Closure of Resource-Bounded Randomness Notions Under Polynomial-Time Permutations

André Nies
Department of Computer Science, The University of Auckland,
Private Bag 92019, Auckland, New Zealand
andre@cs.auckland.ac.nz

Frank Stephan
Department of Mathematics and Department of Computer Science,
National University of Singapore,
10 Lower Kent Ridge Road, Block S17, Singapore 119076, Republic of Singapore
fstephan@comp.nus.edu.sg

Abstract

An infinite bit sequence is called recursively random if no computable strategy betting along the sequence has unbounded capital. It is well-known that the property of recursive randomness is closed under computable permutations. We investigate analogous statements for randomness notions defined by betting strategies that are computable within resource bounds. Suppose that S is a polynomial time computable permutation of the set of strings over the unary alphabet (identified with $\mathbb{N}$). If the inverse of S is not polynomially bounded, it is not hard to build a polynomial time random bit sequence Z such that $Z \circ S$ is not polynomial time random. So one should only consider permutations S satisfying the extra condition that the inverse is polynomially bounded. Now the closure depends on additional assumptions in complexity theory.

Our first main result, Theorem 4, shows that if BPP contains a superpolynomial deterministic time class, such as $\mathsf{DTIME}(n^{\log n})$, then polynomial time randomness is not preserved by some permutation S such that in fact both S and its inverse are in P. Our second main result, Theorem 11, shows that polynomial space randomness is preserved by polynomial time permutations with polynomially bounded inverse, so if $\mathsf{P} = \mathsf{PSPACE}$ then polynomial time randomness is preserved.

2012 ACM Subject Classification Theory of computation → Complexity classes, Theory of computation → Pseudorandomness and derandomization

Keywords and phrases Computational Complexity, Randomness via Resource-bounded Betting Strategies, Martingales, Closure under Permutations

Digital Object Identifier 10.4230/LIPIcs.STACS.2018.51

Funding A. Nies is supported in part by the Marsden Fund of the Royal Society of New Zealand, UoA 13-184. F. Stephan is supported in part by the Singapore Ministry of Education Academic Research Fund Tier 2 grant MOE2016-T2-1-019 / R146-000-234-112. Part of this work was done while F. Stephan was on sabbatical leave at the University of Auckland. The work was completed while Nies visited the Institute for Mathematical Sciences at NUS during the 2017 programme "Aspects of Computation".

Acknowledgements The authors would like to thank Eric Allender, Klaus Ambos-Spies, Alexander Galicki, Elvira Majordomo, and Wolfgang Merkle for discussions and comments.

35th Symposium on Theoretical Aspects of Computer Science (STACS 2018).
Editors: Rolf Niedermeier and Brigitte Vallée; Article No. 51; pp. 51:1–51:10
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany
SYMPOSIUM ON THEORETICAL ASPECTS OF COMPUTER SCIENCE

1 Introduction

Formal randomness notions for infinite bit sequences can be studied via algorithmic tests. A hierarchy of such notions has been introduced. See e.g. Downey and Hirschfeldt [4] or Nies [11, Ch. 3] for definitions and basic properties, and also Li and Vitányi [8]. Criteria for good randomness notions include robustness under certain computable operations on the bit sequences. In the simplest case, such an operation is a computable permutation of the bits. For a permutation S of $\mathbb{N}$ and an infinite bit sequence Z, identified with a subset of $\mathbb{N}$, by $Z \circ S$ we denote the sequence Y such that $Y(n) = Z(S(n))$. (Note that, when viewed as a subset of $\mathbb{N}$, $Z \circ S$ equals $S^{-1}(Z)$.) We say that a class $\mathcal{C}$ of bit sequences is *closed under all members of a class $\mathcal{G}$ of permutations* if $Z \in \mathcal{C}$ implies $Z \circ S \in \mathcal{C}$ for each $S \in \mathcal{G}$.

A central notion of randomness was introduced by Martin-Löf [9]. A Martin-Löf test is a uniformly Σ^0_1 sequence $\langle G_m \rangle_{m \in \mathbb{N}}$ such that the uniform measure of G_m is at most 2^{-m}. Z fails such a test if $Z \in \bigcap_m G_m$; otherwise Z passes the test. Z is Martin-Löf random if it passes each such test. Clearly this randomness notion is closed under computable permutations S: if $Z \circ S$ fails a Martin-Löf-test $\langle G_m \rangle_{m \in \mathbb{N}}$, then Z fails the test $\langle S^{-1}(G_m) \rangle_{m \in \mathbb{N}}$. The weaker notion of Schnorr randomness [13], where one also requires that the measure of G_m is a computable real uniformly in m, is closed under computable permutations by a similar argument. Recursive randomness [13] (see e.g. [11, Ch. 7] as a recent reference) is defined via failure of all computable betting strategies (martingales), rather than by a variant of Martin-Löf's test notion. Nonetheless, by a more involved argument, implicit in [3, Section 4.1], it is closed under computable permutations. Also see Nies [11, Thm. 7.6.24] and Kjos-Hanssen, Nguyen and Rute [7].

Our main purpose is to study analogues of this result in computational complexity theory. In order to guarantee compatibility with the theory developed in Downey and Hirschfeldt [4] and Nies [11] we view sets of numbers (i.e., infinite bit sequences), rather than sets of strings over an alphabet of size at least 2, as our principal objects of study. We note that work of Lutz, Mayordomo, Ambos-Spies and others, beginning in the 1980s and surveyed in Ambos-Spies and Mayordomo [1], studied sets of strings: martingales bet on the strings in length-lexicographical order. Such languages can be identified with bit sequences via this order of strings, but the time bounds imposed on martingales are exponentially larger when they bet on strings.

To be able to apply the notions of resource bounded computability to infinite bit sequences and permutations, we will identify such bit sequences with subsets of the set $\{0\}^*$ of unary strings (also called tally languages). We view permutations as acting on $\{0\}^*$. A bit sequence is *polynomial time random* if no polynomial time computable bettings strategy succeeds on the sequence. This notion was briefly introduced by Schnorr [13], studied implicitly in the above-mentioned work of Lutz, Mayordomo, Ambos-Spies and others, and in more explicit form in Yongge Wang's 1996 thesis [15].

Our leading question is: *under which polynomial time computable permutations S is polynomial time randomness closed?* If S^{-1} is not polynomially bounded, we build a polynomial time random bit sequence Z such that $Z \circ S$ is not polynomial time random. After that, we will assume that S satisfies the extra condition that its inverse is polynomially bounded. Now the closure depends on additional assumptions in complexity theory:

- The first result, Theorem 4, shows that if BPP contains a superpolynomial deterministic time class, such as $\mathsf{DTIME}(n^{\log n})$, then polynomial time randomness is not preserved by some permutation S such that both S and its inverse are in P.
- The second result, Theorem 11, shows that PSPACE-randomness is preserved by polynomial time permutations with polynomially bounded inverse; so if $\mathsf{P} = \mathsf{PSPACE}$ then polynomial time randomness is preserved by such permutations.

Broadly speaking, the idea for the first result, Theorem 4, is as follows. Choose an $O(n^{\log n})$ time computable martingale M only betting on odd positions $1, 3, 5, \ldots$ that dominates (up to a positive factor) all polynomial time computable martingales that only bets on odd positions. Use the hypothesis in order to take a language $A \in \mathsf{BPP}$ which tells at which extension of a string of odd length M does not increase. Now let B be a highly random set (albeit B can be chosen in E). Let Z be the bit sequence that copies $B(n)$ at position $2n$, and takes the value of A at the string $Z(0) \ldots Z(2n)$ at position $2n + 1$. Then one can verify that Z is polynomial time random. If $\widehat{Z}$ is a rearrangement of the bits of Z so that a sufficiently large block of bits of B is interspersed between bits determined by A, then we can use these bits of B as random bits required in a randomised polynomial time algorithm for A. This will show that $\widehat{Z}$ is not polynomial time random.

The second result, Theorem 11, closely follows Buhrman, van Melkebeek, Regan, Sivakumar and Strauss [3, Section 4.1], which introduces and studies resource-bounded betting games. It actually shows that PSPACE-randomness is closed under certain polynomial time scanning functions, which, unlike permutations, can uncover the bits of a set in an order determined by previous bits. Each permutation in question can be seen as a scanning function of the appropriate kind. (We note that removing the resource bounds from Theorem 11 yields a proof that recursive randomness is closed under computable permutations, and in fact under computable scanning functions that scan each position.) Thm. 5.6 in Buhrman et al. [3] is a related result based on the same methods developed there; however, in that result an assumption on the existence of certain pseudorandom generators is made, while our Theorem 11 does not rest on any unproven assumptions.

We note another notion of robustness for randomness notions. One can easily adapt all the randomness notions to an alphabet other than $\{0, 1\}$. Base invariance says that the randomness notion is preserved when one replaces a sequence over one alphabet by a sequence over a different alphabet that denotes the same real number. Brattka, Miller and Nies [2] have shown this for recursive randomness, and Figueira and Nies [5] have shown it for polynomial time randomness, each time relying on the connection of randomness of a real with differentiability at the real of certain effective functions.

Using Figueira and Nies [5], Nies [12] provides a characterisation of polynomial time randomness for real numbers in terms of differentiability of all polynomial time computable nondecreasing functions on the reals.

2 Preliminaries

For a bound h, as usual $\mathsf{DTIME}(h)$ denotes the languages A computable in time $O(h)$. Informally we often say that A is computable in time h. As in Ambos-Spies and Mayordomo [1], we require that martingales have rational values.

▶ **Definition 1.** A martingale M is a function from $\{0, 1\}^*$ to $\{q \in \mathbb{Q} : q > 0\}$ satisfying $M(x) = (M(x0) + M(x1))/2$ for all $x \in \{0, 1\}^*$. A martingale succeeds on a set Z if $\limsup_n M(Z \upharpoonright n) = \infty$. One says that a martingale *does not bet at a position* n if $M(x0) = M(x1)$ for each $x \in \{0, 1\}^n$.

One says that Z is recursively random if no computable martingale succeeds on Z.

Each polynomial in this paper will be non-constant and have natural number coefficients. For a polynomial time version of recursive randomness, we have to be careful how to define polynomial time computability for a martingale: as in [3], a positive rational number q is presented by a pair $\langle k, n\rangle$ consisting of a denominator and a numerator (both written in

binary) such that $q = k/n$ in lowest terms. A martingale M is polynomial time computable if on input x one can determine $M(x)$ in this format in polynomial time. Z is polynomial time random if no such martingale succeeds on Z. In a similar way one defines exponential time randomness.

A martingale M is polynomial space computable if $M(x)$ can be computed in polynomial space (including the space needed to write the output). Z is polynomial space random if no such martingale succeeds on Z.

We first show that polynomial time randomness fails to be closed under polynomial time computable permutations S that are "dishonest" in the sense that $S(n)$ can be much less than n.

▶ **Theorem 2.** *Let S be a polynomial time permutation of $\{0\}^*$ such that for each polynomial p, there are infinitely many n with $p(S(n)) \leq n$. There is a polynomial time random Z computable in time $2^{O(n)}$ such that $Z \circ S$ is not polynomial time random.*

Clearly a permutation S as in Theorem 2 exists: Let $(p_k)_{k\in\mathbb{N}}$ list the non-constant polynomials with natural coefficients in such a way that for $u \leq n$, $O(n^2)$ steps suffice to verify whether $p_k(u) \leq n$. On input n of the form $\langle k, i\rangle$, see whether $p_k(\langle k, 0\rangle) \leq n$. If not let $S(n) = \langle k, i+1\rangle$. If so and n is least such, let $S(n) = \langle k, 0\rangle$. Otherwise $S(n) = n$.

Proof of Theorem 2. Nies [11, Section 7.4] provided a construction template for recursively random sets, going back to Schnorr's work. We adapt some parts of this template to the resource bounded setting.

Let $\langle B_k\rangle$ be an effective listing of the polynomial time martingales with positive rational values. We may assume that B_k is computable in time $p_k(n) = k(n^k + 1)$.

For each n, let $B_{k,n}$ be the martingale with initial capital 1 that does not bet until its input reaches length n, and then uses the same betting factors as B_k. Thus,

$$B_{k,n}(x) = \frac{B_k(x)}{B_k(x \upharpoonright n)}$$

for any string x of length at least n. Let $\widetilde{p}_{k,n}$ be a polynomial so that $B_{k,n}(x)$ for $|x| \geq n$ can be computed in time $\widetilde{p}_{k,n}(|x|)$.

We inductively define a sequence of numbers. Let $n_0 = 0$, and let n_{k+1} be the least $n > n_k$ such that $q_k(S(n)+1) \leq n$, where q_k is a polynomial time bound for the martingale $\sum_{r\leq k} 2^{-r} B_{r,n_r}$ and $q_k(n) \geq n+2$. Let $L = \sum_r 2^{-r} B_{r,n_r}$. Note that L is a rational-valued martingale, because on inputs of length at most n_k, all the B_{r,n_r} for $r > k$ together contribute 2^{-k}.

Let now Z be the left-most non-ascending path of L: $Z(m) = 0$ if $L(Z \upharpoonright m\hat{}0) \leq L(Z \upharpoonright m)$, and $Z(m) = 1$ otherwise. Since L does not succeed on Z and L multiplicatively dominates each B_k, the set Z is polynomial time random.

Note that since $S \in \mathsf{P}$, from n we can in polynomial time recursively recover the sequence $n_0, q_0, n_1, q_1, \ldots$ and thereby compute the maximal k such that $n_k < n$. In particular we can decide whether n is of the form n_{k+1} for some k. By definition, for $n = n_{k+1}$ we have $q_k(S(n)+1) \leq n$ and hence $S(n)+1 < n_{k+1}$. Since q_k as a time bound is sufficient to determine $L(y)$ for strings y of length $S(n)+1$, the bit $Z \circ S(n)$ can be computed in time polynomial in n. Hence $Z \circ S$ is not polynomially random.

We can ensure such a set Z is computable in time $2^{O(n)}$ by choosing the listing $\langle B_k\rangle$ appropriately. ◀

▶ Remark. We note that methods involving the $\langle B_{k,n}\rangle$ similar to the above can be used to show that each class $\mathsf{DTIME}(h)$ with superpolynomial time constructible h contains a polynomial time random (tally) set. We have to initiate a copy $\langle B_{k,n}\rangle$ of B_k finitely many times until a length n is reached such that for $m \geq n$, $h(m)$ time is sufficient to simulate its behaviour on strings of length m.

3 If BPP Contains a Superpolynomial Time Class Then Closure Fails

▶ **Definition 3.** A permutation S of $\{0\}^*$ is called *fully polynomial time computable* if both S and S^{-1} are polynomial time computable.

A complexity theoretic assumption considerably weaker than $\mathsf{BPP} = \mathsf{EXP}$ suffices for non-closure.

▶ **Theorem 4.** *Suppose that* $\mathsf{DTIME}(h) \subseteq \mathsf{BPP}$ *for some time constructible function* h *that dominates all the polynomials. Then there are a polynomial time random set* $Z \in \mathsf{DTIME}(2^{3n})$ *and a fully polynomial time computable permutation* S *such that* $Z \circ S$ *is not polynomial time random.*

Proof. We may assume that $h(n) \leq n^{\log n}$. It is well-known that whenever a martingale in a certain complexity class succeeds on a set Z then there is also a successful martingale in the same class betting only on even positions, or there is a successful martingale betting only on the odd positions.

The construction has two steps. Firstly, by standard methods discussed at the end of Section 2, one can build a martingale M in $\mathsf{DTIME}(h)$ which bets only on odd positions, and dominates up to a multiplicative constant all polynomial time martingales betting on odd positions. Let

$$A = \{x \in \{0,1\}^* : x \text{ has odd length and } M(x1) < M(x0)\}.$$

The set A is in $\mathsf{DTIME}(h)$ and hence by assumption in BPP.

Secondly, let $B \subseteq \{0\}^*$ be a language on which no martingale in $\mathsf{DTIME}(2^{4\cdot n})$ succeeds. Again by standard methods one can ensure that B is in $\mathsf{DTIME}(2^{5\cdot n})$. Define a set $Z \subseteq \mathbb{N}$ as follows:

$$Z(2n) = B(n);\ Z(2n+1) = A(Z \upharpoonright 2n+1).$$

We may visualise Z as follows:

B	A	B	A	B	A	B	A	B	A	B	A	B	...

$B(0)\ A(Z \upharpoonright 1)\ B(1)\ A(Z \upharpoonright 3)\ B(2)\ A(Z \upharpoonright 5) \ldots$

Clearly $Z \in \mathsf{DTIME}(2^{3n})$. It is claimed that Z is polynomial time random. As the martingale M only bets on odd positions, Z is defined such that M never gains capital on Z. As M is universal among the martingales computable in polynomial time with this property, no martingale betting on the odd positions succeeds on Z.

Suppose now that L is a polynomial time martingale which bets on the even positions and note that one can compute in time $O(h(n))$ from $B(0), B(1), \ldots, B(n)$ inductively the values $Z(0), Z(1), \ldots, Z(2n+1)$, as for every x of length $2n+1$ the value $A(x)$ can be computed in time $h(n)$. Thus if L succeeds on Z then there is a new martingale N succeeding on B which satisfies that

$$N(B \upharpoonright n+1) = L(Z \upharpoonright 2n)$$

and which uses that $Z(2n) = B(n)$ while the bits of Z at odd positions on which L does not bet can be computed as indicated above from the other bits. To compute $N(x)$ for x of length $2n$ takes $q(n)+\sum_{i<n} h(2i+1)$ steps for some polynomial q. So $N \in \mathsf{DTIME}(n^{O(\log n)})$, which contradicts the assumption that no such martingale computable in time $O(2^{4n})$ succeeds on B. This verifies the claim.

Since $A \in \mathsf{BPP}$, there is a polynomial p such that an appropriate randomised algorithm $\mathcal{R}$ on input $x \in \{0,1\}^{2n+1}$ computes $A(x)$ in time $p(n)$, with error probability 2^{-4n-2}, using $p(n)$ random bits. Now consider the sequence $\widehat{Z}$ consisting for $n = 0, 1, \ldots$ of $p(n)$ bits taken from B followed by the bit $Z(2n+1)$. Again we visualise $\widehat{Z}$:

B	A	B	B	B	A	B	B	B	B	B	B	A	...

$p(0)$ $p(1)$ $p(2)$

Formally one can define $\widehat{Z}$ from Z as follows:

for $m < p(n)$,

$$\begin{aligned}
\widehat{Z}((\sum_{k<n} p(k)) + n + m) &= B((\sum_{k<n} p(k)) + m) = Z(2(\sum_{k<n} p(k) + m));\\
\widehat{Z}((\sum_{k\leq n} p(k)) + n) &= Z(2n+1) = A(Z \restriction 2n+1).
\end{aligned}$$

This mapping is given by a permutation S so that $\widehat{Z}(r) = Z(S(r))$ for all positions r. So if $r = (\sum_{k<n} p(k)) + n + m$ then $S(r) = 2(\sum_{k<n} p(k)) + 2m$ and if $r = (\sum_{k\leq n} p(k)) + n$ then $S(r) = 2n+1$, for all m, n with $m < p(n)$. The permutation S and its inverse satisfy that the mappings $0^k \mapsto 0^{S(k)}$ and $0^k \mapsto 0^{S^{-1}(k)}$ on the unary strings $\{0\}^*$ are polynomial time computable, thus the S is of the form as required; to see this note that for a polynomial p also the mapping $n \mapsto \sum_{k<n} p(k)$ is a polynomial; similarly for a function bounded by a polynomial.

Now it will be shown that $\widehat{Z}$ is not polynomial time random. Note that there are 2^{2n+1} strings of length $2n+1$. Given a string of $p(n)$ random bits, the probability that when using these bits the randomised algorithm $\mathcal{R}$ computes $A(x)$ correctly for all $x \in \{0,1\}^{2n+1}$ is at least $1 - 2^{2n+1} \cdot 2^{-4n-2} = 1 - 2^{-2n-1}$. We want to show that B provides random bits that allow $\mathcal{R}$ to correctly compute A for almost all inputs. Otherwise, we can build a martingale M computable in time $2^{10\cdot n}$ which succeeds on B: The martingale M splits its capital into bins of value 2^{-n-1} and for each block of $p(n)$ bits starting at $\sum_{k<n} p(k)$, it takes the value 2^{-n-1} from the corresponding bin and bets it on the strings y consisting of $p(n)$ bits that do not compute all values of $A(x)$ with $x \in \{0,1\}^{2n+1}$ correctly using $\mathcal{R}$. This condition can be checked for these bits in the time bound given as it involves running $\mathcal{R}$ with y as the random bits on all strings x of length $2n+1$ and comparing the result with $A(x)$ for all $2^{p(n)}$ choices of random bits y. After these simulations, M distributes the capital from the bin evenly on those strings of random bits which cause $\mathcal{R}$ to make an error. After having processed the bits from the block of $p(n)$ bits, the capital in this bin remains unchanged by future bets. The set of random strings y on which the computation of some of the $A(x)$ in $x \in \{0,1\}^{2n+1}$ is false has at most the probability $2^{-4n-2} \cdot 2^{2n+1} = 2^{-2n-1}$. Therefore the capital from the bin multiplies at least by 2^{n+1} during the block and reaches the value 1.

For the time bound on M, whenever the input has length between $\sum_{k<n} p(k)$ and $\sum_{k\leq n} p(k)$, the martingale computes 2^{n+1} values $A(x)$ for $x \in \{0,1\}^{2n+1}$ with respect to $p(n)$ random bits taking $2^{p(n)}$ possible choices. However, for all polynomials and almost all n, $p(n) \leq \sum_{k<n} p(k)$, as the degree of the sum-polynomial of p is by one above the degree

of p and the polynomial p is positive. Thus, for such n, when $n' = \sum_{k<n} p(k)$ is a lower bound on the length of the input to the martingale M then $p(n) \leq n'$ and $2n+1 \leq n'$ and thus the whole computations can be handled in time $O(2^{3n'})$.

If there are infinitely many blocks in B where the random bits of this block do not compute all $A(x)$ with x of the corresponding length correctly, then this martingale succeeds, contrary to the assumption on B. So, for almost all n, the block of $p(n)$ random bits in $\widehat{Z}$ before $A(Z \upharpoonright 2n)$ permits to compute this value correctly.

Now this property will be used to construct a polynomial time martingale H which succeeds on $\widehat{Z}$. Let $\tilde{A}(n)$ denote $A(Z \upharpoonright 2n+1)$. Given $p(n)$ random bits from B preceding $\tilde{A}(n)$ in $\widehat{Z}$, the martingale H archives these bits without betting on them. It then bets half of its capital on the value for $\tilde{A}(n)$ computed from these random bits; note that due to $\tilde{A}(0), \tilde{A}(1), \ldots, \tilde{A}(n-1)$ and $B(0), B(1), \ldots, B(n)$ being coded in $\widehat{Z}$ in positions before that of $\tilde{A}(n)$, when the bet for $\tilde{A}(n) = Z(2n+1)$ has to be made, one can retrieve besides the random bits also $Z(0)Z(1)\ldots Z(2n)$ from the history. So one can use the random bits to compute the value almost always correctly. Thus the martingale H will only finitely often place a wrong bet and lose some of its capital, but for almost all $\tilde{A}(n)$ predict the value correctly and multiply its capital by $3/2$. Thus the martingale succeeds. As all the operations above are polynomial time computable, the set $\widehat{Z}$ is not polynomial time random. ◀

The proof of Theorem 4 can be adjusted to obtain a corollary.

▶ **Corollary 5.** *Let $A, B \subseteq \{0\}^*$. Suppose that A is in* BPP *and B is* EXP*-random relative to A. Then A is polynomial time computable relative to B, and in particular not polynomial time random relative to B.*

Proof. For ease of notation, we often write $A(n)$ in place of $A(0^n)$ and so on; however, both A and B are viewed as subsets of $\{0\}^*$.

There is a polynomial time algorithm and a polynomial p such that the algorithm uses $p(n)$ random bits to compute $A(n)$ with error probability 2^{-n}. As in the theorem above, one can now query B for getting the random bits and the places where the queries are asked are different for n, m whenever $n \neq m$. So there is a polynomial q with $q(n) + p(n) = q(n+1)$ for all n and where the algorithm asks the bits of B at $q(n), q(n)+1, \ldots, q(n)+p(n)-1$ to compute $A(n)$.

If now there is an error, then an exponential time martingale relative to A can make sufficient profit, as only a slim minority of the possiblities of the bits of B from $q(n)$ to $q(n)+p(n)-1$ are realised. This contradicts the assumption that B is random relative to A. Hence A can be computed relative to B by this algorithm with only finitely many errors; these can then be corrected by a finite table holding the correct values for the positions where the algorithm makes an error. ◀

▶ Remark. In the proof of Theorem 4, $Z = \tilde{A} \oplus B$ is polynomial time random; however, $\tilde{A}$ is not polynomial time random relative to B, as the rearrangement with S shows. Note that van Lambalgen's Theorem [14] says that in a recursion-theoretic setting, $\tilde{A} \oplus B$ is random iff (a) B is random and (b) $\tilde{A}$ is random relative to B. Thus, under the assumption that BPP = EXP, one of the directions of the van Lambalgen Theorem does not hold for polynomial time randomness.

The corollary also shows that one can choose, under the assumption that BPP contains a superpolynomial time class, sets $A, B \subseteq \{0\}^*$ such that A is polynomial time random, B is polynomial time random relative to A and A is polynomial time computable relative to B. Hence this assumption implies that A is a basis for polynomial time randomness

even though A is polynomial time random itself. This contrasts with the setting of Martin-Löf randomness randomness in recursion theory: a basis for Martin-Löf randomness has to be trivial and therefore cannot be random [6, 10]. On the other hand, the bases for recursive randomness include every set below the halting problem that is not diagonally noncomputable (DNC), but no set of PA degree [6]. Every high set computes a recursively random set, and an incomplete high r.e. set is not DNC. So a recursively random set can be a basis for recursive randomness.

4 If P = PSPACE Then Closure Holds

We say that $Z \subseteq \mathbb{N}$ is *polynomial space random* if no martingale computable in polynomial space succeeds on Z. In this section we show that polynomial space randomness is closed under fully polynomial time computable permutations in the sense of Definition 3. If $\mathsf{P} = \mathsf{PSPACE}$ this closure property applies to polynomial time randomness as well.

In fact we show a stronger closure property where the permutations are generalised to certain non-monotonic scanning rules, which adaptively specify an order in which bits are read. We modify the argument given by Buhrman, van Melkebeek, Regan, Sivakumar and Strauss [3, Section 4.1], which was not concerned with polynomial space randomness, but rather was geared to the context of Lutz's theory of resource bounded measure. As already mentioned, in that theory, the positions a martingale bets on are strings in some non-unary alphabet. Such strings can be suitably encoded by natural numbers; however, the resource bounds change when one converts such a martingale into one in the sense of our Definition 1. The next two definitions formalise the idea of betting on bit positions in an order chosen adaptively by the betting strategy. We take some key technical concepts from [3, Section 4.1], somewhat changing the terminology in order to make it compatible with the one of Nies [11, Section 7.5] where non-monotonic randomness notions are studied.

▶ **Definition 6.** A *scanning function* is a function $V\colon \{0,1\}^* \to \{0\}^*$ such that $V(\alpha) \neq V(\alpha \upharpoonright i)$ for each $\alpha \in \{0,1\}^*$ and each $i < |\alpha|$. In the context of V, we will call a string α a *run of* V, thinking of α as a sequence of answers to oracle queries. We will call $V(\alpha \upharpoonright i)$ the *i-th query in the run of V on α*.

As before, subsets of $\mathbb{N}$ will be identified with languages over the unary alphabet $\{0\}$. For $Z \subseteq \mathbb{N}$ let $Z \circ V \subseteq \mathbb{N}$ be the set Y such that $Y(i) = Z(V(Y \upharpoonright i))$ for each i.

▶ **Definition 7.** A *non-monotonic betting strategy* G is a pair (V, B) such that V is a scanning function and B is a martingale. G succeeds on $Z \subseteq \{0\}^*$ if $\lim_n B(Z \circ V \upharpoonright n) = \infty$.

One says that a non-monotonic betting strategy G is computable in polynomial space if both V and B are computable in polynomial space. One says that $Z \subseteq \mathbb{N}$ is *non-monotonically polynomial space random* if no such betting strategy succeeds on Z.

Another concept we need is that of consistency between a run α of V and a string w.

▶ **Definition 8.** For bit strings α, w, we write $\alpha \sim_V w$ if for each $j < |\alpha|$, if the j-th query x in the run of V on α is less than $|w|$, then $w(x) = \alpha(j)$.

▶ **Definition 9.** For a function $g\colon \mathbb{N} \to \mathbb{N}$, one says that V is *g-filling* if for each n and each run α of length $g(n)$, we have $\forall r < n\, \exists i\, V(\alpha \upharpoonright i) = r$.

▶ **Lemma 10.** *Suppose V is g-filling. Let $|\alpha| \geq i := g(|w|)$. Then $\alpha \sim_V w$ iff $\alpha \upharpoonright i \sim_V w$.*

To see this, note that by the definition of being g-filling, any query q with $q < |w|$ has to be asked before stage $g(|w|)$.

▶ **Theorem 11.** *Let V be a scanning function in* PSPACE *that is g-filling for a polynomial bound g. If Z is polynomial space random, then so is $Z \circ V$.*

Proof. Suppose $Z \circ V$ is not polynomial space random. Let $G = (V, B)$ be a betting strategy in PSPACE that succeeds on Z; thus, B succeeds on $Z \circ V$.

We define a martingale D in PSPACE that succeeds on Z. We may assume that $g(n) \geq n$. For $t \geq g(|w|)$ let

$$D(w) = 2^{|w|-t} \sum_{|\alpha|=t \ \wedge \ \alpha \sim_V w} B(\alpha).$$

By the claim above and since B is a martingale, this definition is independent of t. Note that among the runs α of length t, a fraction of $2^{-|w|}$ satisfy that $\alpha \sim_V w$; so $D(w)$ is simply the average value of $B(\alpha)$ over all such α.

If we let $t = g(|w|)$, by the hypotheses that G is in PSPACE and that g is a polynomial, D is in PSPACE.

The rest of the argument somewhat simplifies the one of [3] in the present context.

▶ **Lemma 12.** *D is a martingale.*

Let w be a string of length n. If $|\alpha| = g(n+1)$ and $\alpha \sim_V w$, then either $\alpha \sim_V w0$ or $\alpha \sim_V w1$. Letting $u = g(n+1)$, for each $r = 0, 1$ we have

$$D(wr) = 2^{|w|+1-u} \textstyle\sum_{|\alpha|=u \ \wedge \ \alpha \sim_V wr} B(\alpha).$$

Hence, since the definition of $D(w)$ does not depend on the choice of $t \geq g(|w|)$,

$$D(w0) + D(w1) = 2^{|w|+1-u} \textstyle\sum_{|\alpha|=u \ \wedge \ \alpha \sim_V w} B(\alpha) = 2D(w).$$

▶ **Lemma 13.** *D succeeds on Z.*

We may assume that $B(x) > 0$ for each x. The Savings Lemma (see e.g. Nies [11, 7.1.14]) states that each computable martingale M can be turned into a computable martingale $\widehat{M}$ that succeeds on the same sets, and has the extra property that $\widehat{M}(\beta) \geq \widehat{M}(\alpha) - 2$ for each strings $\beta \supseteq \alpha$ (namely, $\widehat{M}$ never loses more than 2). It is easy to see from the proof that if M is computable in polynomial space, then so is $\widehat{M}$. So we may assume that B has this property.

This implies that for each $c \in \mathbb{N}$ there is a prefix α of $Z \circ V$ such that

$$B(\beta) \geq c \text{ for each string } \beta \succeq \alpha.$$

By definition of $Z \circ V$ we have $\alpha(i) = Z(V(\alpha \upharpoonright i))$ for each $i < |\alpha|$. Let $r = 1 + \max_{i<|\alpha|} V(\alpha \upharpoonright i)$ be 1+ the maximum query asked in the run of V on α, and let $w = Z \upharpoonright r$. So $g(r) \geq |\alpha|$.

If $\beta \sim_V w$ is a string such that $|\beta| = g(r)$, then $\beta \succeq \alpha$, for $\alpha(r) \neq \beta(r)$ for some $r < |\alpha|$ would imply that $\beta \not\sim_V w$ as w answers all such queries correctly. So $B(\beta) \geq c$. Hence $D(w) \geq c$ because $D(w)$ is the average over values $B(\beta)$ for all such β. ◀

▶ **Corollary 14.** *Let S be a polynomial time computable permutation of $\{0\}^*$ such that S^{-1} is polynomially bounded. If Z is polynomial space random, then so is $Z \circ S$.*

Proof. The permutation S can be viewed as a scanning function V_S that only looks at the length of the input: $V_S(\alpha) = S(|\alpha|)$. By hypothesis on S, the scanning function V_S is polynomially filling. So $Z \circ S = Z \circ V_S$ is polynomial space random by the theorem. ◀

The foregoing corollary can be restated in terms of randomness on languages in the sense of [1]: Let S be a exponential time computable permutation of $\{0,1\}^*$ such that $|S^{-1}(x)| = O(|x|)$ for each string x. If a language Z is exponential space random, then so is $Z \circ S$.

We end with a question. Recall that PP denotes probabilistic polynomial time, a subclass of PSPACE. If P = PP, is polynomial time randomness closed under permutations S of $\{0\}^*$ such that S, S^{-1} are polynomial time computable?

References

1. K. Ambos-Spies and E. Mayordomo. Resource-bounded measure and randomness. *Lecture Notes in Pure and Applied Mathematics*, pages 1–48, 1997.
2. V. Brattka, J. Miller, and A. Nies. Randomness and differentiability. *Transactions of the AMS*, 368:581–605, 2016. arXiv version at arxiv.org/abs/1104.4465.
3. H. Buhrman, D. Van Melkebeek, K. Regan, D. Sivakumar, and M. Strauss. A generalization of resource-bounded measure, with application to the BPP vs. EXP problem. *SIAM Journal on Computing*, 30(2):576–601, 2000.
4. R. Downey and D. Hirschfeldt. *Algorithmic randomness and complexity.* Springer-Verlag, Berlin, 2010. 855 pages.
5. S. Figueira and A. Nies. Feasible analysis, randomness, and base invariance. *Theory of Computing Systems*, 56(3):439–464, 2015.
6. D. Hirschfeldt, A. Nies, and F. Stephan. Using random sets as oracles. *J. Lond. Math. Soc. (2)*, 75(3):610–622, 2007.
7. B. Kjos-Hanssen, P. Nguyen, and J. Rute. Algorithmic randomness for doob's martingale convergence theorem in continuous time. *arXiv preprint arXiv:1411.0186*, 2014.
8. M. Li and P. Vitányi. *An introduction to Kolmogorov complexity and its applications.* Graduate Texts in Computer Science. Springer-Verlag, New York, second edition, 1997.
9. P. Martin-Löf. The definition of random sequences. *Inform. and Control*, 9:602–619, 1966.
10. A. Nies. Lowness properties and randomness. *Advances in Mathematics*, 197:274–305, 2005.
11. A. Nies. *Computability and Randomness*, volume 51 of *Oxford Logic Guides.* Oxford University Press, Oxford, 2009. 444 pages. Paperback version 2011. `doi:10.1093/acprof:oso/9780199230761.001.0001`.
12. André Nies. Differentiability of polynomial time computable functions. In Ernst W. Mayr and Natacha Portier, editors, *31st International Symposium on Theoretical Aspects of Computer Science (STACS 2014), STACS 2014, March 5-8, 2014, Lyon, France*, volume 25 of *LIPIcs*, pages 602–613. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2014. `doi:10.4230/LIPIcs.STACS.2014.602`.
13. C.P. Schnorr. *Zufälligkeit und Wahrscheinlichkeit. Eine algorithmische Begründung der Wahrscheinlichkeitstheorie.* Springer-Verlag, Berlin, 1971. Lecture Notes in Mathematics, Vol. 218.
14. Michiel van Lambalgen. The axiomatization of randomness. *J. Symbolic Logic*, 55(3):1143–1167, 1990.
15. Y. Wang. *Randomness and Complexity.* PhD dissertation, University of Heidelberg, 1996.