A Generalization of Resource-Bounded Measure, With Application to the BPP vs. EXP Problem

Harry Buhrman* Dieter van Melkebeek†

Kenneth W. Regan‡ D. Sivakumar§ Martin Strauss¶

January 11, 2000

Abstract

We introduce *resource-bounded betting games*, and propose a generalization of Lutz's resource-bounded measure in which the choice of next string to bet on is fully adaptive. Lutz's martingales are equivalent to betting games constrained to bet on strings in lexicographic order. We show that if strong pseudo-random number generators exist, then betting games are equivalent to martingales, for measure on E and EXP. However, we construct betting games that succeed on certain classes whose Lutz measures are important open problems: the class of polynomial-time Turing-complete languages in EXP, and its superclass of polynomial-time Turing-autoreducible languages. If an EXP-martingale succeeds on either of these classes, or if betting games have the "finite union property" possessed by Lutz's measure, one obtains the non-relativizable consequence

*CWI, Kruislaan 413, 1098SJ Amsterdam, The Netherlands. Partially supported by the Dutch foundation for scientific research (NWO) through SION project 612-34-002, and by the European Union through NeuroCOLT ESPRIT Working Group Nr. 8556, and HC&M grant nr. ERB4050PL93-0516. E-mail: `buhrman@cwi.nl`.

†DIMACS Center, Rutgers University, 96 Frelinghuysen Road, CoRE Building - Room 419, Piscataway, NJ 08854-8018 USA. Partly supported by the European Union through Marie Curie Research Training Grant ERB-4001-GT-96-0783 at CWI and at the University of Amsterdam; by NSF Grant CCR 92-53582; and by the Fields Institute of the University of Toronto; research performed mainly at the University of Chicago. E-mail: `dieter@dimacs.rutgers.edu`

‡Department of Computer Science and Engineering, State Univ. of NY at Buffalo, 226 Bell Hall, Buffalo, NY 14260-2000 USA. Supported in part by the National Science Foundation under Grant CCR-9409104. E-mail: `regan@cse.buffalo.edu`

§IBM Almaden Research Center, Dept. K-53, 650 Harry Road San Jose, CA 95120 USA. Part of this research was performed while at the State Univ. of N.Y. at Buffalo, supported in part by National Science Foundation Grant CCR-9409104. E-mail: `siva@almaden.ibm.com`

¶AT&T Labs Room C216, 180 Park Ave, Florham Park, NJ 07932-0971 USA. Research performed while at Rutgers University and Iowa State University, supported by NSF grants CCR-9204874 and CCR-9157382. E-mail: `mstrauss@research.att.com`

BPP $\neq$ EXP. We also show that if EXP $\neq$ MA, then the polynomial-time truth-table-autoreducible languages have Lutz measure zero, whereas if EXP = BPP, they have measure one.

Key words. Computational complexity, theory of computation, probabilistic computation, complexity classes, resource-bounded measure, betting games, polynomial reductions, pseudorandom generators, sampling, autoreducibility.

AMS/MOS classification. 68Q15

1 Introduction

Lutz's theory of measure on complexity classes is now usually defined in terms of resource-bounded martingales. A martingale can be regarded as a gambling game played on unseen languages A. Let $s_1, s_2, s_3, \ldots$ be the standard lexicographic ordering of strings. The gambler G starts with capital $C_0 = \$1$ and places a bet $B_1 \in [0, C_0]$ on either "$s_1 \in A$" or "$s_1 \notin A$." Given a fixed particular language A, the bet's outcome depends only on whether $s_1 \in A$. If the bet wins, then the new capital C_1 equals $C_0 + B_1$, while if the bet loses, $C_1 = C_0 - B_1$. The gambler then places a bet $B_2 \in [0, C_1]$ on (or against) membership of the string s_2, then on s_3, and so forth. The gambler *succeeds* if G's capital C_i grows toward $+\infty$. The class $\mathcal{C}$ of languages A on which G succeeds (and any subclass) is said to have *measure zero*. One also says G *covers* $\mathcal{C}$. Lutz and others (see [24]) have developed a rich and extensive theory around this measure-zero notion, and have shown interesting connections to many other important problems in complexity theory.

We propose the generalization obtained by lifting the requirement that G must bet on strings in lexicographic order. That is, G may begin by choosing any string x_1 on which to place its first bet, and after the oracle tells the result, may choose any other string x_2 for its second bet, and so forth. Note that the sequences $x_1, x_2, x_3, \ldots$ (as well as $B_1, B_2, B_3, \ldots$) may be radically different for different oracle languages A—in complexity-theory parlance, G's queries are *adaptive*. The lone restriction is that G may not query (or bet on) the same string twice. We call G a *betting game*.

Our betting games remedy a possible lack in the martingale theory, one best explained in the context of languages that are "random" for classes $\mathcal{D}$ such as E or EXP. In this paper, E stands for deterministic time $2^{O(n)}$, and EXP stands for deterministic time $2^{n^{O(1)}}$. A language L is $\mathcal{D}$-*random* if L cannot be covered by a $\mathcal{D}$-martingale. Based on one's intuition about random 0-1 sequences, the language $L' = \{\, \mathit{flip}(x) : x \in L \,\}$ should likewise be $\mathcal{D}$-random, where $\mathit{flip}(x)$ changes every 0 in x to a 1 and vice-versa. However, this closure property is not known for E-random or EXP-random languages, because of the way martingales are tied to the fixed lexicographic ordering of Σ^*. Betting games can adapt to easy permutations of Σ^* such as that induced by *flip*. Similarly, a class $\mathcal{C}$ that is *small* in the sense of being covered by a ($\mathcal{D}$-) betting game remains small if the languages $L \in \mathcal{C}$ are so permuted. In the r.e./recursive theory of random languages, our generalization is similar to "Kolmogorov-Loveland place-selection rules" (see [22]). We make this theory work for complexity classes via a novel definition of "running in time $t(n)$" for an infinite process.

Our new angle on measure theory may be useful for attacking the problem of separating BPP from EXP, which has recently gained prominence in [17]. In Lutz's theory it is open whether the class of EXP-complete sets—under polynomial-time Turing reductions—has EXP-measure zero. If so (in fact if this set does not have measure one), then by results of Allender and Strauss [1], BPP $\neq$ EXP. Since there are oracles A such that $\mathrm{BPP}^A = \mathrm{EXP}^A$ [14], this kind of absolute separation would be a major breakthrough. We show that the EXP-complete sets *can* be covered by an EXP betting game—in fact, by an E-betting game. The one technical lack in our theory as a notion of measure is also interesting here: If the "finite unions" property holds for betting games (viz. $\mathcal{C}_1$ small $\wedge$ $\mathcal{C}_2$ small $\Longrightarrow \mathcal{C}_1 \cup \mathcal{C}_2$ small), then EXP $\neq$ BPP. Likewise, if Lutz's martingales do enjoy the permutation-invariance of betting games, then BPP $\neq$ EXP. Finally, we show that if a pseudo-random number generator of security $2^{n^{\Omega(1)}}$ exists, then for every EXP-betting game G one can find an EXP-martingale that succeeds on all sets covered by G. Pseudo-random generators of higher security $2^{\Omega(n)}$ likewise imply the equivalence of E-betting games and E-measure. Ambos-Spies and Lempp [4] proved that the EXP-complete sets have E-measure zero under a different hypothesis, namely P = PSPACE.

Measure theory and betting games help us to dig further into questions about pseudo-random generators and complexity-class separations. Our tool is the notion of an *autoreducible* set, whose importance in complexity theory was argued by Buhrman, Fortnow, van Melkebeek, and Torenvliet [11] (after [10]). A language L is $\leq_T^p$*-autoreducible* if there is a polynomial-time oracle Turing machine Q such that for all inputs x, Q^L correctly decides whether $x \in L$ without ever submitting x itself as a query to L. If Q is non-adaptive (i.e., computes a polynomial-time truth-table reduction), we say L is $\leq_{tt}^p$*-autoreducible*. We show that the class of $\leq_T^p$-autoreducible sets is covered by an E-betting game. Since every EXP-complete set is $\leq_T^p$-autoreducible [11], this implies results given above. The subclass of $\leq_{tt}^p$-autoreducible sets provides the following tighter connection between measure statements and open problems about EXP:

- If the $\leq_{tt}^p$-autoreducible sets do not have E-measure zero, then EXP = MA.
- If the $\leq_{tt}^p$-autoreducible sets do not have E-measure one in EXP, then EXP $\neq$ BPP.

Here MA is Babai's "Merlin-Arthur" class, which contains BPP and NP and is contained in the level $\Sigma_2^p \cap \Pi_2^p$ of the polynomial hierarchy [5, 7]. Since

EXP $\neq$ MA is strongly believed, one would expect the class of $\leq_{tt}^{p}$-autoreducible sets to have E-measure zero, but *proving* this—or proving any of the dozen other measure statements in Corollaries 6.2 and 6.5—would yield a proof of EXP $\neq$ BPP.

In sum, the whole theory of resource-bounded measure has progressed far enough to wind the issues of (pseudo-)randomness and stochasticity within exponential time very tightly. We turn the wheels a few more notches, and seek greater understanding of complexity classes in the places where the boundary between "measure one" and "measure zero" seems tightest.

Section 2 reviews the formal definitions of Lutz's measure and martingales. Section 3 introduces betting games, and shows that they are a generalization of martingales. Section 4 shows how to simulate a betting game by a martingale of perhaps-unavoidably higher time complexity. Section 5, however, demonstrates that strong pseudo-random generators (if there are any) allow one to compute the martingale in the same order of time. Section 6 presents our main results pertaining to autoreducible sets, including our main motivating example of a concrete betting game. The concluding Section 7 summarizes open problems and gives prospects for future research.

A preliminary version of this paper without proofs appeared in the proceedings of STACS'98, under the title "A Generalization of Resource-Bounded Measure, With an Application."

2 Martingales

A *martingale* is abstractly defined as a function d from $\{0,1\}^*$ into the non-negative reals that satisfies the following "average law": for all $w \in \{0,1\}^*$,

$$d(w) = \frac{d(w0) + d(w1)}{2}. \tag{1}$$

The interpretation in Lutz's theory is that a string $w \in \{0,1\}^*$ stands for an initial segment of a language over an arbitrary alphabet Σ as follows: Let $s_1, s_2, s_3, \ldots$ be the standard lexicographic ordering of Σ^*. Then for any language $A \subseteq \Sigma^*$, write $w \sqsubseteq A$ if for all i, $1 \leq i \leq |w|$, $s_i \in A$ iff the ith bit of w is a 1. We also regard w as a function with *domain* $dom(w) = \{s_1, \ldots, s_{|w|}\}$ and range $\{0,1\}$, writing $w(s_i)$ for the ith bit of w. A martingale d *succeeds on* a language A if the sequence of values $d(w)$ for $w \sqsubseteq A$ is unbounded.

Let $S^\infty[d]$ stand for the (possibly empty, often uncountable) class of languages on which d succeeds. Lutz originally defined the complexity of a martingale d in terms of computing fast-converging rational approximations to d. Subsequently he showed that for certain classes of time bounds one loses no generality by requiring that martingales themselves have rational values a/b such that all digits of the integers a and b (not necessarily in lowest terms) are output within the time bound. That is, given any martingale d meeting the original definition of computability within the time bound, one can obtain a rational-valued d' computable within that bound such that $S^\infty[d] \subseteq S^\infty[d']$ [25, 18]. We adopt this requirement throughout the paper, and specify that integers are represented in standard binary notation, and rationals as pairs of integers, not necessarily in lowest terms. We use the fact that a sum $a_1/b_1 + \dots a_m/b_m$ can be computed and written down in $\ell^{O(1)}$ time, where ℓ is the sum of the lengths of the integers a_i and b_i.

Definition 2.1 (cf. [23, 25]). Let Δ be a complexity class of functions. A class $\mathcal{C}$ of languages *has* Δ*-measure zero*, written $\mu_\Delta(\mathcal{C}) = 0$, if there is a martingale d computable in Δ such that $\mathcal{C} \subseteq S^\infty[d]$. One also says that d *covers* $\mathcal{C}$.

Lutz measured the time to compute $d(w)$ in terms of the length N of w, but one can also work in terms of the largest length n of a string in the domain of w. For $N > 0$, n equals $\lfloor \log_2 N \rfloor$; all we care about is that $n = \Theta(\log N)$ and $N = 2^{\Theta(n)}$. Because complexity bounds on languages we want to analyze will naturally be stated in terms of n, we prefer to use n for martingale complexity bounds. The following correspondence is helpful:

$$\begin{array}{lclcll} \text{Lutz's "}p\text{"} & \sim & N^{O(1)} & = & 2^{O(n)} & \sim \ \text{measure on E} \\ \text{Lutz's "}p_2\text{"} & \sim & 2^{(\log N)^{O(1)}} & = & 2^{n^{O(1)}} & \sim \ \text{measure on EXP} \end{array}$$

Since we measure the time to compute $d(w)$ in terms of n, we write "μ_{E}" for E-measure and "μ_{EXP}" for EXP-measure, and generally μ_Δ for any Δ that names both a language and function class. Abusing notation similarly, we define:

Definition 2.2 ([23]). A class $\mathcal{C}$ has Δ-measure one, written $\mu_\Delta(\mathcal{C}) = 1$, if $\mu_\Delta(\Delta \setminus \mathcal{C}) = 0$.

The concept of resource bounded measure is known to be robust under several changes [25]. The following lemma has appeared in various forms [25, 12]. It essentially says that we can assume a martingale grows almost monotonically (sure winnings) and not too fast (slow winnings).

Lemma 2.1 ("Slow-but-Sure-Winnings" lemma for martingales) *Let d be a martingale. Then there is a martingale d' with $S^\infty[d] \subseteq S^\infty[d']$ such that*

$$(\forall w)(\forall u) \ : \ d'(wu) > d'(w) - 2d(\lambda), \quad \text{and} \tag{2}$$

$$(\forall w) \ : \ d'(w) < 2(|w| + 1)d(\lambda). \tag{3}$$

If d is computable in time $t(n)$, then d' is computable in time $(2^n t(n))^{O(1)}$.

The idea is to play the strategy of d, but in a more conservative way. Say we start with an initial capital of \$1. We will deposit a part c of our capital in a bank and only play the strategy underlying d on the remaining liquid part e of our capital. We start with no savings and a liquid capital of \$1. If our liquid capital reaches or exceeds \$2, we deposit an additional \$1 or \$2 to our savings account c so as to keep the liquid capital in the range $\$[1, 2)$ at all times. If d succeeds, it will push the liquid capital infinitely often to \$2 or above, so c grows to infinity, and d' succeeds too. Since we never take money out of our savings account c, and the liquid capital e is bounded by \$2, once our total capital $d' = c + e$ has reached a certain level, it will never go more than \$2 below that level anymore, no matter how bad the strategy underlying d is. On the other hand, since we add at most \$2 to c in each step, $d'(w)$ cannot exceed $2(|w| + 1)$ either.

We now give the formal proof.

Proof. (of Lemma 2.1) Define $d' : \Sigma^* \to [0, \infty)$ by

$$d'(w) = (c(w) + e(w))d(\lambda),$$

where $c(\lambda) = 0$ and $e(\lambda) = 1$, and

$$\begin{array}{llll} c(wb) = c(w) & \text{and} & e(wb) = e(w) & \text{if } d(w) = 0; \text{ else:} \\ c(wb) = c(w) + 2 & \text{and} & e(wb) = \frac{d(wb)}{d(w)}e(w) - 2 & \text{if } \frac{d(wb)}{d(w)}e(w) \geq 3; \\ c(wb) = c(w) + 1 & \text{and} & e(wb) = \frac{d(wb)}{d(w)}e(w) - 1 & \text{if } 2 \leq \frac{d(wb)}{d(w)}e(w) < 3; \\ c(wb) = c(w) & \text{and} & e(wb) = \frac{d(wb)}{d(w)}e(w) & \text{if } \frac{d(wb)}{d(w)}e(w) < 2. \end{array}$$

To see that the recursion does not excessively blow up the time complexity or size of the answer, note that owing to cancellation of values of d, every value $e(w)$ where $d(w) \neq 0$ is given by a sum of the form

$$\sum_{k=0}^{N} a_k \frac{d(w)}{d(w[1 \dots k])}$$

where each a_k is in $\{-2, -1, 0, 1\}$, $N = |w|$, and $w[1 \dots k]$ stands for the first k bits of w. Each term in the sum is computable in time $O(t(n)^2 N)$ (using the naive quadratic algorithms for multiplication and integer division). Then by the property noted just before Definition 2.1, these terms can be summed in time $(Nt(n))^{O(1)}$.

By induction on $|w|$ we observe that

$$0 \leq e(w) < 2, \tag{4}$$

and that

$$d'(wb) = \begin{cases} \left[c(w) + \frac{d(wb)}{d(w)}e(w)\right] d(\lambda) & \text{if } d(w) \neq 0 \\ d'(w) & \text{otherwise,} \end{cases}$$

from which it follows that d' is a martingale.

Now let ω be an infinite 0-1 sequence denoting a language on which d succeeds. Then $e(w)$ will always remain positive for $w \sqsubseteq \omega$, and $\frac{d(wb)}{d(w)} e(w)$ will become 2 or more infinitely often. Consequently, $\lim_{w \sqsubseteq \omega, |w| \to \infty} c(w) = \infty$. Since $d'(w) \geq c(w)d(\lambda)$, it follows that $S^\infty[d] \subseteq S^\infty[d']$. Moreover, by Equation (4) and the fact that c does not decrease along any sequence, we have that

$$d'(wu) \geq c(wu)d(\lambda) \geq c(w)d(\lambda) = d'(w) - e(w)d(\lambda) > d'(w) - 2d(\lambda).$$

Since c can increase by at most 2 in every step, $c(w) \leq 2|w|$. Together with Equation (4), this yields that

$$d'(w) = (c(w) + e(w))d(\lambda) < 2(|w| + 1)d(\lambda).$$

□

One can also show that $S^\infty[d'] \subseteq S^\infty[d]$ in Lemma 2.1, so the success set actually remains intact under the above transformation.

As with Lebesgue measure, the property of having resource-bounded measure zero is monotone and closed under union ("finite unions property"). A resource-bounded version of closure under countable unions also holds. The property that becomes crucial in resource-bounded measure is that the whole space Δ does not have measure zero, which Lutz calls the "measure conservation" property. With a slight abuse of meaning for "$\neq$," this property is written $\mu_\Delta(\Delta) \neq 0$. In particular, $\mu_{\mathrm{E}}(\mathrm{E}) \neq 0$ and $\mu_{\mathrm{EXP}}(\mathrm{EXP}) \neq 0$. Subclasses of Δ that require substantially fewer resources, do have Δ-measure zero. For example, P has E-measure zero. Indeed, for any fixed $c > 0$, $\mathrm{DTIME}[2^{cn}]$ has E-measure zero, and $\mathrm{DTIME}[2^{n^c}]$ has EXP-measure zero [23].

Apart from formalizing rareness and abundance in computational complexity theory, resource-bounded martingales are also used to define the concept of a random set in a resource-bounded setting.

Definition 2.3. A set A is Δ-*random* if $\mu_\Delta(\{A\}) \neq 0$.

In other words, A is Δ-random if no Δ-martingale succeeds on A.

3 Betting Games

To capture intuitions that have been expressed not only for Lutz measure but also in many earlier papers on random sequences, we formalize a betting game as an *infinite* process, rather than as a Turing machine that has *finite* computations on string inputs.

Definition 3.1. A *betting game* G is an oracle Turing machine that maintains a "capital tape" and a "bet tape," in addition to its standard query tape and worktapes, and works in *stages* $i = 1, 2, 3 \ldots$ as follows: Beginning each stage i, the capital tape holds a nonnegative rational number C_{i-1}. The *initial capital* C_0 is some positive rational number. G computes a query string x_i to bet on, a *bet amount* B_i, $0 \leq B_i \leq C_{i-1}$, and a *bet sign* $b_i \in \{-1, +1\}$. The computation is *legal* so long as x_i does not belong to the set $\{x_1, \ldots, x_{i-1}\}$ of strings queried in earlier stages. G ends stage i by entering a special query state. For a given oracle language A, if $x_i \in A$ and $b_i = +1$, or if $x_i \notin A$ and $b_i = -1$, then the new capital is given by $C_i := C_{i-1} + B_i$, else by $C_i := C_{i-1} - B_i$. We charge M for the time required to write the numerator and denominator of the new capital C_i down. The query and bet tapes are blanked, and G proceeds to stage $i + 1$.

In this paper, we lose no generality by not allowing G to "crash" or to loop without writing a next bet and query. Note that every oracle set A determines a unique infinite computation of G, which we denote by G^A. This includes a unique infinite sequence $x_1, x_2, \ldots$ of query strings, and a unique sequence $C_0, C_1, C_2, \ldots$ telling how the gambler fares against A .

Definition 3.2. A betting machine G *runs in time* $t(n)$ if for all oracles A, every query of length n made by G^A is made in the first $t(n)$ steps of the computation.

Definition 3.3. A betting game G *succeeds* on a language A, written $A \in S^\infty[G]$, if the sequence of values C_i in the computation G^A is unbounded. If $A \in S^\infty[G]$, then we also say G *covers* A.

Our main motivating example where one may wish not to bet in lexicographic order, or according to any fixed ordering of strings, is deferred to Section 6. There we will construct an E-betting game that succeeds on the class of $\leq_T^p$-autoreducible languages, which is not known to have Lutz measure zero in E or EXP.

We now want to argue that the more liberal requirement of being covered by a time $t(n)$ betting game, still defines a smallness concept for subclasses of DTIME$[t(n)]$ in the intuitive sense Lutz established for his measure-zero notion. The following result is a good beginning.

Theorem 3.1 *For every time-$t(n)$ betting game G, we can construct a language in* DTIME$[t(n)]$ *that is not covered by G.*

Proof. Let Q be a non-oracle Turing machine that runs as follows, on any input x. The machine Q simulates up to $t(|x|)$ steps of the single computation of G on empty input. Whenever G bets on and queries a string y, Q gives the answer that causes G to lose money, rejecting in case of a zero bet. If and when G queries x, Q does likewise. If $t(|x|)$ steps go by without x being queried, then Q rejects x.

The important point is that Q's answer to a query $y \neq x$ is the same as the answer when Q is run on input y. The condition that G cannot query a string x of length n after $t(n)$ steps have elapsed ensures that the decision made by Q when x is not queried does not affect anything else. Hence Q defines a language on which G never does better than its initial capital C_0, and so does not succeed. $\square$

In particular, the class E cannot be covered by an E-betting game, nor EXP by an EXP-betting game. Put another way, the "measure conservation axiom" [23] of Lutz's measure carries over to betting games.

To really satisfy the intuition of "small," however, it should hold that the union of two small classes is small. (Moreover, "easy" countable unions of small classes should be small, as in [23].) Our lack of meeting this "finite union axiom" will later be excused insofar as it has the non-relativizing consequence BPP $\neq$ EXP. Theorem 3.1 is still good enough for the "measure-like" results in this paper.

We note also that several robustness properties of Lutz's measure treated in Section 2 carry over to betting games. This is because we can apply the underlying transformations to the *capital function* c_G of G, which is defined as follows:

DEFINITION 3.4. Let G be a betting game, and $i \geq 0$ an integer.

(a) A *play* α of length i is a sequence of i-many oracle answers. Note that α determines the first i-many stages of G, together with the query and bet for the next stage.

(b) $c_G(\alpha)$ is the capital C_i that G has at the end of the play α (before the next query).

Note that the function c_G is a martingale over plays α. The proof of Lemma 2.1 works for c_G. We obtain:

Lemma 3.2 ("Slow-But-Sure Winnings" lemma for betting games) *Let G be a betting game that runs in time $t(n)$. Then we can construct a betting game G' that runs in time $(2^n t(n))^{O(1)}$ such that $S^\infty[G] \subseteq S^\infty[G']$, G' always makes the same queries in the same order as G, and:*

$$\forall \beta, \forall \gamma : c_{G'}(\beta\gamma) > c_{G'}(\beta) - 2c_G(\lambda) \tag{5}$$
$$\forall \alpha : c_{G'}(\alpha) < 2(|\alpha| + 1)c_G(\lambda). \tag{6}$$

Proof. The proof of Lemma 2.1 carries over. $\square$

To begin comparing betting games and martingales, we note first that the latter can be considered a direct special case of betting games. Say a betting game G is *lex-limited* if for all oracles A, the sequence $x_1, x_2, x_3 \ldots$ of queries made by G^A is in lexicographic order. (It need not equal the lexicographic enumeration $s_1, s_2, s_3, \ldots$ of Σ^*.)

Theorem 3.3 *Let $\mathcal{T}(n)$ be a collection of time bounds that is closed under squaring and under multiplication by 2^n, such as $2^{O(n)}$ or $2^{n^{O(1)}}$. Then a class $\mathcal{C}$ has time-$\mathcal{T}(n)$ measure zero iff $\mathcal{C}$ is covered by a time-$\mathcal{T}(n)$ lex-limited betting game.*

Proof. From a martingale d to a betting game G, each stage i of G^A bets on s_i an amount B_i with sign $b_i \in \{-1, +1\}$ given by $b_i B_i = d(w1) - d(w)$, where w is the first $i-1$ bits of the characteristic sequence of A. This takes $O(2^n)$ evaluations of d to run G up through queries of length n, hence the hypothesis on the time bounds $\mathcal{T}(n)$. In the other direction, when G is lex-limited, one can simulate G on a finite initial segment w of its oracle up to a stage where all queries have been answered by w and G will make no further queries in the domain of w. One can then define $d(w)$ to be the capital entering this stage. That this is a martingale and fulfills the success and run-time requirements is left to the reader. $\square$

Hence in particular for measure on E and EXP, martingales are equivalent to betting games constrained to bet in lexicographic order. Now we will see how we can transform a *general* betting game into an equivalent martingale.

4 From Betting Games to Martingales

This section associates to every betting game G a martingale d_G such that $S^\infty[G] \subseteq S^\infty[d_G]$, and begins examining the complexity of d_G. Before defining d_G, however, we pause to discuss some tricky subtleties of betting games and their computations.

Given a finite initial segment w of an oracle language A, one can define the partial computation G^w of the betting game up to the stage i at which it first makes a query x_i that is not in the domain of w. Define $d(w)$ to be the capital C_{i-1} that G had entering this stage. It is tempting to think that d is a martingale and succeeds on all A for which G succeeds—but neither statement is true in general. The most important reason is that d may fail to be a martingale.

To see this, suppose x_i itself is the lexicographically least string not in the domain of w. That is, x_i is indexed by the bit b of wb, and $w1 \sqsubseteq A$ iff $x_i \in A$. It is possible that G^A makes a small (or even zero) bet on x_i, *and then goes back to make more bets in the domain of w, winning lots of money on them.* The definitions of both $d(w0)$ and $d(w1)$ will then reflect these added winnings, and both values will be greater than $d(w)$. For example, suppose G^A first puts a zero bet on $x_i = s_j$, then bets all of its money on $x_{i+1} = s_{j-1}$ not being in A, and then proceeds with $x_{i+2} = s_{j+1}$. If $w(s_{j-1}) = 0$, then $d(w0) = d(w1) = 2d(w)$.

Put another way, a finite initial segment w may carry much more "winnings potential" than the above definition of $d(w)$ reflects. To capture this potential, one needs to consider potential plays of the betting game outside the domain of w. Happily, one can bound the length of the considered plays via the running time function t of G. Let n be the maximum length of a string indexed by w; i.e., $n = \lfloor \log_2(|w|) \rfloor$. Then after $t(n)$ steps, G cannot query any more strings in the domain of w, so w's potential is exhausted. We will define $d_G(w)$ as an *average* value of those plays that can happen, given the query answers fixed by w. We use the following definitions and notation:

Definition 4.1. For any $t(n)$ time-bounded betting game G and string $w \in \Sigma^*$, define:

(a) A play α is *t-maximal* if G completes the first $|\alpha|$ stages, but *not* the query and bet of the next stage, within t steps.

(b) A play α is *G-consistent with* w, written $\alpha \sim_G w$, if for all stages j such that the queried string x_j is in the domain of w, $\alpha_j = w(x_j)$. That is, α is a play that could possibly happen given the information in w. Also let $m(\alpha, w)$ stand for the number of such stages j whose query is answered by w.

(c) Finally, put $d_G(\lambda) = c_G(\lambda)$, and for nonempty w, with $n = \lfloor \log_2(|w|) \rfloor$ as above, let

$$d_G(w) = \sum_{\alpha\ t(n)-\text{maximal}, \alpha \sim_G w} c_G(\alpha)\ 2^{m(\alpha,w)-|\alpha|}\ . \tag{7}$$

The weight $2^{m(\alpha,w)-|\alpha|}$ in Equation (7) has the following meaning. Suppose we extend the simulation of G^w by flipping a coin for every query outside the domain of w, for exactly i stages. Then the number of coin-flips in the resulting play α of length i is $i - m(\alpha, w)$, so $2^{m(\alpha,w)-i}$ is its probability. Thus $d_G(w)$ returns the suitably-weighted average of $t(n)$-step computations of G with w fixed. The interested reader may verify that this is the same as averaging $d(wv)$ over all v of length $2^{t(n)}$ (or any fixed longer length), where d is the non-martingale defined at the beginning of this section.

Lemma 4.1 *The function $d_G(w)$ is a martingale.*

Proof. First we argue that

$$d_G(w) = \sum_{|\alpha'|=t(n), \alpha' \sim_G w} c_G(\alpha')\ 2^{m(\alpha',w)-t(n)}. \tag{8}$$

Observe that when $\alpha' = \alpha\beta$ and α is $t(n)$-maximal, $\alpha \sim_G w \iff \alpha' \sim_G w$. This is because none of the queries answered by β can be in the domain of w, else the definition of G running in time $t(n)$ would be violated. Likewise if $\alpha \sim_G w$ then $m(\alpha', w) = m(\alpha, w)$. Finally, since c_G is a martingale, $c_G(\alpha) = \sum_{|\beta|=t(n)-|\alpha|} c_G(\alpha\beta)\ 2^{|\alpha|-t(n)}$. These facts combine to show the equality of Equations (7) and (8).

By the same argument, the right-hand side of Equation (8) is unchanged on replacing "$t(n)$" by any $t' > t(n)$.

Now consider w such that $|w| + 1$ is not a power of 2. Then the "n" for $w0$ and $w1$ is the same as the "n" for $d_G(w)$. Let P_0 stand for the set of α of length $t(n)$ that are G-consistent with $w0$ but not with $w1$, P_1 for those that are G-consistent with $w1$ but not $w0$, and P for those that are consistent with both. Then the set $\{\, \alpha : |\alpha| = t(n),\ \alpha \sim_G w \,\}$ equals the disjoint union of P, P_0, and P_1. Furthermore, for $\alpha \in P_0$ we have $m(\alpha, w0) = m(\alpha, w) + 1$, and similarly for P_1, while for $\alpha \in P$ we have $m(\alpha, w0) = m(\alpha, w1) = m(\alpha, w)$. Hence $d_G(w0) + d_G(w1)$ is given by:

$$\sum_{\alpha \in P \cup P_0} c_G(\alpha) 2^{m(\alpha,w0)-t(n)} + \sum_{\alpha \in P \cup P_1} c_G(\alpha) 2^{m(\alpha,w1)-t(n)}$$

$$= \sum_{\alpha \in P_0} c_G(\alpha) 2^{m(\alpha,w0)-t(n)} + \sum_{\alpha \in P_1} c_G(\alpha) 2^{m(\alpha,w1)-t(n)} + 2 \sum_{\alpha \in P} c_G(\alpha) 2^{m(\alpha,w)-t(n)}$$

$$= 2 \sum_{\alpha \in P_0} c_G(\alpha) 2^{m(\alpha,w)-t(n)} + 2 \sum_{\alpha \in P_1} c_G(\alpha) 2^{m(\alpha,w)-t(n)} + 2 \sum_{\alpha \in P} c_G(\alpha) 2^{m(\alpha,w)-t(n)}$$

$$= 2 d_G(w).$$

Finally, if $|w| + 1$ is a power of 2, then $d_G(w0)$ and $d_G(w1)$ use $t' := t(n+1)$ for their length of α. However, by the first part of this proof, we can replace $t(n)$ by t' in the definition of $d_G(w)$ without changing its value, and then the second part goes through the same way for t'. Hence d_G is a martingale. □

It is still the case, however, that d_G may not succeed on the languages on which the betting game G succeeds. To ensure this, we first use Lemma 3.2 to place betting games G into a suitable "normal form" satisfying the sure-winnings condition (5).

Lemma 4.2 *If G is a betting game satisfying the sure-winnings condition (5), then $S^\infty[G] \subseteq S^\infty[d_G]$.*

Proof. First, let $A \in S^\infty[G]$, and fix $k > 0$. Find a finite initial segment $w \sqsubseteq A$ long enough to answer every query made in a play α of G such that $\alpha \sim_G w$ and $c_G(\alpha) \geq k + 2$; and long enough to make $t(n)$ in the definition of $d_G(w)$ (Equation 7) greater than $|\alpha|$. Then every α' of length $t(n)$ such that

$\alpha' \sim_G w$ has the form $\alpha' = \alpha\beta$. The sure-winnings condition (5) implies that the right-hand side of Equation (7) defining $d_G(w)$ is an average over terms that all have size at least k. Hence $d_G(w) \geq k$. Letting k grow to infinity gives $A \in S^\infty[d_G]$. □

Now we turn our attention to the complexity of d_G. If G is a time-$t(n)$ betting game, it is clear that d_G can be computed deterministically in $O(t(n))$ *space*, because we need only cycle through all α of length $t(n)$, and all the items in Equation (7) are computable in space $O(t(n))$. In particular, every E-betting game can be simulated by a martingale whose values are computable in deterministic space $2^{O(n)}$ (even counting the output against the space bound), and every EXP-betting game by a martingale similarly computed in space $2^{n^{O(1)}}$. However, we show in the next section that one can *estimate* $d_G(w)$ well without having to cycle through all the α, using a pseudo-random generator to "sample" only a very small fraction of them.

5 Sampling Results

First we determine the accuracy to which we need to estimate the values $d(w)$ of a hard-to-compute martingale. We state a stronger version of the result than we need in this section. In the next section, we will apply it to martingales whose "activity" is restricted to subsets J of $\{0,1\}^*$ in the following sense: for all strings $x \notin J$, and all w such that $s_{|w|+1} = x$, $d(w0) = d(w1) = d(w)$. Intuitively, a martingale d is inactive on a string x if there is *no* possible "past history" w that causes a nonzero bet to be made on x. For short we say that such a d is *inactive outside* J. Recall that $N = \Theta(2^n)$.

Lemma 5.1 *Let d be a martingale that is inactive outside $J \subseteq \{0,1\}^*$, and let $[\epsilon(i)]_{i=0}^\infty$ be a non-negative sequence such that $\sum_{s_i \in J} \epsilon(i)$ converges to a number K. Suppose we can compute in time $t(n)$ a function $g(w)$ such that $|g(w) - d(w)| \leq \epsilon(N)$ for all w of length N. Then there is a martingale d' computable in time $(2^n t(n))^{O(1)}$ such that for all w, $|d'(w) - d(w)| \leq 4K + 2\epsilon(0)$.*

In this section, we will apply Lemma 5.1 with $J = \{0,1\}^*$ and $\epsilon(N) = 1/N^2 = 1/2^{2n}$. In Section 6.3 we will apply Lemma 5.1 in cases where J is finite.

Proof.

First note that for any w (with $N = |w|$),

$$\begin{aligned}\left|g(w) - \frac{g(w0)+g(w1)}{2}\right| &\leq |g(w)-d(w)| + \left|\frac{d(w0)-g(w0)}{2}\right| + \left|\frac{d(w1)-g(w1)}{2}\right| \\ &\leq \epsilon(N) + \epsilon(N+1). \qquad (9)\end{aligned}$$

In case $J = \{0,1\}^*$, we inductively define:

$$\begin{cases} d'(\lambda) &= g(\lambda) + 2K + \epsilon(0) \\ d'(wb) &= d'(w) + g(wb) - \frac{g(w0)+g(w1)}{2}. \end{cases}$$

Note that d' satisfies the average law (1), and that we can compute $d'(w)$ in time $O(2^n t(n))$.

By induction on $|w|$, we can show using the estimate provided by Equation (9) that

$$g(w) + \epsilon(N) + 2\sum_{i=N+1}^{\infty} \epsilon(i) \;\leq\; d'(w) \;\leq\; g(w) + 2\sum_{i=0}^{N-1} \epsilon(i) + \epsilon(N) + 2K.$$

It follows that

$$\begin{aligned} d'(w) &\geq g(w) + \epsilon(N) \\ &= d(w) + (g(w) - d(w)) + \epsilon(N) \geq d(w), \end{aligned}$$

and that

$$\begin{aligned} d'(w) &= d(w) + (g(w) - d(w)) + (d'(w) - g(w)) \\ &\leq d(w) + \epsilon(N) + 2\sum_{i=0}^{N-1} \epsilon(i) + \epsilon(N) + 2K \\ &\leq d(w) + 4K + 2\epsilon(0). \end{aligned}$$

This establishes the lemma in case $J = \{0,1\}^*$. The generalization to other subsets J of $\{0,1\}^*$ is left to the reader. □

Next, we specify precisely which function f_G we will sample in order to estimate d_G, and how we will do it.

Let G be a $t(n)$ time-bounded betting game. Consider a prefix w, and let n denote the largest length of a string in the domain of w. With any string ρ of length $t(n)$, we can associate a unique "play of the game" G defined by using w to answer queries in the domain of w, and the successive bits of ρ to answer queries outside it. We can stop this play after $t(n)$ steps—so that the stopped play is a $t(n)$-maximal α—and then define $f_G(w, \rho)$ to be the capital $c_G(\alpha)$. Note that we can compute $f_G(w, \rho)$ in linear time, i.e. in time $O(|w| + t(n))$. The proportion of strings ρ of length $t(n)$ that map to the same play α is exactly the weight $2^{m(\alpha,w)-|\alpha|}$ in the equation (7) for $d_G(w)$. Letting E stand for mathematical expectation, this gives us:

$$d_G(w) = E_{|\rho|=t(n)}[f_G(w, \rho)].$$

To obtain good and efficient approximations to the right-hand side, we employ *pseudo-random generators*. The following supplies all relevant definitional background.

DEFINITION 5.1 ([26]). (a) The *hardness* $H_A(n)$ of a set A at length n is the largest integer s such that for any circuit C of size at most s with n inputs,

$$\left|\Pr_x[C(x) = A(x)] - \frac{1}{2}\right| \leq \frac{1}{s},$$

where x is uniformly distributed over Σ^n.

(b) A *pseudo-random generator* is a function D that, for each n, maps Σ^n into $\Sigma^{r(n)}$ where $r(n) \geq n+1$. The function r is called the *stretching* of D.

(c) The *security* $S_D(n)$ of D at length n is the largest integer s such that for any circuit C of size at most s with $r(n)$ inputs

$$\left|\Pr_x[C(x) = 1] - \Pr_y[C(D(y)) = 1]\right| \leq \frac{1}{s},$$

where x is uniformly distributed over $\Sigma^{r(n)}$ and y over Σ^n.

We will use pseudo-random generators with the following characteristics:

(1) an E-computable pseudo-random generator D_1 that stretches seeds super-polynomially and has super-polynomial security at infinitely many lengths;

(2) an EXP-computable pseudo-random generator D_2 of security $2^{n^{\Omega(1)}}$; and

(3) an E-computable pseudo-random generator D_3 of security $2^{\Omega(n)}$.

D_1 will be applied in the next section; in this section we will use D_2 and D_3. None of these generators is known to exist unconditionally. However, a highly plausible hypothesis suffices for the weakest generator D_1, as follows simply by combining work of [6] and [26] with some padding.

Theorem 5.2 *If* MA $\neq$ EXP, *then there is an* E*-computable pseudo-random generator D_1 with stretching $n^{\Theta(\log n)}$ such that for any integer k, there are infinitely many n with $S_{D_1}(n) > n^k$.*

Proof. From the proof of Lemma 4.1 of [6], it follows that if MA $\neq$ EXP, then there is a set $A \in$ EXP such that for any integer j, there are infinitely many m such that $H_A(m) > m^j$. From the proof of the main Theorem 1 in [26], it follows that for any set $A \in$ EXP, there is an EXP-computable pseudo-random generator D with stretching $n^{\Theta(\log n)}$ such that $S_D(n) = \Omega(H_A(\sqrt{n})/n)$. Say that D is computable in time 2^{n^c} for some integer constant $c > 0$. For any $k > 0$, the infinitely many m promised above with $j = 2(ck+1)$ yield infinitely many

n of the form $m^{2/c}$ such that $S_D(n^{1/c}) > n^k$. Defining $D_1(x) = D(x')$, where x' denotes the prefix of x of length $|x|^{1/c}$, yields the required pseudo-random generator. $\square$

Exponential-time computable pseudo-random generators with exponential security have the interesting property that we can blow up the stretching exponentially without significantly reducing the security. As with Theorem 5.2, credit for this observation should be distributed among the references cited in the proof.

Theorem 5.3 *(a) Given an* EXP*-computable pseudo-random generator D_0 of security $2^{n^{\Omega(1)}}$, we can construct an* EXP*-computable pseudo-random generator D_2 of security $2^{n^{\Omega(1)}}$ and stretching $2^{n^{\Omega(1)}}$.*

(b) Given an E*-computable pseudo-random generator D_0 of security $2^{\Omega(n)}$, we can construct an* E*-computable pseudo-random generator D_3 of security $2^{\Omega(n)}$ and stretching $2^{\Omega(n)}$.*

Proof. For (a), Nisan and Wigderson [26] showed that the existence of an E-computable pseudo-random generator with stretching $n+1$ (a "quick extender" in their terminology) with security $2^{n^{\Omega(1)}}$ is equivalent to the existence of an E-computable pseudo-random generator with stretching and security $2^{n^{\Omega(1)}}$. See Statements (3) and (4) of their Main Theorem (Theorem 1) instantiated with $s(\ell) = 2^\ell$. As used in [6], their main result carries through if we replace "E-computable" by "EXP-computable" in both statements, owing to padding. Since the existence of D_0 implies the existence of an EXP-computable extender with security $2^{n^{\Omega(1)}}$, the existence of D_2 follows.

For (b), first define $D'(x)$ to be the first $|x|+1$ bits of $D_0(x)$. Then D' is an extender with security $2^{\Omega(n)}$, and this implies that the range of D' is a language in E requiring circuits of size $2^{\Omega(n)}$. Impagliazzo and Wigderson, in their proof of Theorem 2 in [16], showed how to transform such a language into a language $A \in \mathrm{E}$ such that $H_A(n) = 2^{\Omega(n)}$. Using this A in part (3) of Theorem 2 of [26] yields an E-computable pseudo-random generator D_3 of security and stretching $2^{\Omega(n)}$. (It is also possible to argue that the range of D' is sufficiently hard to employ the technique of [26], without going through [16].) $\square$

Pseudo-random generators of security $2^{n^{\Omega(1)}}$ (even polynomial-time computable ones) are fairly widely believed to exist (see [8, 28, 9]), and while those of security $2^{\Omega(n)}$ are more controversial even for EXP-computability, their existence was made more plausible by the result of [16] used in the proof of (b) above. Polynomial-time computable pseudo-random generators of security $2^{\Omega(n)}$

exist relative to a random oracle [33, 15], and E-computable ones also exist if P = NP. (The latter observation follows by combining the techniques of Kannan [19] with padding and the above-mentioned result of [16]; it is noted by the second author as "Corollary 2.2.19" in his dissertation [32].)

The following general result shows how pseudo-random generators can be used to approximate averages. It provides the accuracy and time bounds needed for applying Lemma 5.1 to get the desired martingale.

Theorem 5.4 *Let D be a pseudo-random generator computable in time $\delta(n)$ and with stretching $r(n)$. Let $f : \Sigma^* \times \Sigma^* \to (-\infty, \infty)$ be a function that is computed in linear time on a Turing machine, and let $s, R, m : \mathbf{N} \to \mathbf{N}$ be fully time-constructible functions such that $s(N) \geq N$ and the following relations hold for any integer $N \geq 0$, $w \in \Sigma^N$, and $\rho \in \Sigma^{s(N)}$:*

$$\begin{aligned} |f(w,\rho)| &\leq R(N) \\ r(m(N)) &\geq s(N) \\ S_D(m(N)) &\geq (s(N)+R(N))^6. \end{aligned} \tag{10}$$

Then we can approximate

$$h(w) = E_{|\rho|=s(N)}[f(w,\rho)] \tag{11}$$

to within N^{-2} in time $O(2^{m(N)} \cdot (s(N)+R(N))^4 \cdot \delta(m(N)))$.

Proof. For any $N \geq 0$, let $\mathcal{I}_N$ be a partition of the interval $[-R(N), R(N)]$ into subintervals of length $\frac{1}{2N^2}$. Note that $|\mathcal{I}_N| = 4N^2R(N)$. Define for any $I \in \mathcal{I}_N$ and any string w of length N,

$$\pi(I,w) = \Pr_{|\rho|=s(N)}[f(w,\rho) \in I].$$

The predicate in $[\ldots]$ can be computed by circuits of size $O(s(N)\log s(N))$, using the t-to-$O(t \log t)$ Turing-machine-time-to-circuit-size construction of Pippenger and Fischer [27]. Since $S_D(m(N)) = \omega(s(N)\log s(N))$, it follows that

$$\tilde{\pi}(I,w) = \Pr_{|\sigma|=m(N)}[f(w, D(\sigma)[1 \ldots s(N)]) \in I]$$

approximates $\pi(I,w)$ to within an additive error of $(S_D(m(N)))^{-1}$, and we can compute it in time $O(2^{m(N)} \cdot s(N) \cdot \delta(m(N)))$. We define the approximation $\tilde{h}(w)$ for $h(w)$ as

$$\tilde{h}(w) = \sum_{I \in \mathcal{I}_N} \tilde{\pi}(I,w) \min(I).$$

Since we can write $h(w)$ as

$$h(w) = \sum_{I \in \mathcal{I}_N} \pi(I,w) E_{|\rho|=s(N)}[f(w,\rho) \mid f(w,\rho) \in I],$$

and we can bound the approximation error as follows:

$$
\begin{aligned}
&|h(w) - \tilde{h}(w)| \\
&\quad \le \sum_{I \in \mathcal{I}_N} (\pi(I,w) \left| E_{|\rho|=s(N)}[f(w,\rho) \mid f(w,\rho) \in I] - \min(I) \right| \\
&\qquad + \sum_{I \in \mathcal{I}_N} \left| \pi(I,w) - \tilde{\pi}(I,w) \right| \min(I) \\
&\quad \le \max_{I \in \mathcal{I}_N}(|I|) + |\mathcal{I}_N| \cdot (S_D(m(N)))^{-1} \cdot R(N) \\
&\quad \le \frac{1}{2N^2} + 4N^2 \cdot R^2(N) \cdot (S_D(m(N)))^{-1} \le \frac{1}{N^2}.
\end{aligned}
$$

Computing $\tilde{h}(w)$ requires $|\mathcal{I}_N| = 4N^2R(N)$ evaluations of $\tilde{\pi}$, which results in the claimed upper bound for the time complexity of $\tilde{h}$. □

Now, we would like to apply Theorem 5.4 to approximate $h = d_G$ given by Equation (7) to within N^{-2}, by setting $f = f_G$ and $s(N) = t(\log N)$. However, for a general betting game G running in time $t(n)$, we can only guarantee an upper bound of $R(N) = 2^{t(\log N)} \cdot c_G(\lambda)$ on $|f(w,\rho)|$. Since S_D can be at most exponential, condition (10) would force $m(N)$ to be $\Omega(t(\log N))$. In that case, Theorem 5.4 can only yield an approximation computable in time $2^{O(t(\log N))}$. *However*, we can assume without loss of generality that G satisfies the slow-winnings condition (6) of Lemma 3.2, in which case an upper bound of $R(N) \in O(N)$ holds. Then the term $s(N)$ in the right-hand side of Equation (10) dominates, provided $t(n) = 2^{\Omega(n)}$.

Taking everything together, we obtain the following result about transforming E- and EXP-betting games into equivalent E- respectively EXP-martingales:

Theorem 5.5 *If there is a pseudo-random generator computable in* E *with security $2^{\Omega(n)}$, then for every* E*-betting game G, there exists an* E*-martingale d such that $S^\infty[G] \subseteq S^\infty[d]$. If there is a pseudo-random generator computable in* EXP *with security $2^{n^{\Omega(1)}}$, then for every* EXP*-betting game G, there exists an* EXP*-martingale d such that $S^\infty[G] \subseteq S^\infty[d]$.*

Proof. By Lemma 3.2, we can assume that c_G satisfies both the sure-winnings condition (5) as well as the slow-winnings condition (6). Because of Lemma 4.2 and Lemma 5.1 (since the series $\sum_{i=1}^{\infty} \frac{1}{i^2}$ converges), it suffices to approximate the function $d_G(w)$ given by Equation (7) to within N^{-2} in time $2^{O(n)}$ respectively $2^{n^{O(1)}}$, where $N = |w|$ and $n = \log N$.

Under the given hypothesis about the existence of an E-computable pseudo-random generator D_0, we can take D to be the pseudo-random generator D_3

provided by Theorem 5.3(b). Thus we meet the conditions for applying Theorem 5.4 to $h = d_G$ with $s(N) = N^{O(1)}$, $R(N) = O(N)$, and $m(N) = O(\log N)$, and we obtain the approximation of d_G that we need. In the case of an EXP-betting game G, to obtain an EXP-martingale we can take D to be the pseudo-random generator D_2 of weaker security guarantee $2^{n^{\Omega(1)}}$ provided by Theorem 5.3(a). Then we meet the requirements of Theorem 5.4 with $s(N) = 2^{(\log N)^{O(1)}}$, $R(N) = O(N)$, and $m(N) = (\log N)^{O(1)}$. $\square$

6 Autoreducible Sets

An oracle Turing machine M is said to *autoreduce* a language A if $L(M^A) = A$, and for all strings x, M^A on input x does not query x. That is, one can learn the membership of x by querying strings other than x itself. If M runs in polynomial time, then A is P*-autoreducible*—we also write $\leq_T^p$-autoreducible. If M is also non-adaptive, then A is $\leq_{tt}^p$*-autoreducible*.

One can always code M so that for *all* oracles, it *never* queries its own input—then we call M an *autoreduction*. Hence we can define an effective enumeration $[M_i]_{i=1}^{\infty}$ of polynomial-time autoreductions, such that a language A is autoreducible iff there exists an i such that $L(M_i^A) = A$. (For a technical aside: the same M_i may autoreduce different languages A, and some M_i may autoreduce no languages at all.) The same goes for $\leq_{tt}^p$*-autoreductions*.

Autoreducible sets were brought to the polynomial-time context by Ambos-Spies [3]. Their importance was further argued by Buhrman, Fortnow, Van Melkebeek, and Torenvliet [11], who showed that all $\leq_T^p$-complete sets for EXP are $\leq_T^p$-autoreducible (while some complete sets for other classes are not). Here we demonstrate that autoreducible sets are important for testing the power of resource-bounded measure.

6.1 Adaptively Autoreducible Sets

As stated in the Introduction, if the $\leq_T^p$-autoreducible sets in EXP (or sufficiently the $\leq_T^p$-complete sets for EXP) are covered by an EXP-martingale, then EXP $\neq$ BPP, a non-relativizing consequence. However, it is easy to cover them by an E-betting game. Indeed, the betting game uses its adaptive freedom only to "look ahead" at the membership of lexicographically greater strings, betting nothing on them.

Theorem 6.1 *There is an* E*-betting game G that covers all $\leq_T^p$-autoreducible languages.*

Proof. Let $M_1, M_2, \ldots$ be an enumeration of $\leq_T^p$-autoreductions such that each M_i runs in time $n^i + i$ on inputs of length n. Our betting game G regards its capital as composed of infinitely many "shares" c_i, one for each M_i. Initially, $c_i = 1/2^i$. Letting $\langle \cdot, \cdot \rangle$ be a standard pairing function, inductively define $n_0 = 0$ and $n_{\langle i,j \rangle +1} = (n_{\langle i,j \rangle})^i + i$.

During a stage $s = \langle i, j \rangle$, G simulates M_i on input $0^{n_{s-1}}$. Whenever M_i makes a query of length less than n_{s-1}, G looks up the answer from its table of past queries. Whenever M_i makes a query of length n_{s-1} or more, G places a bet of zero on that string and makes the same query. Then G bets all of the share c_i on $0^{n_{s-1}}$ according to the answer of the simulation of M_i. Finally, G "cleans up" by putting zero bets on all strings with length in $[n_{s-1}, n_s)$ that were not queries in the previous steps.

If M_i autoreduces A, then share c_i doubles in value at each stage $\langle i, j \rangle$, and makes the total capital grow to infinity. And G runs in time $2^{O(n)}$—indeed, only the "cleanup" phase needs this much time. $\square$

Corollary 6.2 *Each of the following statements implies* BPP $\neq$ EXP*:*

1. *The class of $\leq_T^p$-autoreducible sets has* E*-measure zero.*
2. *The class of $\leq_T^p$-complete sets for* EXP *has* E*-measure zero.*
3. E*-betting games and* E*-martingales are equivalent.*
4. E*-betting games have the finite union property.*

The same holds if we replace E *by* EXP *in these statements.*

Proof. Let $\mathcal{C}$ stand for the class of languages that are not $\leq_T^p$-hard for BPP. Allender and Strauss [1] showed that $\mathcal{C}$ has E-measure zero, so trivially it is also covered by an E-betting game. Now let $\mathcal{D}$ stand for the class of $\leq_T^p$-complete sets for EXP. By Theorem 6.1 and the result of [11] cited above, $\mathcal{D}$ is covered by an E-betting game.

If EXP = BPP, the union $\mathcal{C} \cup \mathcal{D}$ contains all of EXP, and:

- If $\mathcal{D}$ would have E-measure zero, so would $\mathcal{C} \cup \mathcal{D}$ and hence EXP, contradicting the measure conservation property of Lutz measure.
- If E-betting games would have the finite-union property, then $\mathcal{C} \cup \mathcal{D}$ and EXP would be covered by an E-betting game, contradicting Theorem 3.1.

Since Equation (1) implies (2), and Equation (3) implies (4), these observations suffice to establish the corollary for E. The proof for EXP is similar. $\square$

Since there is an oracle A giving $\mathrm{EXP}^A = \mathrm{BPP}^A$ [14], this shows that relativizable techniques cannot establish the equivalence of E-martingales and E-betting games, nor of EXP-martingales and EXP-betting games. They cannot refute it either, since there are oracles relative to which strong pseudo-random generators exist—all "random" oracles, in fact [33].

6.2 Non-Adaptively Autoreducible Sets

It is tempting to think that the *non*-adaptively P-autoreducible sets should have E-measure zero, or at least EXP-measure zero, insofar as betting games are the adaptive cousins of martingales. However, it is not just adaptiveness but also the freedom to bet *out of the fixed lexicographic order* that adds power to betting games. If one carries out the proof of Theorem 6.1 to cover the class of $\leq^p_{tt}$-autoreducible sets, using an enumeration $[M_i]$ of $\leq^p_{tt}$-autoreductions, one obtains a *non-adaptive* E-betting game (defined formally below) that (independent of its oracle) bets on all strings in order given by a single permutation of Σ^*. The permutation itself is E-computable. It might seem that an E-martingale should be able to "un-twist" the permutation and succeed on all these sets. However, our next results, which strengthen the above corollary, close the same "non-relativizing" door on proving this with current techniques.

Theorem 6.3 *For any $k \geq 1$, the $\leq^p_{tt}$-complete languages for Δ^p_k are $\leq^p_{tt}$-autoreducible.*

Here is the proof idea, which follows techniques of [11] for the theorem that all EXP-complete sets are $\leq^p_T$-autoreducible. Call a closed propositional formula that has at most k blocks of like quantifiers (i.e., at most $k-1$ quantifier alternations) a "QBF$_k$ formula," and let $TQBF_k$ stand for the set of true QBF formulas. Let A be a $\leq^p_{tt}$-complete set for $\Delta^p_{k+1} = \mathrm{P}^{\Sigma^p_k}$. Since $TQBF_k$ is Σ^p_k-hard, there is a deterministic polynomial-time oracle Turing machine M that accepts A with oracle $TQBF_k$. Let $q(x,i)$ stand for the i-th oracle query made by M on input x. Whether $q(x,i)$ belongs to $TQBF_k$ forms a Δ^p_{k+1}-question, so we can $\leq^p_{tt}$-reduce it to A. It is possible that this latter reduction will include x itself among its queries. Let b^+_i denote the answer it gives to the question provided that any query to x is answered "yes," and similarly define b^-_i in case x is answered "no."

If $b^+_i = b^-_i$, which holds in particular if x is not queried, then we know the correct answer b_i to the i-th query. If this situation occurs for all queries, we are done: We just have to run M on input x using the b_i's as answers to the oracle queries. The b_i's themselves are obtained *without* submitting the (possibly adaptive) queries made by M, but rather by applying the latter $\leq^p_{tt}$-reduction to A to the pair $\langle x,i\rangle$, and without submitting any query on x itself. Hence this process satisfies the requirements of a $\leq^p_{tt}$-autoreduction of A for the particular input x.

Now suppose that $b^+_i \neq b^-_i$ for some i, and let i be minimal. Then we will have two players play the k-round game underlying the QBF$_k$-formula that constitutes the i-th oracle query. One player claims that b^+_i is the correct value for b_i, which is equivalent to claiming that $x \in A$, while his opponent claims that b^-_i is correct and that $x \notin A$. Write $\chi_A(x) = 1$ if $x \in A$, and $\chi_A(x) = 0$ if $x \notin A$. The players' strategies will consist of computing the game history so far, determining their optimal next move, $\leq^p_{tt}$-reducing this computation to A, and finally producing the result of this reduction under their respective assumption about $\chi_A(x)$. This approach will allow us to recover the game history in polynomial time with non-adaptive queries to A different from x. Moreover, it will guarantee that the player making the correct assumption about $\chi_A(x)$ plays optimally. Since this player is also the one claiming the correct value for b_i, he will win the game. So, we output the winner's value for b_i.

It remains to show that we can compute the above strategies in deterministic polynomial time with a Σ_k^p oracle, i.e. in $\mathrm{FP}^{\Sigma_k^p}$. It seems crucial that the number k of alternations be constant here.

Proof. (of Theorem 6.3) Let A be a $\leq_{tt}^p$-complete set for Δ_{k+1}^p accepted by the polynomial-time oracle Turing machine M with oracle $TQBF_k$. Let $q(x,i)$ denote the i-th oracle query of M^{TQBF_k} on input x. Then $q(x,i)$ can be written in the form $(\exists y_1)(\forall y_2)\ldots(Q_k y_k)\ \phi_{x,i}(y_1, y_2, \ldots, y_k)$, where $y_1, \ldots, y_k$ stand for the vectors of variables quantified in each block, or in the opposite form beginning with the block $(\forall y_1)$. By reasonable abuse of notation, we also let y_r stand for a string of 0-1 assignments to the variables in the r-th block. Without loss of generality, we may suppose every oracle query made by M has this form where each y_j is a string of length $|x|^c$, and M makes exactly $|x|^c$ queries, taking the constant c from the polynomial time bound on M. Note that the function q belongs to $\mathrm{FP}^{\Sigma_k^p}$. Hence the language

$$L_0 = \{\, \langle x, y\rangle : q(x,i) \in TQBF_k \,\}$$

belongs to Δ_{k+1}^p. Since A is $\leq_{tt}^p$-complete for Δ_{k+1}^p, there is a polynomial-time nonadaptive oracle Turing machine N_0 that accepts L_0 with oracle A. Now define $b_i^+(x) = N_0^{A\cup\{x\}}(\langle x,i\rangle)$ and $b_i^-(x) = N_0^{A\setminus\{x\}}(\langle x,i\rangle)$. We define languages $L_1, L_2, \ldots, L_k \in \Delta_{k+1}^p$ and $\leq_{tt}^p$-reductions $N_1, N_2, \ldots, N_k$ inductively as follows:

Let $1 \leq \ell \leq k$. The set L_ℓ consists of all pairs $\langle x, j\rangle$ with $1 \leq j \leq |x|^c$, such that there is a smallest i, $1 \leq i \leq |x|^c$, for which $b_i^+(x) \neq b_i^-(x)$, and the following condition holds. For $1 \leq r \leq \ell - 1$, let the s-th bit of y_r equal $N_r^{A\cup\{x\}}(\langle x,s\rangle)$ if $r \equiv b_i^+(x) \bmod 2$, and $N_r^{A\setminus\{x\}}(\langle x,s\rangle)$ otherwise. We put $\langle x,j\rangle$ into L_ℓ iff there is a lexicographically least y_ℓ such that

$$\chi[(Q_{\ell+1}y_{\ell+1})(Q_{\ell+2}y_{\ell+2})\ldots(Q_k y_k)\ \phi_{x,i}(y_1, y_2, \ldots, y_k)] \equiv \ell \bmod 2,$$

and the j-th bit of y_ℓ is set to 1. The form of this definition shows that L_ℓ belongs to Δ_{k+1}^p. Hence we can take N_ℓ to be a polynomial-time non-adaptive oracle Turing machine that accepts L_ℓ with oracle A.

Now, we construct a $\leq_{tt}^p$-autoreduction for A. On input x, we compute $b_i^+(x)$ and $b_i^-(x)$ for $1 \leq i \leq |x|^c$, as well as $y_r^{(b)}$ for $b \in \{0,1\}$ and $1 \leq r \leq |x|^c$. The latter quantity $y_r^{(b)}$ is defined as follows: for $1 \leq s \leq |x|^c$, the s-th bit of $y_r^{(b)}$ equals $N_r^{A\cup\{x\}}(\langle x,s\rangle)$ if $r \equiv b \bmod 2$, and $N_r^{A\setminus\{x\}}(\langle x,s\rangle)$ otherwise. Note that we can compute all these values in polynomial time by making non-adaptive queries to A none of which equals x.

If $b_i^+(x) = b_i^-(x)$ for every $1 \leq i \leq |x|^c$, we run M on input x using $b_i^+(x) = b_i^-(x)$ as the answer to the i-th oracle query. Since it always holds that at least one of $b_i^+(x)$ and $b_i^-(x)$ equals the correct oracle answer $b_i(x)$, we faithfully simulate M on input x, and hence compute $\chi_A(x)$ correctly.

Otherwise, let i be the first index for which $b_i^+(x) \neq b_i^-(x)$. Since $b_j(x) = b_j^+(x) = b_j^-(x)$ for $j < i$, we can determine $q(x,i)$ by simulating M on input x until it asks the i-th query. We then output 1 if

$$b_i^+(x) = \phi_{x,i}(y_1^{(b_i^+(x))}, y_2^{(b_i^+(x))}, \ldots, y_k^{(b_i^+(x))}),$$

and output 0 otherwise. We claim that this gives the correct answer to whether $x \in A$.

In order to prove the claim, consider the game history $y_1^{(b_i^+(x))}$, $y_2^{(b_i^+(x))}$, $\ldots$, $y_k^{(b_i^+(x))}$. The player claiming the correct value for $b_i(x)$ gets to play the rounds that allow him to win the game no matter what his opponent does. Since this player is also the one making the correct assumption about $\chi_A(x)$, an inductive argument shows that he plays optimally: At his stages ℓ, the string y_ℓ in the above construction of L_ℓ exists, and he plays it. The key for the induction is that at later stages $\ell' > \ell$, the value of y_ℓ at stage ℓ' remains the same as what it was at stage ℓ. Thus the player with the correct assumption about $\chi_A(x)$ wins the game—that is, $\phi_{x,i}(y_1^{(b_i^+(x))}, y_2^{(b_i^+(x))}, \ldots, y_k^{(b_i^+(x))})$ equals his guess for $b_i(x)$ (and not the other player's guess). $\square$

In order to formalize the strengthening of Corollary 6.2 that results from Theorem 6.3, we call a betting game G *non-adaptive* if the infinite sequence $x_1, x_2, x_3, \ldots$ of queries G^A makes is the same for all oracles A. If G runs in $2^{O(n)}$ time, and this sequence hits all strings in Σ^*, then the permutation π of the standard ordering $s_1, s_2, s_3, \ldots$ defined by $\pi(s_i) = x_i$ is both computable and invertible in $2^{O(n)}$ time. It is computable in this amount of time because in order to hit all strings, G must bet on all strings in $\{0,1\}^n$ within the first $2^{O(n)}$ steps. Hence its ith bet must be made in a number of steps that is singly-exponential in the length of s_i. And to compute $\pi^{-1}(x_i)$, G need only be run for $2^{O(|x_i|)}$ steps, since it cannot query x_i after this time. Since π and its inverse are both E-computable, π is a reasonable candidate to replace lexicographic ordering in the definition of E-martingales, and likewise for EXP-martingales. We say a class $\mathcal{C}$ has π-E-measure zero if $\mathcal{C}$ can be covered by an E-martingale that interprets its input as a characteristic string *in the order given by* π.

Theorem 6.4 *The class of $\leq_{tt}^p$-autoreducible languages can be covered by a non-adaptive* E*-betting game. Hence there is an* E*-computable and invertible permutation π of Σ^* such that this class has π-*E*-measure zero.*

Proof. With reference to the proof of Theorem 6.1, we can let $M_1, M_2, \ldots$ be an enumeration of $\leq_{tt}^p$-autoreductions such that each M_i runs in time $n^i + i$. The machine G in that proof automatically becomes non-adaptive, and since it queries all strings, it defines a permutation π of Σ^* as above with the required properties. $\square$

Corollary 6.5 *Each of the following statements implies* BPP $\neq$ EXP, *as do the statements obtained on replacing "*E*" by "*EXP*."*

1. *The class of $\leq_{tt}^p$-autoreducible sets has* E*-measure zero.*
2. *The class of $\leq_{tt}^p$-complete sets for* EXP *has* E*-measure zero.*
3. *Non-adaptive* E*-betting games and* E*-martingales are equivalent.*
4. *If two classes can be covered by non-adaptive* E*-betting games, then their union can be covered by an* E*-betting game.*
5. *For all classes $\mathcal{C}$ and all* E*-computable and invertible orderings π, if $\mathcal{C}$ has π-*E*-measure zero, then $\mathcal{C}$ has* E*-measure zero.*

Proof. It suffices to make the following two observations to argue that the proof of Corollary 6.2 carries over to the truth-table cases:

- The construction of Allender and Strauss [1] actually shows that the class of sets that are not $\leq_{tt}^p$-hard for BPP has E-measure zero.
- If EXP = BPP, Theorem 6.3 implies that all $\leq_{tt}^p$-complete sets for EXP are $\leq_{tt}^p$-autoreducible, because BPP $\subseteq \Sigma_2^{\mathrm{p}} \subseteq \Delta_3^{\mathrm{p}} \subseteq$ EXP.

Theorem 6.4 and the finite-unions property of Lutz's measures on E and EXP do the rest. $\square$

The last point of Corollary 6.5 asserts that Lutz's definition of measure on E is invariant under all E-computable and invertible permutations. These permutations include *flip* from the Introduction and (crucially) π from Theorem 6.4. Hence this robustness assertion for Lutz's measure implies BPP $\neq$ EXP. Our "betting-game measure" (both adaptive and non-adaptive) does enjoy this permutation invariance, but asserting the finite-unions property for it also implies BPP $\neq$ EXP. The rest of this paper explores conditions under which Lutz's martingales *can* cover classes of autoreducible sets, thus attempting to narrow the gap between them and betting games.

6.3 Covering Autoreducible Sets By Martingales

This puts the spotlight on the question: Under what hypotheses can we show that the $\leq_{tt}^{p}$-autoreducible sets have E-measure zero? Any such hypothesis must be strong enough to imply EXP $\neq$ BPP, but we hope to find hypotheses weaker than assuming the equivalence of (E- or EXP-) betting games and martingales, or assuming the finite-union property for betting games. Do we need strong pseudo-random generators to cover the $\leq_{tt}^{p}$-autoreducible sets? How close can we come to covering the $\leq_{T}^{p}$-autoreducible sets by an E-martingale?

Our final results show that the hypothesis MA $\neq$ EXP suffices. This assumption is only known to yield pseudo-random generators of super-polynomial security (at infinitely many lengths) rather than exponential security (at almost all lengths). Recall that MA contains both BPP and NP; in fact it is sandwiched between $\mathrm{NP}^{\mathrm{BPP}}$ and $\mathrm{BPP}^{\mathrm{NP}}$.

Theorem 6.6 *If* MA $\neq$ EXP, *then the class of* $\leq_{tt}^{p}$*-autoreducible sets has* E-*measure zero.*

We actually obtain a stronger conclusion.

Theorem 6.7 *If* MA $\neq$ EXP, *then the class of languages A autoreducible by polynomial-time oracle Turing machines that always make their queries in lexicographic order has* E-*measure zero.*

To better convey the essential sampling idea, we prove the weaker Theorem 6.6 before the stronger Theorem 6.7. The extra wrinkle in the latter theorem is to use the pseudo-random generator *twice*, to construct the set of "critical strings" to bet on as well as to compute the martingale.

Proof. (of Theorem 6.6) Let $[M_i]_{i=1}^{\infty}$ enumerate the $\leq_{tt}^{p}$-autoreductions, with each M_i running in time n^i. Divide the initial capital into *shares* $s_{i,m}$ for $i, m \geq 1$, with each $s_{i,m}$ valued initially at $(1/m^2)(1/2^i)$. For each share $s_{i,m}$, we will describe a martingale that is active only on a finite number of strings x. The martingale will be active only if $i \leq m/2\lceil \log_2 m \rceil$ and $m \leq |x| \leq m^i$, and further only if x belongs to a set $J = J_{i,m}$ constructed below. Hence the martingale will be inactive outside J, and we will be able to apply Lemma 5.1. We will arrange that whenever M_i autoreduces A, there are infinitely many m such that share $s_{i,m}$ attains a value above 1 (in fact, close to m) along A. Hence the martingale defined by all the shares succeeds on A. We will also ensure that each active share's bets on strings of length n are computable in time 2^{an}, where the constant a is independent of i. This is enough to make the whole martingale E-computable and complete the proof.

To describe the betting strategy for $s_{i,m}$, first construct a set $I = I_{i,m}$ starting with $I = \{ 0^m \}$ and iterating as follows: Let y be the lexicographically least string of length m that does not appear among queries made by M_i on inputs $x \in I$. Then add y to I. Do this until I has $3\lceil \log_2 m \rceil$ strings in it. This is possible because the bound $3\lceil \log_2 m\rceil m^i$ on the number of queries M_i could possibly make on inputs in I is less than 2^m. Moreover, 2^m bounds the time needed to construct I. Thus we have arranged that

$$\text{for all } x, y \in I \text{ with } x < y,\ M_i(x) \text{ does not query } y. \tag{12}$$

Now let J stand for I together with all the queries M_i makes on inputs in I. Adapting ideas from Definition 4.1 to this context, let us define a finite Boolean function $\beta : J \to \{0,1\}$ to be *consistent with* M_i *on* I, written $\beta \sim_I M_i$, if for all $x \in I$, M_i run on input x with oracle answers given by β agrees with the value $\beta(x)$. Given a characteristic prefix w, also write $\beta \sim w$ if $\beta(x)$ and $w(x)$ agree on all x in J and the domain of w. Since I and J depend only on i and m, we obtain a "probability density" function for each share $s_{i,m}$ via

$$\pi_{i,m}(w) = \Pr_{\beta\sim w}[\beta \sim_I M_i]. \tag{13}$$

The martingale $d_{i,m}$ standardly associated to this density (as in [23]) is definable inductively by $d_{i,m}(\lambda) = 1$ and

$$d_{i,m}(w1) = d_{i,m}(w)\frac{\pi_{i,m}(w1)}{\pi_{i,m}(w)}, \qquad d_{i,m}(w0) = d_{i,m}(w)\frac{\pi_{i,m}(w0)}{\pi_{i,m}(w)}. \tag{14}$$

(In case $\pi_{i,m} = 0$, we already have $d_{i,m}(w) = 0$, and so both $d_{i,m}(w1)$ and $d_{i,m}(w0)$ are set to 0.)

Note that the values $\pi_{i,m}(wb)$ for $b = 0,1$ can only differ from $\pi_{i,m}(w)$ if the string x indexed by b belongs to J; i.e., $d_{i,m}$ is inactive outside J.

Claim 6.8 *If M_i autoreduces A, then for all sufficiently large m, if share $s_{i,m}$ could play the strategy $d_{i,m}$, then on A its value would rise to (at least) $m/2^i$. That is, $s_{i,m}$ would multiply its initial value by (at least) m^3.*

To see this, first note that for any $w \sqsubseteq A$ long enough to contain J in its domain, $\pi_{i,m}(w) = 1$. We want to show that for any v short enough to have domain disjoint from I, $\pi_{i,m}(v) = 1/2^{|I|}$. To do this, consider any fixed 0-1 assignment β_0 to strings in $J\setminus I$ that agrees with v. This assignment determines the computation of M_i on the lexicographically first string $x \in I$, using β_0 to answer queries, and hence forces the value of $\beta(x)$ in order to maintain consistency on I. This in turn forces the value $\beta(x')$ on the next string x' in I, and so on. Hence only one out of $2^{|I|}$ possible completions of β_0 to β is consistent with M_i on I. Thus $\pi_{i,m}(v) = 1/2^{|I|}$. Since $d_{i,m}(w) = d_{i,m}(v)\cdot(\pi_{i,m}(w)/\pi_{i,m}(v))$ by Equation (14), and $2^{|I|} = 2^{3\lceil \log_2 m\rceil} \geq m^3$, Claim 6.8 is proved.

The main obstacle now is that $\pi_{i,m}$ in Equation (13), and hence $d_{i,m}(w)$, may not be computable in time 2^{an} with a independent of i. The number of assignments β to count is on the order of $2^{|J|} \approx 2^{m^i} \approx 2^{n^i}$. Here is where we use the E-computable pseudo-random generator D_1, with super-polynomial stretching and with super-polynomial security at infinitely many lengths, obtained via Theorem 5.2 from the hypothesis MA $\neq$ EXP. For all i and sufficiently large m, D_1 stretches a seed s of length m into at least $3\lceil\log_2 m\rceil m^i$ bits, which are enough to define an assignment β_s to J (agreeing with any given w). We estimate $\pi_{i,m}(w)$ by

$$\hat{\pi}_{i,m}(w) = \Pr_{|s|=m}[\beta_s \sim_I M_i]. \tag{15}$$

Take $\epsilon = 1/m^{i+4}$. By Theorem 5.2 there are infinitely many "good" m such that $S_{D_1}(m) > m^{i+4}$.

Claim 6.9 *For all large enough good* m, every *estimate* $\hat{\pi}_{i,m}(w)$ *gives* $|\hat{\pi}_{i,m}(w) - \pi_{i,m}(w)| \leq \epsilon$.

Suppose not. First note that Equations (13) and (15) do not depend on all of w, just on the up-to-$3\lceil\log_2 m\rceil m^i < m^{i+1}$ bits in w that index strings in J, and these can be hard-wired into circuits. The tests $[\beta \sim_I M_i]$ can also be done by circuits of size $o(m^{i+1})$, because a Turing machine computation of time r can be simulated by circuits of size $O(r \log r)$ [27]. Hence we get circuits of size less than $S_{D_1}(m)$ achieving a discrepancy greater than $1/S_{D_1}(m)$, a contradiction. This proves Claim 6.9.

Finally, observe that the proof of Claim 6.8 gives us not only $d_{i,m}(w) \geq \pi_{i,m}(w) \cdot m^3$, but also $d_{i,m}(w) = \Theta(\pi_{i,m}(w) \cdot m^3)$, when $w \sqsubseteq A$. For $w \sqsubseteq A$ and good m, we thus obtain estimates $g(w)$ for $d_{i,m}(w)$ within error bounds $\epsilon' = \Theta(\epsilon) = \Theta(1/m^{i+1})$. Now applying Lemma 5.1 for this $g(w)$ and $J = J_{i,m}$ yields a martingale $d'_{i,m}(w)$ computable in time 2^{an}, where the constant a is independent of i. This $d'_{i,m}(w)$ is the martingale computed by the actions of share $s_{i,m}$. Since $K = \sum_{s_i \in J} \epsilon' = |J|\epsilon' \leq (1/m)\cdot 3\lceil\log_2 m\rceil = o(1)$, we actually obtain $|d'_{i,m}(w) - d_{i,m}(w)| = o(1)$, which is stronger than what we needed to conclude that share $s_{i,m}$ returns enough profit. This completes the proof of Theorem 6.6. $\square$

To prove Theorem 6.7, we need to construct sets $I = I_{i,m}$ with properties similar to Equation (12), in the case where M_i is no longer a $\leq_{tt}^p$-autoreduction, but makes its queries in lexicographic order. To carry out the construction of I, we use the pseudorandom generator D_1 a second time, and actually need only that M_i on input 0^m makes all queries of length $< m$ before making any query of length $\geq m$. To play the modified strategy for share $s_{i,m}$, however, appears to require that all queries observe lexicographic order.

Proof. (of Theorem 6.7). Recall that the hypothesis EXP $\neq$ MA yields a pseudo-random generator D_1 computable in time $2^{O(m)}$ and stretching m bits to $r(m)$ bits such that for all i, all sufficiently large m give $r(m) > m^i$, and infinitely many m give hardness $S_{D_1}(m) > m^i$. Let $[M_i]_{i=1}^{\infty}$ be a standard enumeration of $\leq_T^p$-autoreductions that are constrained to make their queries in lexicographic order, with each M_i running in time $O(n^i)$. We need to define strategies for "shares" $s_{i,m}$ such that whenever M_i autoreduces A, there are infinitely many m such that share $s_{i,m}$ grows its initial capital from $1/m^2 2^i$ to $1/2^i$ or more. The strategy for $s_{i,m}$ must still be computable in time 2^{am} where a is independent of i.

To compute the strategy for $s_{i,m}$, we note first that $s_{i,m}$ can be left inactive on strings of length $< m$. The overall running time allowance $2^{O(m)}$ permits us to suppose that by the time $s_{i,m}$ becomes active and needs to be considered, the initial segment w_0 of A (where A is the language on which the share happens to be playing) that indexes strings of length up to $m-1$ is known. Hence we may regard w_0 as fixed. For any $\alpha \in \{0,1\}^{m^i}$ let $M_i^\alpha(x)$ stand for the computation in which w_0 is used to answer any queries of length $< m$ and α is used to answer all other queries. Because of the order in which M_i makes its queries, those queries y answered by w_0 are the same for all α, so that those answers can be coded by a string u_0 of length at most m^i. Now for any string y of length equal to m, define

$$P(x,y) = \Pr_\alpha[M_i^\alpha(x) \text{ queries } y].$$

Note that given u_0 and α, the test "$M_i^\alpha(x)$ queries y" can be computed by circuits of size $O(m^{i+1})$. Hence by using the pseudo-random generator D_1 at length m, we can compute uniformly in E an approximation $P_1(x,y)$ for $P(x,y)$ such that for infinitely many m, said to be "good" m, *all* pairs x, y give $|P_1(x,y) - P(x,y)| \leq \epsilon_m$, where we choose $\epsilon_m = 1/m^4$.

Here is the algorithm for constructing $I = I_{i,m}$. Start with $I := \emptyset$, and while $|I| < 3\log_2 m$, do the following: Take the lexicographically least string $y \in \Sigma^m \setminus I$ such that for all $x \in I$, $P_1(x,y) \leq \epsilon_m$. The search for such a y will succeed within $|I| \cdot m^{i+4}$ trials, since for any particular x, there are fewer than m^{i+4} strings y overall that will fail the test. (This is so even if m is not good, because it only involves P_1, and because P_1 involves simulating $M_i^{D_1(s)}$ over all seeds s.) There is enough room to find such a y provided $|I| m^{i+4} \leq 2^m$, which holds for all sufficiently large m. The whole construction of I can be completed within time 2^{2am}. It follows that for any sufficiently large good m and $x, y \in I$ with $x < y$, $\Pr_\alpha[M_i^\alpha(x) \text{ queries } y] < 2\epsilon_m = 2/m^4$.

At this point we would like to define J to be "I together with the set of strings queried by M_i on inputs in I" as before, but unlike the previous case where M_i was non-adaptive, this is not a valid definition. We acknowledge the dependence of the strings queried by M_i on the oracle A by defining

$$J_A := I \cup \{\, y : (\exists x \in I) M_i^A(x) \text{ queries } y \,\}.$$

Let $r = m^i \cdot \lceil 3\log m \rceil$. Then $|J_A| \leq r$; that is, J_A has the same size as J in the previous proof. This latter definition will be OK *because* M_i makes its queries in lexicographic order. Hence the share $s_{i,m}$, having already computed I without any reference to A, can determine the strings in J_A on which it should be active on the fly, in lexicographic order. Thus we can well-define a mapping β from $\{0,1\}^r$ to $\{0,1\}$ so that for any $k \leq r$, $\beta(k) = 1$ means that the query string y that happens to be kth in order in the on-the-fly construction of J_A is answered "yes" by the oracle. Then we may write J_β for J_A, and then write $\beta(y) = 1$ in place of $\beta(k) = 1$. Most important, given any $x \in I$, every such β well-defines a computation $M_i^\beta(x)$. This entitles us to carry over the two "consistency" definitions from the proof of Theorem 6.6:

- $\beta \sim w$ if $\beta(y) = w(y)$ for all $y \in J_\beta$;
- $\beta \sim_I M_i$ if for all $x \in I$, $M_i^\beta(x)$ equals (i.e., "agrees with") $\beta(x)$.

Finally, we may apply the latter notion to initial subsets of I, and define for $1 \leq \ell \leq 3\log m$ the predicate

$$\begin{aligned} R_\ell(\beta) \quad &= \quad (\beta \sim_{x_1,\ldots,x_\ell} M_i) \;\wedge \\ &\quad\quad (\forall j,k : 1 \leq j \leq k \leq \ell) M_i^\beta(x_j) \text{ does not query } x_k. \end{aligned}$$

Claim 6.10 *For all ℓ, $\Pr_\beta[R_\ell(\beta)] \leq 1/2^\ell$.*

For the base case $\ell = 1$, $\Pr_\beta[R_1(\beta)] = 1/2$, because $M_i(x)$ does not query x_1, M_i being an autoreduction, and because whether $\beta \sim_{x_1} M_i$ depends only on the bit of β corresponding to x_1. Working by induction, suppose $\Pr_\beta[R_{\ell-1}(\beta)] \leq 1/2^{\ell-1}$. If $R_{\ell-1}(\beta)$ holds, then taking β' to be β with the bit corresponding to x_ℓ flipped, $R_{\ell-1}(\beta')$ also holds. However, at most one of $R_\ell(\beta)$ and $R_\ell(\beta')$ holds, again because $M_i(x_\ell)$ does not query x_ℓ. Hence $\Pr_\beta[R_\ell(\beta)] \leq (1/2)\Pr_\beta[R_{\ell-1}(\beta)]$, and this proves Claim 6.10. (It is possible that neither $R_\ell(\beta)$ nor $R_\ell(\beta')$ holds, as happens when $M_i^\beta(x_j)$ queries x_ℓ for some j, but this does not hurt the claim.)

Now we can rejoin the proof of Theorem 6.6 at Equation (13), defining the probability density function $\pi_{i,m}(w) = \Pr_{\beta\sim w}[\beta \sim_I M_i]$. We get a martingale $d_{i,m}$ from $\pi_{i,m}$ as before, and this represents an "ideal" strategy for share $s_{i,m}$ to play. The statement corresponding to Claim 6.8 is:

Claim 6.11 *If M_i autoreduces A and m is good and sufficiently large, then the ideal strategy for share $s_{i,m}$ multiplies its value by at least $m^3/2$ along A.*

To see this, note that we constructed $I = \{x_1, \ldots, x_{3\log m}\}$ above so that for all $j < k$, $\Pr_\alpha[M_i^\alpha(x_j) \text{ queries } x_k] \leq 2/m^4$. It follows that

$$\Pr[(\exists j,k : 1 \leq j \leq k \leq 3\log m)\; M_i(x_j) \text{ queries } x_k] \leq \binom{\lceil 3\log m\rceil}{2} \cdot \frac{2}{m^4} \leq \frac{1}{m^3},$$

provided $m \geq \lceil 3\log m\rceil^2$. Hence, using Claim 6.10 with $\ell = 3\log m$, we get:

$$\Pr_\beta[\beta \sim_I M_i] \leq \frac{1}{2^{3\log m}} + \frac{1}{m^3} = \frac{2}{m^3}.$$

Since the β defined by A satisfies $\beta \sim_I M_i$, it follows by the same reasoning as in Claim 6.8 that $d_{i,m}$ profits by at least a fraction of $m^3/2$ along A. This proves Claim 6.11.

Finally, we (re-)use the pseudo-random generator D_1 as before to expand a seed s of length m into a string β_s of (at least) $r = 3\lceil \log_2 m \rceil m^i$ bits. Given any w, β_s well-defines a β and a set J_β of size at most r as constructed above, by using w to answer queries in the domain of w and β_s for everything else. We again obtain the estimate $\hat{\pi}_{i,m}(w) = \Pr_{|s|=m}[\beta_s \sim_I M_i]$ from Equation (15), with the same time complexity as before. Now we repeat Claim 6.9 in this new context:

Claim 6.12 *For all large enough good m, every estimate $\hat{\pi}_{i,m}(w)$ gives $|\hat{\pi}_{i,m}(w) - \pi_{i,m}(w)| \leq \epsilon$.*

If not, then for some fixed w the estimate fails. The final key point is that because M_i always makes its queries in lexicographic order, the queries in the domain of w that need to be covered are the same for every β_s. Hence the corresponding bits of w can be hard-wired by circuitry of size at most r. The test $[\beta_s \sim_I M_i]$ can thus still be carried out by circuits of size less than m^{i+1}, and we reach the same contradiction of the hardness value S_{D_1}.

Finally, we want to apply Lemma 5.1 to replace $d_{i,m}(w)$ by a martingale $d'_{i,m}(w)$ that yields virtually the same degree of success and is computable in time $2^{O(n)}$. Unlike the truth-table case we cannot apply Lemma 5.1 verbatim because we no longer have a single small set J that d' is active on. However, along any set A, the values $d'_{i,m}(w)$ and $d'_{i,m}(wb)$ ($b = 0$ or 1) can differ only for cases where b indexes a string in the small set J corresponding to A, and the reader may check that the argument and bounds of Lemma 5.1 go through unscathed in this case. This finishes the proof of Theorem 6.7. $\square$

7 Conclusions

The initial impetus for this work was a simple question about measure: is the pseudo-randomness of a characteristic sequence invariant under simple permutations such as that induced by *flip* in the Introduction? The question for *flip* is tantalizingly still open. However, in Section 6.2 we showed that establishing a "yes" answer for any permutation that intuitively *should* preserve the same complexity-theoretic degree of pseudo-randomness, or even for a single specific such permutation as that in the simple proof of the non-adaptive version of Theorem 6.1, would have the major consequence that EXP $\neq$ BPP.

Our "betting games" in themselves are a natural extension of Lutz's measures for deterministic time classes. They preserve Lutz's original idea of "betting" as a means of "predicting" membership in a language, without being tied to a fixed order of which instances one tries to predict, or to a fixed order of how one goes about gathering information on the language. We have shown some senses in which betting games are robust and well-behaved. We also contend that some current defects in the theory of betting games, notably the lack of a finite-unions theorem pending the status of pseudo-random generators, trade off with lacks in the resource-bounded measure theory, such as being tied to the lexicographic ordering of strings.

The main open problems in this paper are interesting in connection with recent work by Impagliazzo and Wigderson [17] on the BPP vs. EXP problem. First we remark that the main result of [17] implies that either BPP = EXP or BPP has E-measure zero [31]. Among the many measure statements in the last section that imply BPP $\neq$ EXP, the most constrained and easiest to attack seems to be item 4 in Corollary 6.5. Indeed, in the specific relevant case starting with the assumption BPP = EXP, one is given a non-adaptive E-betting game G and an E-martingale d, and to obtain the desired contradiction that proves BPP $\neq$ EXP, one need only construct an EXP-betting game G' that covers $S^\infty[G] \cup S^\infty[d]$. What we *can* obtain is a "randomized" betting game G'' that flips one coin at successive intervals of input lengths to decide whether to simulate G or d on that interval. (The intervals come from the proof of Theorem 6.4.) Any hypothesis that can de-randomize this G'' implies BPP $\neq$ EXP. We do not know whether the weak hypotheses considered in [17], some of them shown to follow from BPP $\neq$ EXP itself, are sufficient to do this.

Stepping back from trying to prove BPP $\neq$ EXP outright or trying to prove that these measure statements are equivalent to BPP $\neq$ EXP, we also have the problem of narrowing the gap between BPP $\neq$ EXP and the sufficient condition EXP $\neq$ MA used in our results. Moreover, does EXP $\neq$ MA suffice to make the $\leq_T^p$-autoreducible sets have E-measure zero? Does that suffice to simulate every betting game by a martingale of equivalent complexity? We also inquire whether there exist oracles relative to which EXP = MA but strong pseudo-random generators still exist. Our work seems to open many opportunities to tighten the connections among pseudo-random generators, the structure of classes within EXP, and resource-bounded measure.

The kind of statistical sampling used to obtain martingales in Theorems 5.4 and 5.5 was originally applied to construct martingales from "natural proofs" in [30]. The de-randomization technique from [6] based on EXP $\neq$ MA that is used here is also applied in [13, 20, 21]. "Probabilistic martingales" that can use this sampling to simulate betting games are formalized and studied in [29]. This paper also starts the task of determining how well the betting-game and random-sampling ideas work for measures on classes below E. Even straightforward attempts to carry over Lutz's definitions to classes below E run into difficulties, as described in [25] and [1, 2]. We look toward further applications of our ideas in lower-level complexity classes.

Acknowledgments The authors specially thank Klaus Ambos-Spies, Ron Book (pace), and Jack Lutz for organizing a special Schloss Dagstuhl workshop in July 1996, where preliminary versions of results and ideas in this paper were presented and extensively discussed. We thank the STACS'98 referees and the referees of this journal paper for helpful comments—one of the latter gave us many insightful notes that helped us steer away from errors and ambiguities in this final version.

References

[1] E. Allender and M. Strauss, *Measure on small complexity classes, with applications for BPP*, Tech. Report DIMACS TR 94-18, Rutgers University and DIMACS, April 1994.

[2] ——, *Measure on P: Robustness of the notion*, in Proc. 20th International Symposium on Mathematical Foundations of Computer Science, vol. 969 of Lect. Notes in Comp. Sci., Springer Verlag, 1995, pp. 129–138.

[3] K. Ambos-Spies, *P-mitotic sets*, in Logic and Machines, Lecture Notes in Computer Science 177, E. Börger, G. Hasenjäger, and D. Roding, eds., Springer-Verlag, 1984, pp. 1–23.

[4] K. Ambos-Spies and S. Lempp, July 1996. Presentation at a Schloss Dagstuhl workshop on "Algorithmic Information Theory and Randomness".

[5] L. Babai, *Trading group theory for randomness*, in Proc. 17th Annual ACM Symposium on the Theory of Computing, 1985, pp. 421–429.

[6] L. Babai, L. Fortnow, N. Nisan, and A. Wigderson, *BPP has subexponential time simulations unless EXPTIME has publishable proofs*, Computational Complexity, 3 (1993), pp. 307–318.

[7] L. Babai and S. Moran, *Arthur-Merlin games: A randomized proof system, and a hierarchy of complexity classes*, J. Comp. Sys. Sci., 36 (1988), pp. 254–276.

[8] M. Blum and S. Micali, *How to generate cryptographically secure sequences of pseudorandom bits*, SIAM J. Comput., 13 (1984), pp. 850–864.

[9] D. Boneh, *Twenty years of attacks on the RSA cryptosystem*, Notices of the American Mathematical Society, 46 (1999), pp. 203–213.

[10] H. Buhrman, L. Fortnow, and L. Torenvliet, *Using autoreducibility to separate complexity classes*, in 36th Annual Symposium on Foundations of Computer Science, Milwaukee, Wisconsin, 23–25 Oct. 1995, IEEE, pp. 520–527.

[11] H. Buhrman, L. Fortnow, D. van Melkebeek, and L. Torenvliet, *Separating complexity classes using autoreducibility*, Tech. Report FI-CXT1998-002, Fields Institute, 1998. This is the journal version of [10], and is to appear in SIAM J. Comput.

[12] H. Buhrman and L. Longpré, *Compressibility and resource bounded measure*, in 13th Annual Symposium on Theoretical Aspects of Computer Science, vol. 1046 of lncs, Grenoble, France, 22–24 Feb. 1996, Springer, pp. 13–24.

[13] H. Buhrman and D. van Melkebeek, *Hard sets are hard to find*, in Proc. 13th Annual IEEE Conference on Computational Complexity, 1998, pp. 170–181.

[14] F. Heller, *On relativized exponential and probabilistic complexity classes*, Inform. and Control, 71 (1986), pp. 231–243.

[15] R. Impagliazzo, *Very strong one-way functions and pseudo-random generators exist relative to a random oracle*, 1999. Unpublished manuscript / personal communication.

[16] R. Impagliazzo and A. Wigderson, *P = BPP if E requires exponential circuits: Derandomizing the XOR Lemma*, in Proc. 29th Annual ACM Symposium on the Theory of Computing, 1997, pp. 220–229.

[17] ———, *Randomness vs. time: De-randomization under a uniform assumption*, in Proc. 39th Annual IEEE Symposium on Foundations of Computer Science, 1998. to appear.

[18] D. Juedes and J. Lutz, *Weak completeness in E and E_2*, Theor. Comp. Sci., 143 (1995), pp. 149–158.

[19] R. Kannan, *Circuit-size lower bounds and reducibility to sparse sets*, Inform. and Control, 55 (1982), pp. 40–56.

[20] J. Köbler and W. Lindner, *On the resource bounded measure of P/poly*, in Proc. 13th Annual IEEE Conference on Computational Complexity, 1998, pp. 182–185.

[21] W. Lindner, R. Schuler, and O. Watanabe, *Resource bounded measure and learnability*, in Proc. 13th Annual IEEE Conference on Computational Complexity, 1998, pp. 261–270.

[22] D. W. Loveland, *A variant of the Kolmogorov concept of complexity*, Inform. and Control, 15 (1969), pp. 510–526.

[23] J. Lutz, *Almost everywhere high nonuniform complexity*, J. Comp. Sys. Sci., 44 (1992), pp. 220–258.

[24] ——, *The quantitative structure of exponential time*, in Complexity Theory Retrospective II, L. Hemaspaandra and A. Selman, eds., Springer Verlag, 1997, pp. 225–260.

[25] E. Mayordomo, *Contributions to the Study of Resource-Bounded Measure*, PhD thesis, Universidad Polytécnica de Catalunya, Barcelona, April 1994.

[26] N. Nisan and A. Wigderson, *Hardness versus randomness*, J. Comp. Sys. Sci., 49 (1994), pp. 149–167.

[27] N. Pippenger and M. Fischer, *Relations among complexity measures*, J. Assn. Comp. Mach., 26 (1979), pp. 361–381.

[28] A. Razborov and S. Rudich, *Natural proofs*, J. Comp. Sys. Sci., 55 (1997), pp. 24–35.

[29] K. Regan and D. Sivakumar, *Probabilistic martingales and BPTIME classes*, in Proc. 13th Annual IEEE Conference on Computational Complexity, 1998, pp. 186–200.

[30] K. Regan, D. Sivakumar, and J.-Y. Cai, *Pseudorandom generators, measure theory, and natural proofs*, in Proc. 36th Annual IEEE Symposium on Foundations of Computer Science, 1995, pp. 26–35.

[31] D. van Melkebeek, *On the measure of BPP*, Tech. Report TR-98-07, Department of Computer Science, University of Chicago, July 1998.

[32] ——, *Randomness and Completeness in Computational Complexity*, PhD thesis, University of Chicago, Department of Computer Science, June 1999. Available as UC CS Technical Report TR 99-04.

[33] M. Zimand, *On randomized cryptographic primitives*, Tech. Report TR586, University of Rochester, Computer Science Department, Nov. 1995.