[BG] Bennett and Gill, Relative to a random oracle, P(A) $\neq$ NP(A) $\neq$ Co-NP(A) with probability 1, *SIAM J. Comput.* **10** (1981), 96-113.

[Bu] S. R. Buss, The Boolean formula value problem is in ALOGTIME, *Proc. 19th ACM Symposium on Theory of Computing*, 1987, pp. 123–131.

[JL] D. Juedes and J. Lutz, The complexity and distribution of hard problems, *Proc. 34th FOCS Conference*, pages 177–185, 1993.

[KM] S. Kautz and P. Milterson, Relative to a random oracle, NP is not small, *Proc. 9th Structure in Complexity Theory Conference*, to appear, 1994.

[L90a] J. Lutz, Category and measure in complexity classes, *SIAM J. Comput* **19** (1990), 1100–1131.

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

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

[L93] J. Lutz, A Pseudorandom Oracle Characterization of BPP, *SIAM J. Comput.,* **22** 1993, 1075-1086.

[L93a] J. Lutz, The quantitative structure of exponential time, *Proc. 8th Structure in Complexity Theory Conference*, pages 158–175, 1993.

[LM] J. Lutz and E. Mayordomo, Measure, stochasticity, and the density of hard languages, to appear in *SIAM J. Comput.* (See also Proc. 10th STACS, Lecture Notes in Computer Science 665, 1993, pp. 38–47.

[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*, Topics in Computer Science series, Gordon and Breach.

[NW] N. Nisan and A. Widgerson, Hardness vs. Randomness, *Proc. 29th Annual IEEE Symp. on Foundations of Computer Science*, (1988), 2-11.

[RST] W. Ruzzo, J. Simon, and M. Tompa, Space-bounded hierarchies and probabilistic computation, *J. Comput. and System Sci.* **28** (1984) 216–230.

[S] M. Strauss, Normal numbers and sources for BPP, DIMACS Technical Report 94-17, 1994.

A relative to which P^A would have measure zero in P^A, and thus this does not constitute a meaningful notion of measure on P^A. On the other hand, if one adopts the convention[4] that a DTIME($\log^{O(1)}$) machine is permitted to write only queries of length $\log^{O(1)} n$ on its query tape, then it is straightforward to show that one obtains a measure on P^A. Similar observations hold for the measures on other subexponential time classes and on PSPACE.

5.2 Summary

Lutz's resource-bounded measure forms the basis for a large and growing body of interesting work. A limitation of Lutz's notion of measure is that it does not apply to P and other important subexponential time classes. We have remedied that situation by providing a notion of measure that does apply to these classes, and we have used this measure to show, among other things, that almost all sets in E_ϵ are hard for BPP, substantially improving a result of [L93].

It is worth noting that Lutz's definitions of resource-bounded measure have evolved somewhat over the years. Similarly, we may expect that as experience is gained, alternative formulations of measure on small time classes may arise. However, we have established that interesting results can be obtained with our current notion of measure, and we look forward to further work in this area.

Acknowledgments

We gratefully acknowledge helpful discussions and correspondence with Lane Hemaspaandra, Elvira Mayordomo, Jack Lutz, and David Juedes.

References

[A90] E. Allender, Oracles vs proof techniques that do not relativize, *Proc. SIGAL International Symposium on Algorithms*, Lecture Notes in Computer Science 450, 1990, pp. 39–52.

[AS] E. Allender and M. Strauss, Towards a Measure for P, DIMACS Technical Report 94-14, 1994.

[AJ] C. Álvarez and B. Jenner, On Dlogtime and Polylogtime reductions, Proc. 11th STACS, Lecture Notes in Computer Science 775, 1994.

[Ba] J. Balcázar, Self-Reducibility, *Journal of Computer and System Sciences* **41** (1990) 367-388.

[4] In most other settings, allowing only short queries to the oracle does *not* provide a satisfactory notion of relativization, because one cannot reduce the set A to itself (because one cannot write the input on the query tape).

oracle are answered randomly). That is, A is not only hard for BPP, but it is hard because it "looks random." It follows from [BG] that almost all sequences are pseudorandom sources for BPP.

Theorem 16 *Almost every* $A \in$ ESPACE *is a pseudorandom sequence for* BPP.

Proof. Theorem 13 shows that for most A in PSPACE, the language $F(A) = \{u : 0^{2^{|u|}}u \in A\}$ has hardness $2^{\eta m}$. An essentially equivalent way of restating this is to say that for almost all A in PSPACE, for almost all m, the function $f : \Sigma^m \to \{0,1\}$ defined by $f(u) = \chi_A(0^{m^2+u})$ has hardness $2^{\eta m}$. Thus, informally, almost all tally sets in PSPACE look "random enough" to serve as sources for BPP. Since tally sets in PSPACE are essentially the same thing as sets in ESPACE, a direct argument patterned after the proof of Theorem 13 can be used to show that almost all sets in ESPACE are pseudorandom sequences for BPP. ∎

Theorem 16 is in some sense analogous to the main result of [L90], but note that in contrast to [L90], we can make a limited claim of optimality, in the following sense. Theorem 16 shows that there are sources in DTIME$\left(2^{2^{O(n)}}\right)$. If there is any source A in DTIME$(T(n))$ for some $T(n) = 2^{2^{o(n)}}$, then $T(A) \in$ DTIME $\left(2^{n^{o(1)}}\right)$, and hence simulating any n^k-time bounded BPP machine would require at most time $n^k 2^{(n^k)^{o(1)}}$, and thus BPP would have subexponential time complexity. (We should note that our ESPACE upper bound on sources can be improved to exponential-time bounded machines with O(1) alternations, but there is at this point still no satisfactory notion of resource-bounded measure for classes defined in terms other than deterministic time and space.)

5 Conclusions

5.1 Does this Relativize?

What is "relativized one-way polylogarithmic space"? One's initial response is likely to be that this is a ludicrous notion, and that attaching oracles to such limited automata would probably not give rise to a very meaningful class. If followed further, this line of reasoning suggests that non-relativizing results might be obtained by studying $\mu_{\Phi(\text{PSPACE})}$-measure, and similar observations apply to the similarly-limited $\mu_{\Gamma(\text{P})}$-measure. Might one be able to show that most sets in P are hard for BPP? Such a result would, of course, require nonrelativizing proof techniques, but if the notion of measure on P does not relativize in a meaningful way, then perhaps it could be used to obtain results of this sort.

Unfortunately, it turns out that these notions of measure *do* relativize in a natural way. When one is dealing with extremely weak models of computation, the question of how to provide access to the oracle is often rather controversial. (A survey of papers discussing the issues involved may be found in [A90].) It is easy to see that, for instance, if one were to use the so-called "Ruzzo-Simon-Tompa" relativization method [RST], then there are oracles

at least $2^{n-1}(1-\eta)$ of the words of length n, where the circuits are now allowed to have "oracle" gates that query the oracle A.

Fix positive $\epsilon < 1/3$. We show that for almost every $A \in \mathrm{E}$ we have $E^A \cap H_\epsilon^A \neq \emptyset$. We use a "fully-relativized" version of [NW, Theorem 3]:

Theorem 15 *If* $E^A \cap H_\epsilon^A \neq \emptyset$, *then* $\mathrm{BPP}^A = \mathrm{P}^A$.

The proof proceeds exactly as in Theorem 11, where now we simulate oracle circuits γ. [We know of no way to simulate these oracle circuits using small dependency sets (or one-way access to the input in the case of the Φ(PSPACE)-measure).]

∎

4 Pseudorandom Sources

In [L90], Lutz proposed a notion of *source* for BPP. He gave a criterion for a particular sequence to be useful as a substitute for the sequence of independent unbiased coin flips used by a BPP machine. Based on this work, we formulate three intuitive properties a notion of source should have:

Universality A single source should "work" for all BPP languages.

Abundance The set of sources should have measure 1 at some level of resource-boundedness. This implies that a random sequence of coin flips is a source with probability 1.

Hardness If the bits of a source can be obtained in polynomial time, then P = BPP.

The definition in [L90] captures the first two properties, but lacks the third, as one can construct sources in AC^0 without showing P = BPP [S].

We seek an alternate criterion for a particular computable sequence to be "random enough." We will capture universality, abundance, and hardness.

Intuitively, a sequence A is a pseudorandom sequence if it is possible to use A in place of a sequence of random bits to recognize each BPP language (where it is crucial to the definition that this simulation would work if A were replaced with a random sequence). Formally, we will say that a sequence $A = \langle a_i \rangle$ is a *pseudorandom sequence for* BPP if for each $L \in \mathrm{BPP}$, there is a bounded-error probabilistic polynomial-time machine accepting L such that, for each x, $M(x, A) = 1 \Leftrightarrow x \in L$, where "$M(x, A)$" denotes the result of running machine M on input x along the path given by taking the result of the i^{th} coin flip to be a_i. That is, on each input x, the first polynomially-many bits of the sequence A are used in place of probabilistic bits.

Stated another way, A is a pseudorandom sequence for BPP if the set $T(A) = \{0^i | a_i = 1\}$ is hard for BPP under $\leq_T^p$ reductions, where the machine M that reduces BPP language L to $T(A)$ also accepts L when M is viewed as a BPP machine (i.e., where queries to the

Figure 1: $\Phi(PSPACE)$ program to cover A for which $F(A)$ is easy

---

```
 d_n(w) :
a = 0
if |w| ≤ pos(0^{2^n} 0^n)
    return(2^{-2^{Θ(n)}})
if |w| = pos(0^{2^n} x) for |x| = n
    {
        for |z| = 2^n − x − 1        /* so |wz| = pos(0^{2^n} 1^n) */
            for |γ| ≤ 2^{εn}
                {
                    r = 0
                    for |u| = n
                        if (u is accepted by γ) iff (u ∈ F(C_wz))
                            r++
                    if  r ≥ 2^{n−1}(1 − 2^{−εn})
                        {
                            a++
                            break       /* found a circuit; consider next z */
                        }
                }
        return(a/2^{−(2^n−x−1)})
    }
if |w| ≥ pos(0^{2^n} x) for |x| = n
    {
        for |γ| ≤ 2^{εn}
            {
                r = 0
                for |u| = n
                    if (u is accepted by γ) iff (u ∈ F(C_wz))
                        r++
                if  r ≥ 2^{n−1}(1 − 2^{−εn})
                    return(1)
            }
        return(0)
    }
```

There are $2^{2^n} = 2^{\log^{1/b}|w|}$ extensions and $2^{2^{\eta n}}$ circuits to consider, and each simulation takes time $2^{\eta n}$. Thus the total time to do all simulations is less than $2^{\log^{\epsilon}|w|}$. The dependency set consists of the positions of all words of the form $0^{2^{b|v|}}v$, so the dependency bound is met also.

Finally, we apply the Borel-Cantelli-Lutz lemma to the density system d_n, using modulus $m(n) = \log n$. That is, putting $j = 2^n$,

$$\sum_{n>\log k} 2^{-2^n} \leq \sum_{j>k} 2^{-j} \leq 2^{-k},$$

so we conclude $\mu_{\Gamma(\mathrm{E}_\epsilon)}(\limsup X_n) = 0$. ∎

We turn now to space bounds.

Theorem 13 *For almost every* $A \in$ PSPACE *we have* $\mathrm{BPP} \subset \mathrm{P}^A$.

Proof.

The proof proceeds like the proof in the time-bounded case. We present the details for the $\mu_{\Phi(\mathrm{PSPACE})}$measure; the analysis is similar for the $\mu_{\Gamma(\mathrm{E}_\epsilon)}$ measure. Compared to the $\Gamma(E_\epsilon)$ proof, the Φ(PSPACE) proof can set $b = 1$ and $\eta = \epsilon$. Pseudocode is shown in figure 1.

The length of the input w is 2^{2^n}, and we have space polylogarithmic in $|w|$, i.e., 2^{cn} space to work with. The inside of the loops is a simulation of a circuit of size $2^{\epsilon n}$ with n inputs, which can be done in the allowed space. The loop indices take only 2^n space to write down, and the counters r and a (which count the number of words *r*ight and the number of languages *a*pproximated, respectively) grow to 2^n and 2^{2^n}, respectively, so take no more than 2^n space also.

It remains to show that the above algorithm for d_n can be computed by an online machine. This is immediate: For each n, the function d_n only needs to read the 2^n consecutive bits of input from $\mathrm{pos}(0^{2^n}0^n)$ through $\mathrm{pos}(0^{2^n}1^n)$ (or as many of these bits are actually present in the input). There is enough space to cache these bits when read in a single pass.

As in the $\Gamma(E_\epsilon)$ proof above, apply the Borel-Cantelli-Lutz lemma, and conclude that $F(A)$ is hard except for a set of A of measure 0. Using the [NW] generator we conclude that $\mathrm{BPP} \subset \mathrm{P}^A$ for almost every A in PSPACE. ∎

Note that any improvement in the time result would involve showing that each language in BPP has subexponential time complexity.

We also show that almost every set A in E satisfies $\mathrm{BPP}^A = \mathrm{P}^A$, improving the result of [L93] from ESPACE to E.

Theorem 14 *For almost every* $A \in \mathrm{E}, \mathrm{BPP}^A = \mathrm{P}^A$.

Proof. The proof of this theorem is essentially the same as the proof of Theorem 13; the only modification is to replace H_η in that proof with H_η^A: the set of languages L of "relativized" hardness $2^{\eta n}$, using oracle A. That is, for sufficiently large n any circuit of size $2^{\eta n}$ errs on

Fix positive ϵ. We show that for almost every $A \in \mathrm{E}_\epsilon$ and every $\eta < \min\{\epsilon, 1/3\}$ we have $E^A \cap H_\eta \neq \emptyset$. As in [L93], we describe for each oracle A, a language $F(A) \in E^A$. We show that the set of $A \in E_\epsilon$ such that $F(A) \notin H_\eta$ has zero measure. It follows from Theorem 12 that BPP $\subset \mathrm{P}^A$ for other A, and hence for almost all $A \in \mathrm{E}_\epsilon$:

Theorem 12 *If* $E^A \cap H_\eta \neq \emptyset$, *then* BPP $\subseteq \mathrm{P}^A$.

Proof. This is a "partially" relativized version of [NW, Theorem 3]. ∎

Let $0 < \eta < \epsilon$, and let $b > 1/\epsilon$. Let $F(A)$ denote the language

$$\{u : 0^{2^{b|u|}} u \in A\}.$$

Then $F(A) \in \mathrm{E}^A$. We will show that for almost all $A \in E_\epsilon$, $F(A)$ has hardness $2^{\eta n}$.

Fix a circuit γ with n inputs. For a word u of length n, the set $\{A : F(A) = \gamma \text{ on } u\}$ has (Lebesgue) measure 1/2. That is, if we choose A at random, the circuit $\gamma(u)$ is too small to query $0^{2^n} u \in A$, so with probability 1/2 we have $(u \in F(A))$ iff (γ accepts u). The events in $\{(u \in F(A)) \text{ iff } (\gamma \text{ accepts } u)\}_{|u|=n}$ are mutually independent, so we can apply the Chernoff bound to get

$$\{A : \gamma \neq F(A) \text{ on at most } 2^{n-1}(1 - 2^{-\eta n}) \text{ words}\}$$

has measure less than

$$e^{-(2^n - \eta n)^2/2^n} \leq 2^{-2^{n/3}}$$

for all sufficiently large n. Finally, the set

$$\begin{aligned} X_n &= \{A : \text{some } \gamma \text{ of size } 2^{\eta n} \text{ with } n \text{ inputs equals } F(A) \text{ on at least } 2^n - 2^{\eta n} \text{ words}\} \\ &= \{A : F(A)_{=n} \notin H_\epsilon(n)\} \end{aligned}$$

has measure at most

$$2^{2^{\eta n}} \leq 2^{2^{n/3}}$$

times the above, which is less than

$$2^{-2^{\Theta(n)}}.$$

The set $X = \limsup X_n$ contains the A for which $F(A) \notin H_\epsilon$ infinitely often, and we wish to show X is null.

The density function $d_n(w)$ is the conditional probability that a random A satisfies $F(A) \notin H_\eta$ given $A \in C_w$. For short w, i.e., $|w| \leq 2^{2^{bn}}$, the conditional probability is $2^{-2^{\Theta(n)}}$, independent of w, and this can be computed in the time allowed as a function of n. If $|w| \geq 2^{2^{bn}}$, then we will exhaustively generate all extensions of w corresponding to elements of $F(A)$ of length n, and for each such extension simulate all circuits of size $2^{\eta n}$ on all strings (inputs) of length n, and count the number of extensions that can be approximated in that way, to compute the conditional probability.

It should be observed that $\mu_{\Gamma(\mathcal{C})}(\mathrm{ODD}\,|\,\mathcal{C}) \neq 1$, since $C_0 \subseteq \overline{\mathrm{ODD}}$. Thus $\mathrm{ODD} \cap \mathcal{C}$ is not "measurable" according to $\Gamma(\mathcal{C})$.

We do not view this limitation of our measure as a drawback. Indeed, we conjecture that AC^0 and even $\mathrm{NTIME}(\log n)$ are *not* measure-zero subsets of P. Backing up our conjecture is the fact that, although $\mathrm{NTIME}(\log^{O(1)} n)$ is properly contained in $\mathrm{DTIME}(2^{\log^{O(1)} n})$, it is easy to show that if $\mathrm{NTIME}(\log^{O(1)} n)$ has measure zero in $\mathrm{DTIME}(2^{\log^{O(1)} n})$, then NP has measure zero in $\mathrm{DTIME}(2^{n^{O(1)}})$, which seems to be an unlikely consequence [KM, L93a]. Of course, proving that NP does *not* have measure zero in $\mathrm{DTIME}(2^{n^{O(1)}})$ entails proving that $\mathrm{P} \neq \mathrm{NP}$. However, we anticipate that due to the severely limited computational power of $\Gamma(\mathrm{P})$ machines, it should be possible to show that some interesting classes of sets are not $\Gamma(\mathrm{P})$-null. In [S], it is shown that what Lutz in [L90] called "pseudorandom sources for BPP" does *not* have measure one in P.

We note in passing that one can also define a measure analogous to our measure on time-bounded classes, using *space* bounds, as opposed to time bounds. In this way, one can obtain a measure $\mu_{\Gamma(\mathrm{PSPACE})}$ for PSPACE. Mayordomo has previously defined a notion of measure for PSPACE, where, instead of limiting the size of the dependency sets for the machines computing the density functions, instead she requires that the density functions be computed by polylogspace-bounded machines with one-way access to the input [M]. Let us denote her measure $\mu_{\Phi(\mathrm{PSPACE})}$. It is easy to show that $\mu_{\Phi(\mathrm{PSPACE})}(\mathrm{ODD}) = 0$, whereas the proof of Theorem 9 applies also to $\mu_{\Gamma(\mathrm{PSPACE})}$, and hence ODD is *not* $\Gamma(\mathrm{PSPACE})$-null. It is not clear if these two measures on PSPACE are incomparable, or if $\mu_{\Phi(\mathrm{PSPACE})}$ is a strictly richer notion. Our measure results on PSPACE in Section 3 hold for both the $\mu_{\Phi(\mathrm{PSPACE})}$ and $\mu_{\Gamma(\mathrm{PSPACE})}$ measures.

3 Hard Sets for BPP

It was shown in [BG] that for almost every A, $\mathrm{BPP}^A = \mathrm{P}^A$. In [L93] it was shown that almost every set in ESPACE has this property, and thus in particular almost every such set is hard for BPP. Note, on the other hand, that only a measure zero set of languages is hard for Σ_2^p (assuming $\mathrm{BPP} \neq \Sigma_2^p$), and thus the "reason" that a PSPACE-complete set is hard for BPP is in a fundamental way quite different from the "reason" that a random set is hard for BPP.

In this section, we improve the results of [L93] by showing that almost every set in PSPACE and in E_ϵ for $\epsilon > 0$ is hard for BPP, because almost every such set "looks random enough". As in [L93], we make use of the pseudorandom generators of [NW], although our construction differs from that of [L93] in several fundamental ways.

Theorem 11 *For almost every* $A \in \mathrm{E}_\epsilon$ *we have* $\mathrm{BPP} \subseteq \mathrm{P}^A$.

Proof. Define H_η to be the set of languages L of hardness $2^{\eta n}$. That is, for sufficiently large n any circuit of size $2^{\eta n}$ errs on at least $2^{n-1}(1-\eta)$ of the words of length n.

Theorem 9 *Let $\epsilon > 0$, and let $\mathcal{C}$ be a subexponential time complexity class. Then* ODD *is not $\Gamma(\mathcal{C})$-null, where* ODD *is defined to be $\{L$: for each n, $L^{=n}$ has an odd number of elements$\}$.*

Proof. For this sketch, assume that d is equal to the computation $\hat{d}$; the general case is proved in a similar manner.

We first note that $\Gamma(\mathcal{C})$-machines obey a non-increasing property. If the dependency set size for M is sublinear, then M ignores many bits of its input. In particular, if $M(w0)$ doesn't query the last bit of input then $M(w0) = M(w1)$, and, if M satisfies the average law (e.g., if M computes an exact computation of a density function), then $M(w0)$ and $M(w1)$ have common value at most $M(w)$. In general, it can be shown that if $M(w)$ doesn't query any bit from position i through position j, then "M is non-increasing on $[w_i, w_j]$," i.e.,$M(w') \leq M(w[1 \ldots i])$ for any w' with $w[1 \ldots i] \sqsubseteq w' \sqsubseteq w$.

Suppose $\mu_{\Gamma(\mathcal{C})}(\text{ODD}) = 0$, and let $d_k(w)$ be a null cover of ODD. We will construct an integer k and a language L in ODD not covered by d_k. Find l so that $f(2^{l'}) < 2^{l'}$ for $l' \geq l$, where f is the time bound for d. Put $n = \sum_{j=0}^{l-1} 2^j$, so n is the number of words of length less than l. Let w_1 be a prefix in ODD of length n and fix $k > n$; note that $d_k(w_1) < 1$. Let $w_2' \sqsupset w_1$ be of length $|w_1| + 2^l$, and such that for each $i \in [|w_1|, |w_2'|)$ bit i of $w_2' = 1$ iff $d_k(w_2'[1..i-1]1) \leq d_k(w_2'[1..i-1]0)$. Since d_k doesn't increase along this path, $d_k(w_2') < 1$. Since $f(2^l) < 2^l$, some bit i_2 corresponding to a word of length l is not queried by d_k; let $w_2 = w_2'$ except that bit i of w_2 is set so that w_2 is a prefix of ODD. Continue, forming w_3' of length $|w_2| + 2^{l+1}$, along the path of nonincreasing d_k, so that $d_k(w_3') < 1$. Some new bit i_3 corresponding to a word of length $l+1$ is not queried by d_k; put $w_3 = w_3'$ except that bit i_3 is set so that w_3 is a prefix of ODD. Continue in this manner constructing w_m for all $m \geq 2$. Finally, the language given by the limit of w_m is in ODD, but not covered by d_k by the non-increasing property above. ∎

Theorem 9 does *not* show that ODD has measure zero in $\mathcal{C}$, i.e., Theorem 9 leaves open the statement "almost every set in $\mathcal{C}$ is not ODD." The distinction is that Theorem 9 shows merely that every $\Gamma(\mathcal{C})$-null cover fails to cover some language in ODD, while the next theorem shows that any such null cover fails to cover some language in ODD $\cap\, \mathcal{C}$.

Theorem 10 *Let $\epsilon > 0$, and $\mathcal{C}$ be a subexponential time complexity class. Then*

$$\mu_{\Gamma(\mathcal{C})}(\text{ODD} \mid \mathcal{C}) \neq 0.$$

Proof. We will follow the strategy of the proof of Theorem 9, but make sure that the language L that we construct is in $\mathcal{C}$. Our algorithm for L is as follows. On input x, compute the dependency set $G_{d_k, 2^{2|x|}}$. Define the word w as follows: for each $i \in G_{d_k, 2^{2|x|}}$, determine the choice of $w_i \in \{0, 1\}$ that minimizes $d_k(w[1..i])$. Let the other bits of w be defined so that w is the lexicographically least string in ODD that is consistent with the bits defined thus far. Accept x iff bit $\text{pos}(x)$ of w is 1. It is easy to verify that this can be done in $\mathcal{C}$. ∎

2.4 Elementary Facts Concerning the Measure

We have established that our notion of measure satisfies the measure axioms, but the reader may wonder if enough sets are measurable for the measure to useful in any setting. Indeed, we show in this subsection that the class of P-measurable sets is an *extremely* limited class. However, the following theorem and its corollaries (and also the results of the next section) show that measure-zero results can be established for some important classes of sets.

Theorem 5 *Let* $\mathcal{C} = \mathrm{DTIME}(\mathcal{F})$. *If* $f \in \mathcal{F}$, *then* $\mu_{\Gamma(\mathcal{C})}(\mathrm{DTIME}(f)) = 0$.

Corollary 6 *For all* k, $\mu_{\Gamma(\mathrm{P})}(\mathrm{DTIME}(n^k)) = 0$.

Corollary 7 *Let* $0 < \delta < \epsilon$. *Then* $\mu_{\Gamma(\mathrm{E}_\epsilon)}(\mathrm{E}_\delta) = 0$.

Proof. (of Theorem 5). It will suffice to express $\mathrm{DTIME}(f(n))$ as a $\mathcal{C}$-union of $\Gamma(\mathcal{C})$-null sets. But this is easy. Let $M_1, M_2, \ldots$ be a clocked enumeration of $\mathrm{DTIME}(f(n))$ machines, and let

$$d_{j,k}(w) = \begin{cases} 0 & \text{if any of bits } \{1, 2, 4, \ldots, 2^k\} \text{ of } w \text{ are inconsistent with } L(M_j) \\ 2^{\lfloor \log |w| \rfloor - k} & \text{otherwise.} \end{cases}$$

The result follows. ∎

It is also important for our applications to note that the "Borel-Cantelli-Lutz Lemma" (see [L92]) holds also for our measure:

Lemma 8 *Let* $\mathcal{C} = \mathrm{DTIME}(\mathcal{F})$. *Let* $\{X_j\}$ *be a collection of sets, where* X_j *is covered by* d_j *where* $d \in \Gamma(\mathcal{C})$. *Let* m *(the "modulus of convergence") be of the form* $f(\log n)$ *for some* $f \in \mathcal{F}$, *such that for all* k *we have*

$$\sum_{i=m(k)}^{\infty} d_{j,i}(\lambda) < 2^{-k}.$$

Then

$$\mu_{\Gamma(\mathcal{C})}(\bigcap_{t=0}^{\infty} \bigcup_{k=t}^{\infty} X_j) = 0.$$

The proof of this is the same as in [L92], but it is important here to note that for subexponential complexity classes it seems unavoidable that the modulus be *sublinear*. Thus, for the Borel-Cantelli-Lutz Lemma to be applied to subexponential-time classes, the sets X_j must be covered by a "rapidly-vanishing" collection of null covers.

Due to the extremely limited computation power of sublinear-time machines, it is often quite easy to prove that certain sets are *not* $\Gamma(\mathrm{P})$-null. Below, we present one concrete example; a related result appears in [S].

Note that δ is clearly a constructor in $\Gamma(\mathcal{C})$, and thus defines a language L in $\mathcal{C}$. (To see that δ is in $\Gamma(\mathcal{C})$, note that

$$G_{\delta,|w|} \subset G_{d,|w|} \cup (G_{d,|w0|} \setminus \{|w|+1\}),$$

which provides the bound on the size of the dependency set.)

Finally, we have to show L is not covered by d, i.e., for all n we have $d(\delta^n(\lambda)) < 1$. Again we use a strategy in [L92], an inductive argument. We have

$$\hat{d}_{a(x)}(\delta(x)) \leq \hat{d}_{a(x)}(x) + 2^{1-a(x)} \tag{1}$$

since, if $\hat{d}_{a(x)}(x0) > \hat{d}_{a(x)}(x) + 2^{1-a(x)}$ then

$$\begin{aligned}
\hat{d}_{a(x)}(x1) &\leq d(x1) + 2^{-a(x)}, \text{ since } \hat{d} \text{ is a computation} \\
&\leq 2d(x) - d(x0) + 2^{-a(x)}, \text{ by the average condition} \\
&\leq 2\hat{d}_{a(x)}(x) - \hat{d}_{a(x)}(x0) + 2^{2-a(x)}, \text{ a computation} \\
&< \hat{d}_{a(x)}(x) + 2^{1-a(x)}, \text{ by hypothesis.}
\end{aligned}$$

Thus, if $d(x) < q + 2^{-l}(1 - 2^{-|x|})$, we get

$$\begin{aligned}
d(\delta(x)) &\leq \hat{d}_{a(x)}(\delta(x)) + 2^{-a(x)}, \text{ a computation} \\
&\leq \hat{d}_{a(x)}(x) + 3 \cdot 2^{-a(x)}, \text{ by (1)} \\
&\leq d(x) + 2^{2-a(x)}, \text{ a computation} \\
&< q + 2^{-l}(1 - 2^{-|x|}) + 2^{2-a(x)}, \text{ by hypothesis} \\
&\leq q + 2^{-l}(1 - 2^{-|x|}) + 2^{-(|x|+l+1)}, \text{ by definition of } a(x) \\
&= q + 2^{-l}(1 - 2^{-|\delta(x)|}).
\end{aligned}$$

Thus, by induction,

$$\begin{aligned}
d(\delta^k(\lambda)) &< q + 2^{-l}(1 - 2^{-|\delta^k(\lambda)|}) \\
&< q + 2^{-l} \\
&\leq 1.
\end{aligned}$$

∎

Let $\sigma = \sum_{j=0}^{\log(r+|w|)} d_{j,k+2^j+1}(w)$. Then (as in [L92], and letting $j' = 2^j$) :

$$\begin{aligned}
|d'_k(w) - \sigma| &\leq \sum_{j=\log(r+|w|+1)}^{\infty} d_{j,k+2^j+1}(w) \\
&\leq \sum_{j=\log(r+|w|+1)}^{\infty} 2^{|w|} d_{j,k+2^j+1}(\lambda) \\
&\leq \sum_{j=\log(r+|w|+1)}^{\infty} 2^{|w|} 2^{-(k+2^j+1)} \\
&\leq \sum_{j'=r+|w|+1}^{\infty} 2^{|w|} 2^{-(k+j'+1)} \\
&= 2^{-(k+r+1)} \\
&\leq 2^{-(r+1)}.
\end{aligned}$$

Also,

$$\begin{aligned}
|\sigma - \hat{d}'_{k,r}(w)| &\leq \sum_{j=0}^{\log(r+|w|)} |d_{j,k+2^j+1} - \hat{d}_{j,k+2^j+1,r+2^j+2}(w)| \\
&\leq \sum_{j=0}^{\log(r+|w|)} 2^{-(r+2^j+2)} \\
&\leq \sum_{j=0}^{\infty} 2^{-(r+j+2)} \\
&= 2^{-(r+1)}
\end{aligned}$$

as in [L92]. ∎

Theorem 4 $\mu_{\Gamma(\mathcal{C})}(\mathcal{C}) \neq 0$.

Proof. We show that if d is any $\Gamma(\mathcal{C})$-computable density function with $d(\lambda) < 1$;, then we can construct a language in $\mathcal{C}$ not covered by d.

As in [L92], find q, l such that

$$d(\lambda) < q < q + 2^{-l} \leq 1.$$

Let $a(x) \geq |x| + l + 3$, and define the desired language L by

$$\delta(w) = \begin{cases} 1 & \text{if } \hat{d}_{a(w)}(w0) > \hat{d}_{a(w)}(w) + 2^{1-a(w)} \\ 0 & \text{otherwise.} \end{cases}$$

2.3 The Measure Axioms Are Satisfied

A set X is a *$\mathcal{C}$-union of $\Gamma(\mathcal{C})$-null sets* if $X = \bigcup_{j=0}^{\infty} X_j$, and there is a 2-DS $d_{j,k}$ so that $d_{j,k}$ covers X_j with value 2^{-k}, and $d_{j,k}$ has a $\Gamma(\mathcal{C})$-computation $\hat{d}_{j,k,r}$.

Theorem 2 *For each $L \in \mathcal{C}$, the singleton set $\{L\}$ is $\Gamma(\mathcal{C})$-null.*

Proof. Given a language $L \in \mathcal{C}$, we will construct a null cover $d_k(w)$ of the singleton set $\{L\}$ as follows: Decide membership of each word x such that $\mathrm{pos}(x)$ is a power of 2 and is at most $|w|$. Suppose there are s of these. Output 2^{s-k} if bits $1, 2, 4, \ldots 2^s$ of w are all consistent with L, and output 0 otherwise.

It is easy to check that this yields a cover; we show here that $d \in \Gamma(\mathcal{C})$. Since $L \in \mathcal{C}$ by hypothesis, we can check membership of any single word x in time $f(|x|)$. (This involves no queries to the input w of d). We can in polylog time check all log-many words whose lexicographic position is a power of 2. Also, we query the input at the bits whose positions are powers of 2, generating log-many points in the dependency set. The crucial observation is that the transitive closure of these bits is trivial; no more points are added. ∎

Theorem 3 *If X is a $\mathcal{C}$-union of $\Gamma(\mathcal{C})$-null sets, then X is $\Gamma(\mathcal{C})$-null.*

Proof. Let $d_{j,k}$ be a 2-DS witnessing that X is a $\mathcal{C}$-union of $\Gamma(\mathcal{C})$-null sets. Let

$$d'_k(w) = \sum_{j=0}^{\infty} d_{j,k+2^j+1}(w).$$

We will show d' is a $\Gamma(\mathcal{C})$-null cover of X.

It is immediate that d' is a 1-DS, and that $d'_k(\lambda) \leq 2^{-k}$. Now, to see that d' covers X, note that if ω is any sequence in X, then ω is in some X_j, and for all k there is a $w_k \sqsubseteq \omega$ and $d_{j,k}(w_k) \geq 1$. Now note that for all k, $d'_k(w_{k+2^j+1}) \geq 1$, and thus d' is a null cover of X. It remains only to show that d' has $\Gamma(\mathcal{C})$-computations.

Define the computation

$$\hat{d}'_{k,r}(w) = \sum_{j=0}^{\log(r+|w|)} \hat{d}_{j,k+2^j+1,r+2^j+2}(w).$$

If $\hat{d}$ is computable in time $f(\log n)$ with dependency set size also bounded by $f(\log n)$, then $\hat{d}'$ is computable in time $O(\log n (f(\log n))^2)$, and is in $\Gamma(\mathcal{C})$by the closure properties assumed of $\mathcal{F}$. (There are only $\log n$ many terms to add. Computing the sum of two addends amounts to sorting all the powers of 2 and performing carries (carries correspond to ties in the sort), and carries can propagate at most $f(\log n)$ positions.)

Note that if the functions in $\mathcal{F}$ are at least exponential, then $\Delta(\mathcal{C}) = \Gamma(\mathcal{C})$, and hence our definitions coincide with those of [L92] for complexity classes at least as large as E.

If f is a function in $\Gamma(\mathcal{C})$, where $f : \{0,1\}^* \to \{0,1\}$, then f defines a *constructor* δ, where $\delta(w) = wf(w)$. A constructor specifies the sequence that is the limit as $j \to \infty$ of $\delta^j(\lambda)$. This gives rise to the class $R(\Gamma(\mathcal{C}))$, which is the class of languages whose characteristic sequences are given by some constructor in $\Gamma(\mathcal{C})$. The definitions presented above were designed to maintain the following important property of the system developed in [L92]:

Proposition 1 $R(\Gamma(\mathcal{C})) = \mathcal{C}$.

Proof. Any language in $\mathcal{C}$ has a trivial constructor that ignores all but the length of its input, so we have $\mathcal{C} \subseteq R(\Gamma(\mathcal{C}))$.

On the other hand, given such a constructor δ, we can decide if $x \in R(\delta)$ as follows: Start running δ on $w = \delta^{\mathrm{pos}(x)-1}(\lambda)$. When δ needs bit i of w, recursively call δ on $w[1,..i-1]$ to generate that bit. Unwinding this recursion effectively generates the dependency set $G_{\delta,|w|}$, which by hypothesis has size at most $f(\log|w|) \approx f(|x|)$. For each string y in the dependency set, we require at most time $f(|y|)$, for total computation time about $(f(|x|))^2$, which is in $\mathcal{F}$ by the assumed closure properties. ∎

We now make precise the notion of a density system (DS) being easy to compute. A $\Gamma(\mathcal{C})$-*computation* of an n-DS $d_{i_1,\ldots,i_n}$ is an $(n+1)$-subscripted function $\hat{d}_{i_1,\ldots,i_n,r}$ satisfying

- $\hat{d}_{i_1,\ldots,i_n,r}(w)$ is computable in $\Gamma(\mathcal{C})$, (and numeric output is represented as a formal sum of powers of 2).
- $|d_{i_1,\ldots,i_n}(w) - \hat{d}_{i_1,\ldots,i_n,r}(w)| \leq 2^{-r}$, where the subscripts $i_1,\ldots,i_n,r$ are represented in unary.

Note that a $\Gamma(\mathcal{C})$-computation does not have to satisfy the average law (item 3 of Definition 1), though it has to be close, depending on r.

For any complexity class $\mathcal{C}$, we say that a set X is $\Gamma(\mathcal{C})$-null if there is a $\Gamma(\mathcal{C})$-computation of a 1-DS $d_k(w)$ covering X such that for all k, $d_k(\lambda) \leq 2^{-k}$. In this case we write[3] $\mu_{\Gamma(\mathcal{C})}(X) = 0$.

If a class X has nonzero measure, we can talk about another set Y having "measure 0 in X," and we write $\mu_{\Gamma(\mathcal{C})}(Y|X) = 0$, if $\mu_{\Gamma(\mathcal{C})}(Y \cap X) = 0$ or "measure 1 in X," $\mu_{\Gamma(\mathcal{C})}(Y|X) = 1$, if $\mu_{\Gamma(\mathcal{C})}(X \setminus Y) = 0$.

---

reader should note that providing the length of a unary argument obviates any need to provide the argument itself. We continue to determine the complexity as if the subscripts were presented in unary.

[3]This is a slight change of notation from that used by Lutz and others. For example, if $\mathcal{C}$ is taken to be E, then $\Delta(\mathcal{C}) = \Gamma(\mathcal{C})$ is the class of functions computed in polynomial time, and if X is a measure-zero subset of E, we would express that as $\mu_{\Gamma(\mathrm{E})}(X) = 0$, while [L92] would express this as $\mu_{\mathrm{p}}(X) = 0$.

in [M]: the binary sequences that are constructible in DTIME($\log^{O(1)} n$) correspond *not* to sets in P, but rather to *"word-decreasing self-reducible"* sets, some of which are hard for E [Ba]. This motivates our notion of *limiting the dependency set size*, which allows us to obtain subexponential bounds on the complexity.

2.2 Formal Definitions of the Measure

The preceding paragraphs motivate some detailed definitions of a class of functions computed by sublinear-time Turing machines. In order for sublinear-time machines to perform interesting computation, we follow the usual convention of providing these machines with random access to their input; that is, the machines have an "address tape," and if i is written in binary on the address tape, then the machine can in unit time move its read/write head to bit position i of the input. Among other things, such machines can, in logarithmic time, compute the length of their input ([Bu]). In order to avoid uninteresting technicalities regarding encoding of pairing functions, we adopt the convention that a machine computing a k-ary function is provided with k input tapes (with an "address tape" for each input tape). The running time of such a machine must be polylogarithmic in m, where m is the sum of the lengths of its k inputs. (Note that by choosing a suitable encoding, such a machine can be simulated by a machine with a single input tape.) The machines we consider will write their output on a write-only output tape; thus the output is restricted to be of length bounded by the running time.

Given a machine M and natural number n, define a *dependency set* $G_{M,n} \subset \{1, \ldots, n\}$ to be a set such that for each $i \in G_{M,n} \cup \{n\}$, and each word w of length n, M can compute $M(w[1..i])$ querying only bits in $G_{M,n} \cap \{1, 2, \ldots i\}$. Note that for all M and n, there is a unique minimal dependency set for M and n, which can easily be computed by expanding the tree of queries that one obtains by assuming both possible values for each queried bit. In what follows, we let $G_{M,n}$ denote this minimal dependency set. Given a function f computed by machine M, we may abuse notation and speak of $G_{f,n}$ instead of $G_{M,n}$, where this causes no confusion.

Given a complexity class $\mathcal{C}$ of the form DTIME($\mathcal{F}$) (where we will always assume that $\mathcal{F}$ is a set of time-constructible functions such that $f(n) \in \mathcal{F} \Rightarrow (f(n))^2 \in \mathcal{F}$), let $\Delta(\mathcal{C})$ be the class of functions computed by Turing machines running in time $f(\log n)$ for some $f \in \mathcal{F}$, and let $\Gamma(\mathcal{C})$ be the class of functions computed by machines whose runtime and dependency set size are both bounded by functions of the form $f(\log n)$ for some $f \in \mathcal{F}$.[2]

[2]One uninteresting technicality that needs to be addressed concerns the manner in which machines determine the length of their input. If one were to use the usual binary search approach to determine the length of the input, then the dependency sets cannot be kept small while allowing any sort of interesting computation. Our approach to this problem is to have the length of the input be given explicitly as an argument to the machine. (An equivalent approach would be to declare that the input consists of 0's and 1's and is followed by blanks, and to include in the dependency set only queries that ask to distinguish between a 0 and a 1, and not to count those queries that ask only to distinguish between a blank and a nonblank.) We follow the (useful) convention of [L92] that "subscripts" to density functions are presented in unary; however the

In defining notions of measure[1] for E, ESPACE, and other large complexity classes, Lutz defines a Δ-measure for various classes of functions Δ. Roughly, the sets with Δ-measure zero are the sets that have a null cover that can be approximated very closely by a function in Δ. If Δ is chosen to be the polynomial-time computable functions, Lutz obtains a measure on E; if Δ is chosen to be the functions computable in space $n^{\log^{O(1)} n}$, then he obtains a measure on DSPACE($2^{n^{O(1)}}$), and in general functions computed in space or time $\{f(\log n) : f \in \mathcal{C}\}$ for a sufficiently nice class $\mathcal{C}$ of functions yield a measure on space or time $\{f(n) : f \in \mathcal{C}\}$.

In order to justify his claim that the notion he defines does, in fact, constitute a reasonable notion of measure on a complexity class X, Lutz [L92] notes that if Δ is chosen to be the class of *all* functions, then his definitions are equivalent to Lebesgue measure, and he also formulates three "measure axioms" that his system satisfies:

M 1 *Easy unions of null sets are null.*

M 2 *Singleton sets of easy languages are null.*

M 3 *X itself is not null.*

Our goal in defining a measure on small complexity classes is to formulate a system that is as simple and natural as possible, while still satisfying these axioms.

When faced with the task of defining measure on classes smaller than E, it is natural to try to modify Lutz's definitions, merely using smaller resource bounds $\mathcal{C}$. For instance, to define a measure on P, one would consider density functions computed in DTIME($\log^{O(1)} n$). This fails to work, because

- The usual convention (see, e.g., [AJ]) for having sublinear-time Turing machines compute some function f is to have them recognize the language $\{x, i, b :$ bit i of $f(x)$ is $b\}$. However it is easy to see that the class of functions computed in this way by, for instance, polylog-time-bounded machines, is not closed under addition, which seems to be necessary for many constructions. Similar problems arise when one uses the usual binary notation or scientific notation for the numeric values of the density functions. Our solution is to have the run time bound the length of the output, and to express numbers as formal powers of two, which allows us to perform the necessary arithmetic operations in the restricted time available.

- Using this representation for numeric values, one obtains a system that quite possibly *does* define a measure on P. It unfortunately seems quite difficult to verify that P itself does not have measure zero (which would violate Measure Axiom 3). This is discussed in more detail in [AS], but the central problem lies with an observation made previously

[1] The classes of sets that we are interested in measuring are complexity classes, which are closed under finite variants and thus have measure zero or one if they are measurable at all. For this reason, and for the reason that an extra layer of complication is involved in making the notion of measure precise for sets of nonzero measure, we follow Lutz's lead in defining only what it means for a set to have measure zero or one.

Section 4 presents our results concerning pseudorandom sources for BPP.

Section 5 includes a discussion of whether the notion of measure introduced in this paper relativizes in a meaningful way. (That is, we discuss the question of whether nonrelativizing results might be obtained using this notion of measure.)

2 Defining Resource-Bounded Measure

2.1 Background Concerning Measure

In order to present our new definitions of resource-bounded measure and explain the obstacles to extending Lutz's definitions to subexponential complexity classes, it is necessary to sketch Lutz's formulation. For more details, consult [L92].

The intuition behind Lutz's formulation of resource-bounded measure is that a measure-zero set is a set such that, for all k, the set can be covered by a collection of intervals whose sizes sum to 2^{-k}, where the "intervals" are easy to compute in some sense.

To make this more precise, fix an enumeration $\{s_i\}_{i=0}^{\infty}$ of all words, and let $\mathrm{pos}(x)$ denote the position of x in this ordering. A language L is identified with its characteristic sequence: an infinite sequence $\omega = \langle a_i \rangle$ with $a_i = 1$ iff $s_i \in L$. Let $w[i..j]$ denote the string of the i^{th} through j^{th} bits of w. Let $y \sqsubseteq w$ indicate that y is an initial substring of w.

The sets we will try to measure are sets of languages (i.e., sets of sequences), and the analogs of "intervals" in this context are the sets of the form $C_x = \{\omega \sqsupseteq x\}$, which is called the "cylinder at x". (Thus C_x is the set of languages with the $|x|$ initial words according to x and the other words arbitrary).

The next few definitions describe what it means in this context for a set to be covered, for all k, by a collection of intervals whose size sums to 2^{-k}.

Definition 1 *A* density-system *is a function* $d_{i_1,\ldots,i_r}\{0,1\}^* \to [0,\infty)$, *where –*

1. *The subscripts are natural numbers.*
2. *The argument is considered a partial characteristic sequence of a language.*
3. $d_{(\cdot)}(w) \geq \frac{d_{(\cdot)}(w0)+d_{(\cdot)}(w1)}{2}$.

A density-system is called an n-DS according to the number of subscripts. A 0-DS is a density function. (Note there is no mention here of resource bounds.)

A density function *covers* X if $X \subset \bigcup_{\{w|\, d(w)\geq 1\,\}} C_w$. A *null cover* of X is a 1-DS such that for all k, d_k covers X and $d_k(\lambda) \leq 2^{-k}$. A set of languages has Lebesgue-measure zero iff it has a null cover. Of course, every complexity class is countable and thus has Lebesgue-measure zero. Thus it was necessary for Lutz to restrict the class of density systems in order to define measures on complexity classes.

1 Introduction

Resource-bounded measure theory was introduced in [L90a] and has proved to be a useful tool in complexity theory. Among the reasons for studying resource-bounded measure, we mention the following:

- The probabilistic method can be applied only if there is a meaningful notion of probability on the underlying space. To show that a set in E having some property Q exists, it is sufficient to show that the class of sets that do *not* have property Q has measure zero in E. (This sometimes presents itself as an easier task than explicitly building a set in E having property Q.) For example, Lutz and Mayordomo [LM] were able to use this version of the probabilistic method to prove stronger results concerning the density of hard languages than had been proved previously.

- Certain interesting notions of intractability can be formulated only in the context of resource-bounded measure. Lutz and Juedes [JL] studied "weakly complete" sets in E, which are sets in E that are hard for more than a measure-zero subset of E.

- Many separation results and hierarchy theorems in complexity theory have stronger versions provable in the context of resource-bounded measure. For example, the $\leq_m^p$-complete sets are a measure-zero subset of E [JL], and DTIME(2^{cn}) is a measure-zero subset of E [L92]. However, for many other separations, no corresponding measure result is known. Often, when a measure-theoretic strengthening of a separation result *is* possible, the proof gives enlightening additional information concerning the complexity classes involved.

- Unproven but plausible hypotheses such as "P $\neq$ NP" or the stronger hypothesis "the polynomial hierarchy is infinite" provide useful information concerning complexity-theoretic propositions. (That is, "$X \Rightarrow$ the polynomial hierarchy collapses" is taken as evidence for $\neg X$.) Lutz [L93a] argues that the hypothesis "NP is not a measure zero subset of DTIME($2^{n^{O(1)}}$)" (a stronger hypothesis than P $\neq$ NP) is plausible and offers a great deal of explanatory power.

Unfortunately, the formulation of resource-bounded measure studied by Lutz and others applies only to complexity classes at least as large as E, which greatly limits the range of questions that can be explored in this framework. The goal of the work presented here is to remedy this situation by providing a meaningful notion of measure for P and other subexponential time classes, and demonstrate its usefulness.

Section 2 presents Lutz's definition of resource-bounded measure, explains why it does not extend in any obvious way to provide a measure on smaller classes, and then shows how these obstacles were overcome to provide a measure on P and other classes.

Section 3 presents our proof that sets that are hard for BPP are abundant in E_ϵ for every $\epsilon > 0$, as well as related results for PSPACE.

ABSTRACT

The main contributions of this work are:

1. We present a notion of resource-bounded measure for P and other subexponential-time classes. This is a generalization of Lutz's notion of measure, but Lutz's definitions apply only to classes at least as large as E, and several obstacles needed to be overcome to give a meaningful notion of measure for smaller classes. We present some of the basic properties of this measure; for example, $\forall k$ DTIME(n^k) is a measure 0 subset of P.

2. We use this new notion of measure to show that for all $\epsilon > 0$ almost every set A in the subexponential class E_ϵ is hard for BPP, where $\mathrm{E}_\epsilon = \bigcup_{\delta<\epsilon}\mathrm{DTIME}(2^{n^\delta})$. This is best possible without improving the known bounds on the deterministic time complexity of sets in BPP. Using similar techniques, we show that almost every set $A \in$ PSPACE also satisfies this property. (This exponentially improves on the result of [L93] showing that almost every set in ESPACE satisfies this property.)

DIMACS Technical Report 94-18
April 1994

Measure on Small Complexity Classes, with Applications for BPP

by

Eric Allender[1,2]
Department of Computer Science
Rutgers University
New Brunswick, NJ 08903

Martin Strauss[3,4]
Department of Mathematics
Rutgers University
New Brunswick, NJ 08903

[1]Permanent Member
[2]Research supported by NSF grant CCR-9204874.
[3]Affiliated Graduate Student Member
[4]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.