DIMACS Technical Report 94-17
April 1994

Normal Numbers and Sources for BPP

by

Martin Strauss[1,2]
Department of Mathematics
Rutgers University
New Brunswick, NJ 08903

[1]Affiliated Graduate Student Member
[2]Research supported by NSF grant CCR-9204874.

DIMACS is a cooperative project of Rutgers University, Princeton University, AT&T Bell Laboratories and Bellcore.

DIMACS is an NSF Science and Technology Center, funded under contract STC–91–19999; and also receives support from the New Jersey Commission on Science and Technology.

ABSTRACT

In [L90], Lutz proposed a notion of *source*, a nonrandom sequence that can substitute in a certain way for the random bits used by bounded-error probabilistic machines. He showed that almost every sequence in DSPACE($2^{\text{polynomial}}$) is a source. We improve this abundance result to PSPACE, by first showing that the sources are exactly the classical normal numbers of Borel. We go on to show there are sources in AC^0. Unfortunately, this suggests that alternate notions of source should be explored.

1 Introduction

In [L90], Lutz examines a particular kind of pseudorandomness useful for simulating the bounded-error probabilistic machines. The pseudorandomness is not in the form of a generator, that expands a short truly random string, but instead is a single computable sequence, called a source, whose elements can substitute for random bits in a certain repeated simulation of a bounded-error machine. Thus a source is a particular sequence that is "random enough" in a quantifiable way.

Lutz's work captures two intuitive properties of sources. Intuitively, sources are *universal,* i.e., a single source should work for all machines and all inputs, and sources are *abundant*, i.e., almost all sequences should be sources. Universality is built in to the definition of source (see below), while abundance is at a particular level of resource boundedness – almost every sequence in $\mathrm{E_2SPACE} = \mathrm{DSPACE}(2^{\mathrm{polynomial}})$ is shown to be a source. Lutz also trades some universality for abundance, showing that for any one machine, almost all sequences in $\mathrm{ESPACE} = \mathrm{DSPACE}(2^{\mathrm{linear}})$ are, for all inputs, sources.

While Lutz was primarily interested in the BPP machines, he notes that the results hold for all bounded error machines, i.e., almost every sequence in $\mathrm{E_2SPACE}$ can replace a random sequence for bounded-error machines of arbitrary complexity. This suggests that the complexity of a sequence S has little to do with the complexity of languages having machines for which S is a source. We show that if a sequence is known to be a source for the constant-time bounded-error machines, then it will be a source for arbitrary bounded-error machines, since these sequences are exactly the classical normal sequences (or normal numbers) of [Bor]. The condition of normality proves easier to work with than the condition of being a source, and in this way we are able to improve the arguments of [L90], showing that almost all sequences in PSPACE are sources.

A third intuitive property of sources is hardness. There should be no sources in P unless P = BPP. Unfortunately, there are very simple normal numbers, and we go on to construct a normal number in AC^0. The existence of our AC^0 language does not make BPP languages any easier, and so the sources considered here are not hard by this criterion. Other definitions of source addressing this issue are explored in [AS].

2 Definitions

A *probabilistic machine* is a Turing machine with a usual input tape and another one-way, read-only input tape, called the random tape, each cell of which contains a 0 or a 1. For each input x and each sequence S on the random tape the machine halts. For each x we write $\Pr_M(x) = \sum 2^{-|s|}$, where the sum is over all strings s such that, in the course of deciding x with some (and therefore any) extension S of s on the random tape, M reads exactly the bits in s and then accepts x. A *bounded-error machine* is a probabilistic machine with the property that there's a real number $r > 0$ (a reliability of the machine) such that for each x,

$|\Pr_M(x) - 1/2| \geq r$. This machine is said to recognize the language $L(M)$ of words x such that $\Pr(x \in L) = \Pr_M(x) > 1/2$. The class of bounded-error machines will be denoted B.

We also consider a small class of machines, B1. A B1 machine ignores its input, flips some finite number of coins (the number of coins flipped may depend on the outcome of already-flipped coins, but is finite on all paths), and decides whether or not to accept its input (the 1 in B1 is supposed to stand for constant time: since a B1 machine ignores its input, it runs in constant time). The probability of acceptance is bounded away from 1/2 (the probability is exactly $a/2^b$, for some $a \neq 2^{b-1}$ and b equal to the maximum number of coins flipped). The class of languages accepted by such machines is $\{\emptyset, \{0,1\}^*\}$. Note that a B1 machine is a BPP machine, which in turn is a B machine, where BPP denotes the polynomial-time bounded-error *machines*.

For many of the machines in this paper, the input is either ignored or can be considered part of the machine. We are interested in how changes to the random tape affect the outcome of the computation. In these cases, it will be convenient to let "run M on s" mean "run M with s on the random tape."

For a sequence S, let S_N denote the first N bits of S. For a string s, let s_- denote the string formed by dropping the last bit of s, and let s' denote the string formed by changing the last bit of s. Finally, let $S[i..j]$ denote bits i through j of S.

We define source as in [L90].

Definition 1 *A* source *is a sequence S such that for all bounded-error machines M and all inputs x, for almost all N and all r' less than the reliability of M, if we run M on x repeatedly, using successive bits from S_N, then the fraction of correctly deciding runs is at least* $(1/2 + r')$.

That is, for a bounded-error machine M of reliability r and input x, let M' be the machine that repeatedly runs M on x and outputs a sequence $\langle a_i \rangle$ such that $a_i = 1$ if run number i of M correctly decides x and $a_i = 0$ otherwise. The machine M' is a probabilistic transducer whose output depends on coin flip outcomes, but M' uses randomness only to simulate M. A sequence S is a source if for all M and x we have

$$\liminf_n \frac{1}{n} \sum_{i=1}^{n} a_i \geq \frac{1}{2} + r.$$

Strictly speaking, this definition is stronger than in [L90], where the liminf must be greater than 1/2 but may be less than $1/2 + r$. In practice, however, proofs here and most proofs in [L90] that S is a source show that for any $r' \in (0, r)$ the fraction of correct runs of M on S is at least $1/2 + r'$, while proofs that S is not a source show that the fraction of correct runs is at most 1/2.

As in [L90], we will write SOURCE(B) for the set of all sources. We will also speak of sources for the bounded-error, input-ignoring machines, denoted SOURCE(B1), and the sources for the polynomial-time bounded-error machines will be denoted SOURCE(BPP). Note that a sequence is a source if it succeeds on every *machine* in some class, so we are letting the symbol BPP do double duty representing a class of languages and a class of machines. Let UB1, UBPP and UB denote the classes of B1, BPP and B machines, respectively, that, for each input x, always flip some number $m(x)$ random bits on each path of the computation. Note the languages recognized by UBPP and BPP machines coincide, and the languages in UB and B coincide (the latter are the recursive languages), and the conversion of a B machine to a UB machine involves small resource cost. Thus UB and UBPP might have developed as *the* classes of bounded-error machines, but historically B and BPP have been so designated instead.

We turn now from machines to sequences.

Definition 2 *An r-ary sequence is* normal *in the scale of r if each finite string s appears $r^{-|s|}N + o(N)$ times in S_N.*

(A string is said to "appear at position i in a sequence S" if $S_{i-|s|+j} = s_j$ for all $j = 1, 2, \ldots |s|$. The string appears k times if there are k different i such that the string appears at position i. Thus 00 appears three times in 0000.) If a string s appears k times in the finite sequence S_n, we will say "the density of s on S_n is k/n" (usually the term density is used for the asymptotic frequency of appearance of a finite string within an infinite string).

Normal sequences were first mentioned by Borel, and discussed in [Bor], where it is noted that all reals except on a set of measure zero are normal numbers, i.e., have binary expansions which are normal in the scale of 2. In [Ch], it is shown that the sequence

$$1\;10\;11\;100\;101\;110\;111\;1000\ldots,$$

formed by concatenating the binary numbers, is normal in the scale of 2, and in [CE] a criterion is given for a set $\{a_i\}$ of integers so that the concatenation of the a_i's is normal.

The concept of resource-bounded measure on exponential and larger time and space classes was introduced by Lutz, and we refer the reader to [L92] for relevant definitions. Extending the concept of measure downward to PSPACE or P involves some subtleties, and is accomplished in [M2] and [AS]. We will use the notation of [AS], which is convenient for talking about several different kinds of measure. For the reader familiar with resource-bounded measure in general, we recall that a measure on PSPACE is defined in [M2] by writing PSPACE as the result of plogon constructors. (Plogon stands for "polylogspace, online". Here and in [AS] plogon machines are called Φ(PSPACE) machines.) A Φ(PSPACE) machine works in polylogspace, has the allowable space marked off on its worktape, and reads its input once, from left to right. A measure on P was defined in [AS] by writing P as the result of Γ(P) constructors, where a Γ(P) machine M works in polylog time, is given the length of its input, and, for each integer n, there's a polylog-sized *dependency set* $G_{M,n}$, such that for each w of length n and each $i \in G_{M,n} \cup \{|w|\}$, M can, in polylog time, compute $M(w[1..i])$ querying only bits in $G_{M,n} \cap \{1, 2, \ldots i\}$.

A language L will be identified with its characteristic sequence χ_L: enumerate the strings, and set bit i of χ_L to 1 iff the i^{th} string is in L. Following [L90], we will say that a sequence is in a complexity class iff it is the characteristic sequence of a language in that class. Note that string s appears in position $1s - 1$ (prepend a 1 to s then subtract 1), starting with λ in position 0.

string	λ	0	1	00	01	...
postition + 1	1	10	11	100	101	...

In this paper we will only consider classes powerful enough to compute $s \mapsto 1s - 1$, so the above definition is equivalent to the following, which will be used here:

Definition 3 *$S \in \mathcal{C}$ if the language*

$$\{x : \text{ the } x^{\text{th}} \text{ bit of } S \text{ is } 1\}$$

is in $\mathcal{C}$.

In the next section we will show that a sequence is a source for B or B1 iff it is normal, thus characterizing the sources. In subsequent sections, we will show that almost all PSPACE sequences are normal, and finally, that there's a normal sequence in DLOGTIME-uniform AC^0, which implies there are uncountably many normal sequences in nonuniform AC^0.

3 SOURCE(B1) = NORMAL = SOURCE(B)

We will show that B1 sources are normal, then that normal sequences are sources for B. Since a B1 machine is a B machine, all sources for B are sources for B1. In particular, this shows that a sequence is a source for B iff it is a source for any class of bounded error machines between B1 and B, e.g., BPP.

Theorem 4 *Any source for* B1 *is normal.*

Proof. Suppose S is not normal. Then there's a string s with asymptotic density bounded away from $2^{-|s|}$. We may assume s is of minimal length; so in particular s_- has density that is asymptotically uniform. Let $\epsilon \in (0, 1/2)$ be such that the relative frequency of s to s_- is at least $1/2 + \epsilon$ infinitely often (or is at most $1/2 - \epsilon$, wlog assume the former). We will construct a B1 machine that detects this bias.

First we consider the case $s_- \neq \lambda$. Choose η' small enough that for all $\eta < \eta'$ we have $(1 - \eta)(1/2 + \epsilon) > 1/2$. Choose $N' > |s|$ large enough that for all $N > N'$

$$\text{the density of } s_- \text{ is at least } \tfrac{1}{2}2^{-|s_-|} \text{ on } S_N, \tag{1}$$

and

$$\frac{1 - \eta + |s|}{\eta N + (1 - \eta)|s|} < \frac{1}{2}2^{-|s_-|}.$$

The machine M is defined as follows: Read random bits until either s or s' appears, but not more than N random bits. If s appears before s' accept, if s' appears before s reject (and read no more bits in these cases), and if neither s nor s' appears in N random bits reject (but read all N bits even if after reading $N - |s| + 1$ bits it is known that neither s nor s' will appear).

As a probabilistic machine, M accepts with probability strictly less than $1/2$, so M is a bounded-error machine accepting $\emptyset$. Also, note that M ignores its input. Thus M is a B1 machine. We will check the performance of M run on S, verifying that M behaves differently on S than on random strings. First note that when M is run on S_n, at least $(1-\eta)$ of the runs see s or s'. Otherwise, suppose that of T runs, only $(1-a)T < (1-\eta)T$ of the runs see s or s'. These runs use at least $(1-a)|s|T$ bits, and the other runs use aNT bits, so at least $(aN + (1-a)|s|)T > (\eta N + (1-\eta)|s|)T$ bits are used altogether. The string s_- appears at most $(1-\eta)T$ times among appearances seen by a run of M, and at most $|s|T$ times in appearances spanning two runs of M. This would give the overall asymptotic density of s_- at most

$$\frac{1-\eta+|s|}{\eta N + (1-\eta)|s|} < \frac{1}{2}2^{-|s_-|},$$

contradicting (1). Next, of the runs that see s_-, at least $(1/2+\epsilon)$ see s, and so these $(1-\eta)(1/2+\epsilon) > 1/2$ runs accept, while fewer than $1/2$ of the runs reject. Thus S causes M to accept $\{0,1\}^*$ i.o., i.e., for infinitely many n, the machine M accepts $\{0,1\}^*$ in at least $1/2$ of the runs on S_n. Therefore S is not a source.

If $s_- = \lambda$, then the density of 0 (wlog) is greater than $1/2+\epsilon$ infinitely often. Pick $N > -\log\epsilon$. By the pigeonhole principle, there is some $i \in [0, N]$ such that, for an infinite set A of n's, the density of 0 in positions less than n and congruent to i mod $(N+1)$ is at least $1/2+\epsilon$. Of the 2^N strings of length N, there is some string σ, so that for infinitely many $n \in A$, the density on S_n of the two (2^{N+1})-digits formed by inserting 0 or 1 into σ at position i is at most $2^{-N} < \epsilon$. The machine M will read $N+1$ random bits. If the string formed by removing the bit in position i is σ, reject. Otherwise, if the bit in position i is 0 accept, and if the bit in position i is 1 reject (the machine M reads all $N+1$ bits even if $i < N+1$ and the outcome is determined after reading position i).

As a probabilistic machine, M rejects with probability

$$1/2(1-2^{-N}) + 1 \cdot 2^{-N} = 1/2 + 2^{-(N+1)},$$

so M is a bounded error machine accepting $\emptyset$. On the other hand, as a deterministic machine using S instead of random bits, M accepts with probability

$$(1/2+\epsilon)(1-2^{-N}) > 1/2 + \epsilon/2 - \epsilon^2 > 1/2$$

i.o., i.e., for infinitely many n, M accepts $\{0,1\}^*$ more than half the time when run on S_n, so S is not a source. ∎

Before proving the next inclusion, we need some lemmas:

Lemma 5 *If $S = \langle b_0, b_1 \ldots \rangle$ is normal in the scale of 2, then for all m the sequence*

$$\langle b_0 b_1 \ldots b_{m-1}, b_m b_{m+1} \ldots b_{2m-1}, \ldots \rangle$$

is normal in the scale of 2^m.

Here we regard S as a sequence of digits in base 2^m by the natural grouping of bits.
Proof. See [NZ], or [K, Theorem C] for a "simplified" proof of a statement more general than the lemma. ∎

Corollary 6 *Any normal sequence S is a source for* UB.

Proof. Fix a machine M of reliability r, and an input x, and suppose M on x always flips m bits. Suppose we are given $r' < r$. By lemma 5, pick N' large enough that for all $N > N'$, every 2^m-digit appears with density between $2^{-m}\left(\frac{1/2+r'}{1/2+r}\right)$ and $2^{-m}\left(\frac{1/2-r'}{1/2-r}\right)$ in S_N. Then, since M succeeds on $(1/2+r)2^m$ of the 2^m sequences of length m, we have that M succeeds on at least $(1/2+r')$ of the runs in S_N, and similarly, M fails on at most $(1/2-r')$ of the runs. ∎

As we noted above, languages accepted by B machines are accepted by a UB machines with little time penalty, so we might reasonably stop here. The rest of this section is devoted to showing that normal numbers are sources for B machines.

Lemma 7 *For all machines M (with reliability r), all inputs x, and all $\eta > 0, \epsilon > 0$ there's an N such that if M is run repeatedly on x through at least N random bits, the probability is less than η that M will decide correctly on fewer than $(1/2 + r - \epsilon)$ of the runs.*

Proof. If we run M through N complete runs rather than through N random bits, the conclusion follows from the law of large numbers. A proof of the lemma is only slightly more subtle. See [F, equation 6.4]. ∎

Now to prove the next inclusion:

Theorem 8 *Any normal sequence is a source for* B.

Proof. Fix a normal sequence S, a machine M of reliability $r > 0$, an input x, and suppose that M flips between m_1 and m_2 coins on any path of computation. Suppose $r' < r$; we will show that for large enough K, if M is repeatedly run on x using random bits from S_K, at least $(1/2 + r')$ of the runs will be correct. Put

$$R = \left(\frac{1/2 + r'}{1/2 + r}\right)^{1/3} < 1,$$

so it is sufficient to show at least $(1/2+r)R^3$ runs succeed using bits of S. Pick B', C' small enough that for all $B < B'$ and $C < C'$,

$$\begin{cases} (1-B) & > \quad R \\ (1-2Cm_2^2/m_1) & > \quad R \end{cases}$$

Next, by lemma 7, pick n' so large that for all $n > n'$

$$\Pr \begin{pmatrix} \text{on runs through at least } n \text{ random bits, less than} \\ \text{the fraction } (1/2+r)(1-B) \text{ of runs are correct} \end{pmatrix} < C \tag{2}$$

and

$$\left(1 - \frac{4Cm_2^3 + m_1 m_2}{Rnm_1}\right) > R.$$

Finally, by lemma 5, pick K' large enough so that for $K > K'$, among S_K each (2^{n+2m_2})-digit occurs with density at most $2 \cdot 2^{-(n+2m_2)}$, i.e., at most

$$2 \cdot 2^{-(n+2m_2)} \frac{K}{n+2m_2} \tag{3}$$

times among the (2^{n+2m_2})-digits in S_K, of which there are $K/(n+2m_2)$.

When M is run repeatedly on S_K, some runs of M span two consecutive (2^{n+2m_2})-digits of S_K, while others use bits entirely within a single (2^{n+2m_2})-digit. For any (2^{n+2m_2})-digit d in S_K, let the *interior* of d denote the maximal infix substring s of d such that some run of M on S_K starts on the first bit of s, and some later run of M on S_K ends on the last bit of s. Note that the interior of any (2^{n+2m_2})-digit is at least n bits long.

Call a string of length at least n "good" if it causes M to run correctly at least $(\frac{1}{2}+r)(1-B)$ of the time, otherwise call it "bad." For each $i \in [0, m_2)$, there are at most $C2^{n+2m_2}$ strings of length $n+2m_2$ containing a bad string starting in position i, by (2). Thus there are at most $m_2 C 2^{n+2m_2}$ strings of length $n+2m_2$ containing bad strings starting in any position $i < m_2$. By (3), each of these strings of length $n+2m_2$ appears with density at most $2 \cdot 2^{-(n+2m_2)}$ among the (2^{n+2m_2})-digits of S_K, so at most $2Cm_2 \frac{K}{n+2m_2}$ of the interiors of S_K are bad, the other $(1-2Cm_2)\frac{K}{n+2m_2}$ are good.

The ratio of correct to total runs on the good interiors is at least $(\frac{1}{2}+r)(1-B)$. The good interiors account for at least $\frac{K}{n+2m_2}\frac{n}{a}$ runs, where $a \le m_2$ is the average on the good interiors of the number of bits used per run. Besides these runs, there are at most $\frac{K}{n+2m_2}$ runs spanning boundaries of consecutive (2^{n+2m_2})-digits of S_K, and and at most $2Cm_2\frac{K}{n+2m_2}\frac{n+2m_2}{m_1}$ runs on the $2Cm_2\frac{K}{n+2m_2}$ bad interiors (which have length at most $n+2m_2$ and so account for at most $\frac{n+2m_2}{m_1}$ runs each).

The correct-to-total ratio on all S_K is at least the correct-to-total ratio on the good interiors times the ratio of runs on good interiors to total runs. That is, cancelling $\frac{K}{n+2m_2}$ in

(4), using $\frac{1}{1+x} > 1 - x$ in (5), and using $a \le m_2$ in (6), the ratio of correct runs to total runs on S_K is at least

$$
\begin{aligned}
\left(\frac{1}{2}+r\right)(1-B)\frac{\frac{n}{a}}{\frac{n}{a}+\frac{2Cm_2(n+2m_2)}{m_1}+1} \qquad (4) \\
> \left(\frac{1}{2}+r\right) R \frac{1}{1+\frac{2Cam_2(n+2m_2)}{nm_1}+\frac{a}{n}} \\
> \left(\frac{1}{2}+r\right) R \left(1-\frac{2Cam_2(n+2m_2)}{nm_1}-\frac{a}{n}\right) \qquad (5) \\
> \left(\frac{1}{2}+r\right) R \left(1-\frac{2Cm_2^2(n+2m_2)}{nm_1}-\frac{m_2}{n}\right) \qquad (6) \\
= \left(\frac{1}{2}+r\right) R \left(1-\frac{2Cm_2^2}{m_1}-\frac{4Cm_2^3+m_1m_2}{nm_1}\right) \\
> \left(\frac{1}{2}+r\right) R \left(R-\frac{4Cm_2^3+m_1m_2}{nm_1}\right) \\
= \left(\frac{1}{2}+r\right) R^2 \left(1-\frac{4Cm_2^3+m_1m_2}{Rnm_1}\right) \\
> \left(\frac{1}{2}+r\right) R^3 \\
= \left(\frac{1}{2}+r'\right).
\end{aligned}
$$

∎

Since a B1 machine is a B machine, we have, for any class B′ with B1 ⊆ B′ ⊆ B (e.g., B′ = BPP),

Corollary 9 *The following are equivalent:*

- *The sequence S is a source for* B.
- *The sequence S is a source for* B′.
- *The sequence S is normal.*

∎

4 Abundance of Normal Numbers

It is easily shown that (in the sense of Lebesgue measure) almost every real number between 0 and 1 is normal, i.e., has a normal binary expansion (see [Bor]; the fact also follows from a proof that almost every number is normal in a resource bounded sense). Lutz has shown [L90] that almost every E_2SPACE sequence is a source for B, (in the sense of measure defined there). We now show that almost every PSPACE sequence is normal (in the sense of measure in [M]), thereby concluding that almost every PSPACE sequence is a source. We also show it is not the case that almost every P sequence is normal (in the sense of measure in [AS]).

First, we cover the abnormal sequences in PSPACE. The strategy is to cover the set of abnormal sequences by writing this as a Φ(PSPACE)-union of more manageable pieces.

Definition 10 *For a string s and real number ϵ, let s' denote the string formed by changing the last bit of s. A sequence S is (s,ϵ)-*terminally-upper-seminormal *((s,ϵ)-tusn for short) if for almost every N, the ratio of appearances of s to appearances of s_- among the first N bits of S (we'll write $s :_N s_-$) is at most $(\frac{1}{2}+\epsilon)$.*

Lemma 11 *For each (s,ϵ), almost every* PSPACE *language is (s,ϵ)-tusn.*

Proof. Define a cover d for the non-(s,ϵ)-tusn languages by

$$d(w) = \begin{cases} (1+2\epsilon)d(w_-); & s \text{ is a suffix of } w \\ (1-2\epsilon)d(w_-); & s' \text{ is a suffix of } w \\ d(w_-); & \text{otherwise.} \end{cases}$$

Suppose for some w we have $s :_{|w|} s_- > (\frac{1}{2}+\epsilon)$. Then

$$\begin{aligned} & d(w)/d(\lambda) \\ = \;& (1+2\epsilon)^{(s:_{|w|}s_-)|w|}(1-2\epsilon)^{(1-(s:_{|w|}s_-))|w|} \\ > \;& (1+2\epsilon)^{(\frac{1}{2}+\epsilon)|w|}(1-2\epsilon)^{(\frac{1}{2}-\epsilon)|w|} \\ = \;& \left[(\frac{1}{2}+\epsilon)^{(\frac{1}{2}+\epsilon)}(\frac{1}{2}-\epsilon)^{(\frac{1}{2}-\epsilon)}\right]^{|w|} 2^{|w|} \\ = \;& 2^{\log\left(\left[(\frac{1}{2}+\epsilon)^{(\frac{1}{2}+\epsilon)}(\frac{1}{2}-\epsilon)^{(\frac{1}{2}-\epsilon)}\right]^{|w|}2^{|w|}\right)} \\ = \;& 2^{|w|\left(1-h\left(\frac{1}{2}+\epsilon\right)\right)}, \end{aligned}$$

where $h(x)$ is the entropy function, strictly less than 1 for $x \neq \frac{1}{2}$. Thus if S is a sequence with infinitely many prefixes w such that $s :_{|w|} s_- > (\frac{1}{2}+\epsilon)$, then we can find such a w of length at least $-\log(d(\lambda))/(1-h(\frac{1}{2}+\epsilon))$, and for that w we have $d(w) \geq 1$.

We have to show this density function is Φ(PSPACE)-computable. There are two issues: First, as the algorithm was given above we need $d(w_-)$ in order to compute $d(w)$, but a Φ(PSPACE) algorithm must do this bottom-up. Also, we need to be able to do the necessary arithmetic in polylogspace.

As we scan the input w, having just read a prefix $z \sqsubseteq w$, keep track of three things: the number of occurrences of s and s', the last $|s|-1$ bits of z, and $d(z)$. The number of occurrences of s and s' is at most $|w|$. Storing the last $|s|-1$ bits of z takes $|s|$ space (s is constant in this lemma, but in the following theorem s will be input). We can record $d(z)$ by recording two exponents, a and b, such that $d(z) = (1+2\epsilon)^a(1-2\epsilon)^b$. Each of a and b is at most $|w|$, so we can store these in $\log|w|$ space.

Finally, we need to compute, in polylog($|w|$) space, $d(w) = (1+2\epsilon)^a(1-2\epsilon)^b$ but only for the final a and b. Note $a, b \sim |w|$, so $d(w)$ will have about $|w|$ bits, too many to write down all at once, but the model of computation requires that we spit out the output bits in order without seeing them again. We may assume that ϵ is a power of 2, say $2\epsilon = 2^{-k}$. We can write

$$d(w) = \left(\sum_{i=0}^{a}\binom{a}{i}2^{-ik}\right)\left(\sum_{i=0}^{b}\binom{b}{i}(-2)^{-ik}\right).$$

Perform the operations without storing intermediate results, but rather restarting the computation for each required bit in an intermediate result. Note that no intermediate result is bigger than $|w|!$, which has $\sim |w|\log|w|$ bits, and so needs less than $2\log|w|$ space for a register. ∎

Theorem 12 *Almost every* PSPACE *sequence is normal, and so a source.*

Proof. Let $X_{s,j,+}$ denote the non-$(s, 2^{-j})$-tusn languages, and let $X_{s,j,-}$ the non-$(s, 2^{-j})$-terminally-lower-seminormal languages (an (s,ϵ)-tlsn language may be defined similarly to an (s,ϵ)-tusn language, or simply as the complement of an (s',ϵ)-tusn language). Each $X_{s,j,\pm}$ is null, by the lemma. By examining the proof of the lemma carefully, we see the cover for $X_{s,j,\pm}$ constructed there can be computed in space polynomial in (the values of) s and j (and the third subscript). Thus $X = \bigcup X_{s,j,\pm}$ is a Φ(PSPACE)-union of Φ(PSPACE)-null sets, and is null.

It remains to show that X contains the abnormal sequences. Suppose S is not normal. Then there's a string s and an $\epsilon > 0$ such that for infinitely many N, s appears at least $(2^{|s|}+\epsilon)N$ times (or at most $(2^{|s|}-\epsilon)N$ times; wlog assume the former) in S_N. We may assume s is the shortest string for which there exists ϵ so that this holds, so s_- appears close to $2^{|s_-|}N$ times in S_N for almost all N. Thus $s :_N s_- > \frac{1}{2} + \epsilon 2^{1-|s|}$ i.o., and $S \in X_{s,j,+} \subset X$ for $j > |s| - 1 - \log\epsilon$. ∎

We now make a small digression from probabilistic machines.

Definition 13 *An* absolutely normal *number is a real number x such that for all natural numbers b, the b-ary expansion of x is a sequence normal in the scale of b.*

Constructions for absolutely normal numbers are known (see [Lev], which cites [Leb] and [S]), but no "natural" absolutely normal number is known (e.g., it is unknown if $\sqrt{2}$ is absolutely normal). On the other hand, a random number in $[0,1]$ is absolutely normal with probability 1: the set of absolutely normal numbers is the (countable) intersection over b of the sets of numbers normal to base b, and so has Lebesgue measure 1. We note below that the measure argument can be carried out in Φ(PSPACE) measure, so we have a construction of an absolutely normal number, in which the n^{th} digit can be computed from n in space polynomial in n.

Theorem 14 *Almost every sequence in* PSPACE *is the binary expansion of an absolutely normal number.*

Proof. Modify the above proof to show that for arbitrary base b, almost every sequence in PSPACE is normal to the base b. The computation can be done in space polynomial in b so we can write the non-absolutely normal numbers as a Φ(PSPACE)-union of Φ(PSPACE)-null sets. ∎

We close this section by noting that the set of normal numbers is not measurable in P. In other words, it is *not* the case that almost every language in P is normal (respectively abnormal). In a sense, a Γ(P) machine looks only at $o(n)$ of its input, and normality is a condition on all but $o(n)$ of the bits [Ch]. The proof is straightforward given the discussion in [AS] about a similar theorem.

Theorem 15 *Let X be the set of abnormal (respectively normal) numbers. For the measure $\mu_{\Gamma(\mathrm{P})}(\cdot \mid \mathrm{P})$ of [AS], we have $\mu_{\Gamma(\mathrm{P})}(X \mid \mathrm{P}) \neq 0$.*

Proof. Suppose we are given a Γ(P) null density-system d; we construct a characteristic sequence w for an abnormal (normal) language P. We will assume d is its own computation; the general case is similar.

On input x, compute the dependency set $G = G_{d,2^{2|x|}}$. If $\mathrm{pos}(x) \in G$ then for increasing $i \in G \cap [1..\mathrm{pos}(x)]$, determine the choice of $w[i] \in \{0,1\}$ that minimizes $d(w[1..i])$. For other i, put $w[i] = 0$. (For other i, set $w[i]$ according to Champernowne's normal number, respectively.)

This algorithm is in P by hypothesis about the size of the dependency set. Since we follow a path of non-increasing d whenever d queries its input, the language accepted is not covered by d (the conclusion makes use of the "non-increasing" property in [AS]). It is clear that the language constructed is abnormal (normal). ∎

5 An Easily-Computed Normal Sequence

In this section we construct a normal sequence in DLOGTIME-uniform AC^0.

Let s_r denote the concatenation of the strings of length r, so for example $s_2 = 00\,01\,10\,11$ (we will say that s_r is the concatenation of the padded numerals of length r). In [Ch] it is shown that the decimal version of the sequence

$$S = s_1 s_2 s_3 \ldots = 0\,1\,00\,01\,10\,11\,000\ldots,$$

Figure 1: Program to compute S'

for $j = 1$ to ∞
  if j is a power of 2
    list the numerals of length j
  else
    {
      let j' be the largest power of 2 less than $\frac{j}{2}$
      for $i = 1$ to $\frac{j2^j}{j'2^{j'}}$
        list the numerals of length j'
    }

Figure 2: Comparison of S and S'

Position																
		$0 = g_0$	$2 = g_1$				$10 = g_2$				$34 = g_3$			$98 = g_4$	
		$\downarrow$	$\downarrow$				$\downarrow$				$\downarrow$			$\downarrow$	
S	$=$	0	1	00	01	10	11	000	001	$\ldots$	111	0000	$\ldots$	1111	00000	00001
	$=$	$s_1s_2s_3s_4s_5s_6s_7s_8\ldots$														
S'	$=$	0	1	00	01	10	11	0 1 0 1 0 1		$\ldots$	1 0 1	0000	$\ldots$	1111	00 01 10 11 00	
	$=$	$s_1s_2s_1^{12}s_4s_2^{20}s_2^{48}s_2^{112}s_8\ldots$														

formed by concatenating the strings, is normal. While this sequence is easy to describe, we have no proof that it's in AC^0. Moreover, a straightforward way of computing it requires division by small numbers, and division by 3 is known not to be in AC^0 (see [BL]). Thus we use a variant, S', computed by the program in figure 1, and shown in part in figure 2. In the figure 2 we put

$$g_r = \sum_{j=1}^{r} |s_j| = \sum_{j=1}^{r} j2^j = 2(r-1)2^r + 2,$$

so there are g_r bits in the listing of all numerals of length at most r.

Note that j' of the program is a power of 2 and $2^{j-j'} > j - j' > j'$, so j' divides $2^{j-j'}$. Thus, for example, in the 24 bits from bit $10 = g_2$ to bit $33 = g_3 - 1$, S lists the numerals of length 3, while S' twelve times lists the numerals of length 1 (1 is the greatest power of 2 less than 3/2). The numeral 0000 starts at bit $34 = g_3$ in both S and S'. In general, S' is a concatenation of numerals with lengths a power of 2, and, for r a power of 2, the string 0^r first appears as a single numeral at position g_{r-1} in both S and S'. Note that from position g_{4k} on, S' lists numerals longer than k.

Before proving that S' is normal, we mention the following lemma, proved (for base 10) in [Ch, lemma (ii)], and stated using the notation there:

Lemma 16 *Any string of length ρ appears $2^{-\rho}x + o(|s_r|)$ times in the first x digits of s_r.* ■

Here $r \to \infty$, and x can assume any value less than $|s_r| = r2^r$.

Proposition 17 *The sequence S' is normal.*

Proof. Let s be a string of length ρ. First, only finitely-many of the s_r's in S' have $r < \rho$. For $r \geq \rho$, the density of s is perfect on any (complete) appearance of s_r, ignoring $o(|s_r|)$ appearances of s spanning the boundary of two consecutive numerals. Comparing the sequences S and S' we see there are at least $\Theta(r2^r)$ bits preceding any appearance of s_r, so by the lemma, the density of s is asymptotically correct on the last fragment of an s_r. ■

Proposition 18 *The sequence $S' \in$ AC^0.*

Proof. We need to show how to accept the language of x's such that bit x of S' is 1. First we describe an algorithm without regard to complexity, then show that the algorithm can be computed in DLOGTIME-uniform-AC^0.

Given x, find the largest k such that $g_{2^k-1} \leq x$. If $x - g_{2^k-1} < 2^k 2^{2^k}$ then bit x is in the first appearance of s_{2^k}. Divide $x - g_{2^k-1}$ by 2^k. Pad the quotient if necessary with leading zeros, and bit x of S' will be the bit of the padded quotient indexed by the remainder. If $x - g_{2^k-1} \geq 2^k 2^{2^k}$, then bit x falls within an appearance of $s_{2^{k-1}}$ following the first appearance of s_{2^k}. Divide $x - g_{2^k-1} - 2^k 2^{2^k}$ by 2^{k-1}; the lowest 2^{k-1} bits of the padded quotient is a string such that bit x of S' will be the bit indexed by the remainder.

Note that $k \sim \log\log x$ is truly small. In AC^0 we can perform addition (and therefore subtraction and comparison) of numbers of linear size, and shift left and right by a linear number of bits. Computation of $g_{2^k-1} = (2^k - 2)2^{2^k}$ involves nothing more than these operations. We can search all $k \leq \log\log x$ to find the maximal k with $g_{2^k-1} \leq x$. The divisions and remainders mentioned can all be done by bit shifts, as can be the final test of a specified bit of a specified string, for bit position up to $2^k \sim \log x$. ■

6 Conclusion

We've shown that the sources, as defined, are exactly the normal numbers, and so there's a source in AC^0. This result does not, however, make BPP problems easier. We consider this is a drawback in the definition of source. Intuitively, we feel that a notion of source should make the complexity of a source at least the complexity of the languages decidable using the source. [AS] explores other possible notions of source.

Acknowledgment

I gratefully acknowledge helpful discussions with Eric Allender.

References

[AS] E. Allender and M. Strauss, "Measure on Small Complexity Classes, with Applications for BPP." DIMACS Technical Report 94-18, (1994).

[BDG] J. Balcázar, J. Díaz and J. Gabarró, *Structural Complexity I,* Springer-Verlag EATCS Monographs on Theoretical Computer Science, (1988), New York, pp. 130-141.

[BL] Ravi Boppana and Jeff Lagarias, "One-Way Functions and Circuit Complexity." *Information and Computation* **74**, (1987), pp. 226-240.

[Bor] E. Borel, *Leçons sur la théorie des fonctions,* (1928), pp. 182-216.

[Ch] D. G. Champernowne, "The Construction of Decimals Normal in the Scale of Ten." *J. London Math Soc.,* **8** (1933), pp. 254-260.

[CE] A. Copeland and P. Erdos, "Note on Normal Numbers." *Bull. AMS* (1946), pp. 857-860.

[F] W. Feller, *An Introduction to Probability Theory and Its Applications,* Wiley, (1968), New York, pp. 320-322.

[K] D. Knuth, *The Art of Computer Programming v. 2,* Addison-Wesley, (1981), Reading, Mass, pp. 149-151.

[L90] J. Lutz, "Pseudorandom Sources for BPP." *J. Computer and System Sciences,* **41** (1990), pp. 307-320.

[L92] J. Lutz, "Almost Everywhere High Nonuniform Complexity." *Journal of Computer and System Sciences* **44** (1992), pp. 220-258.

[Leb] H. Lebesgue "Sur certains démonstrations d'existence." *Bull. Soc. Math. France* **45** (1917), pp. 132-144.

[Lev] M. B. Levin "On Absolutely Normal Numbers." *Moscow University Mathematics Bulletin* **34:**1 (1979), pp. 32-39.

[M] E. Mayordomo, PhD Thesis, to appear. See also [M2], in which a preliminary version of the PSPACE measure appears.

[M2] E. Mayordomo, Measuring in PSPACE, to appear in *Proc. International Meeting of Young Computer Scientists '92,* Lecture Notes in Computer Science.

[NZ] I. Niven and H. Zuckerman, "On the Definition of Normal Numbers." *Pacific J. Math,* **1** (1951), pp. 103-109.

[S] W. Sierpinski "Démonstration élémentaire du théorème du M. Borel sur Les nombres absolument normaux et determination effective d'un tel nombre." *Bull. Soc. Math. France* **45** (1917), pp. 127-132.