA, 1979.

[Huy85] D. HUYNH, *Non-uniform complexity and the randomness of certain complete languages*, Tech. Report TR 85-34, Iowa State University, Ames, IA, 1985.

[Huy86] ———, *Resource-bounded Kolmogorov complexity of hard languages*, in Proc. Conference on Structure in Complexity Theory, A. Selman, ed., Springer-Verlag, Berlin, New York, 1986, pp. 184-195.

[HY84] J. HARTMANIS AND Y. YESHA, *Computation times of* NP *sets of different densities*, Theoret. Comput. Sci., 34 (1984), pp. 17-32.

[IR86] O. IBARRA AND B. RAVIKUMAR, *On sparseness, ambiguity, and other decision problems for acceptors and transducers*, in Proc. 3rd Annual Symposium on Theoretical Aspects of Computer Science, Springer-Verlag, Berlin, New York, 1986, pp. 171-179.

[Ko86] K. KO, *On the notion of infinite pseudorandom sequences*, Theoret. Comput. Sci., 48 (1986), pp. 9-13.

[Kol65] A. KOLMOGOROV, *Three approaches for defining the concept of information quantity*, Problems Inform. Transmission, 1 (1965), pp. 1-7.

[KOSW86] K. KO, P. ORPONEN, U. SCHÖNING, AND O. WATANABE, *What is a hard instance of a computational problem?*, in Proc. Conference on Structure in Complexity Theory, A. Selman, ed., Springer-Verlag, Berlin, New York, 1986, pp. 197-217.

[Lon86] L. LONGPRÉ, *Resource bounded Kolmogorov complexity, a link between computational complexity and information theory*, Ph.D. thesis, Cornell University, Ithaca, NY, 1986.

[Mor82] S. MORAN, *On the accepting density hierarchy in* NP, SIAM J. Comput., 11 (1982), pp. 344-349.

[Orp86] P. ORPONEN, *The structure of polynomial complexity cores*, Ph.D. thesis, University of Helsinki, Helsinki, Finland, 1986.

[Rub86a] R. RUBINSTEIN, *Immunity for generalized Kolmogorov complexity classes*, Tech. Report NU-CCS-86-2, Northeastern University, Boston, MA, December 1986.

[Rub86b] ———, *A note on sets with small generalized Kolmogorov complexity*, Tech. Report TR 86-4, Iowa State University, Ames, IA, March 1986.

[Sip83] M. SIPSER, *A complexity theoretic approach to randomness*, in Proc. 15th ACM Symposium on Theory of Computing, 1983, pp. 330-335.

[Val76] L. VALIANT, *Relative complexity of checking and evaluating*, Inform. Process. Lett., 5 (1976), pp. 20-23.

[vzG84] J. VON ZUR GATHEN, *Parallel powering*, in Proc. 25th Annual ACM Symposium on Theory of Computing, 1984, pp. 31-36.

[WB79] G. WECHSUNG AND A. BRANDSTADT, *A relation between space, return and dual return complexities*, Theoret. Comput. Sci., 9 (1979), pp. 127-140.

[Wec80] G. WECHSUNG, *A note on the return complexity*, Elektron. Informationsverarb. Kybernet., 16 (1980), pp. 139-146.