ROBUSTNESS OF RESOURCE BOUNDED RANDOMNESS NOTIONS UNDER POLYNOMIAL TIME PERMUTATIONS

ANDRÉ NIES AND FRANK STEPHAN

Abstract. It is well known that recursive randomness is closed under computable permutations. We investigate analogous statements for resource bounded randomness notions. Suppose S is a polynomial time computable permutation of the set of strings over the unary alphabet (identified with $\mathbb{N}$). If its inverse is not polynomially bounded, it is easy 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.

The first result, Theorem 3.2 shows that if $\mathsf{BPP} = \mathsf{EXP}$ then polynomial time randomness is not preserved by some permutation S such that in fact both S and its inverse a in P. As our second result, Theorem 4.5, we show 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.

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. Nies [8, Ch. 3] for definitions and basic properties. 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 new sequence Y such that $Y(n) = Z(S(n))$. (Note that the set $S(Z)$ equals $Z \circ S^{-1}$ in this notation.) We say that a class $\mathcal{C}$ of subsets of $\mathbb{N}$ 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 formal notion of randomness is Martin-Löf's [6]. Clearly it 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 is closed under computable permutations by a similar argument. Recursive randomness (e.g. [8, Ch. 7]) 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, it is closed under computable permutations [5].

Our main purpose is to study analogies in computational complexity theory of this result. To be able to apply resource bounded notions, we identify $\mathbb{N}$ with the set 0^* of unary strings, view permutations as acting on this 0^*, and view infinite

---

Date: August 11, 2017.

2010 *Mathematics Subject Classification.* 03D32,68Q30.

Key words and phrases. Computational Complexity, Algorithmic Randomness, Martingales.

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 to the University of Auckland.

bit sequences as languages over the unary alphabet. A bit sequence is polynomial time random if no polynomial time computable bettings strategy succeeds. Which polynomial time permutations S is polynomial time randomness is closed under?

Firstly, 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 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 3.2, shows that if $\mathsf{BPP} = \mathsf{EXP}$ then polynomial time randomness fails to be preserved by some permutation S such that in fact both S and its inverse a in P. The second result, Theorem 4.5 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. Theorem 4.5 closely follows Buhrman, van Melkebeek, Regan, Sivakumar and Strauss [2, Section 4.1]. 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 questions can be seen as a scanning function of the appropriate kind. Stripping Theorem 4.5 of its resource bounds yields a proof that recursive randomness is closed under computable permutations.

We remark that another notion of robustness for randomness notions is base invariance. One can easily adapt all the notions to an alphabet other than $\{0,1\}$. Base invariance says that randomness is preserved by changing between sequences in different alphabets that denote the same real number. Brattka, Miller and Nies [1] have shown this for recursive randomness and Figueira and Nies [3] 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.

2. Preliminaries

Definition 2.1. 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. For a polynomial time version of this notion, we have to be careful how to define polynomial time computability for a martingale: a positive rational number is presented by pair of denominator and numerator 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 and polynomial space randomness.

Each polynomial in this paper will be non-constant and have natural number coefficients. We first show that polynomial time randomness is not closed under polynomial time computable permutations S that are "dishonest" in the sense that $S(n)$ can be much less than n.

Proposition 2.2. *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 exponential time such that $Z \circ S$ is not polynomial time random.*

Clearly a permutation S as in Proposition 2.2 exists: Let (p_k) 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 Proposition 2.2. Nies [8, Section 7.4] provides a construction template for recursively random sets, originally 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 $B_{k,n}$ 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+1}, all the B_{r,n_r} for $r \geq k$ 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 Z is computable in exponential time by choosing the listing $\langle B_k\rangle$ appropriately. □

3. If $\mathsf{BPP} = \mathsf{EXP}$ Then Closure Fails

Definition 3.1. A permutation S of $\{0\}^*$ is called fully polynomial time computable if both S and S^{-1} are polynomial time computable.

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

Theorem 3.2. *Suppose that* $\mathsf{DTIME}(h) \subseteq \mathsf{BPP}$ *for some time constructible function h that dominates all the polynomials. Then there is polynomial time random set* $Z \in \mathsf{DTIME}(2^{O(n)})$ *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 betting only on even positions or there is a successful martingale betting only on the odd positions, where this successful martingale is from the same complexity class.

The construction has two steps. First let M be a martingale in $\mathsf{DTIME}(h)$ which bets only on odd positions, and is universal for 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 computable in time h and hence by assumption in BPP. Furthermore, let $B \subseteq \{0\}^*$ be a tally language which is random relative to every martingale in $\mathsf{DTIME}(2^{10 \cdot n})$. Thus one can then compute $B(0^n)$ in time $2^{11 \cdot n}$. Define a set $Z \subseteq \mathbb{N}$ as follows:

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

Now it is shown 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 $N \in \mathsf{DTIME}(nh(2n+1))$ 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. Now $N \in \mathsf{DTIME}(2^{21 \cdot n})$ and no polynomial time martingale succeeds on Z. This shows the claim.

By assumption $A \in \mathsf{BPP}$. Therefore there is a polynomial p such that for every $x \in \{0,1\}^{2n+1}$ a randomised algorithm $\mathcal{R}$ computes $A(x)$ with error probability 2^{-4n-2} in time $p(n)$, using $p(n)$ random bits. Now consider the sequence Q consisting for $n = 0, 1, \ldots$ of $p(n)$ bits taken from B followed by the bit $Z(2n+1)$. More precisely, one can define Q from Z as follows:

$$\begin{aligned}
&\text{for } m < p(n),\\
&Q((\sum_{k<n} p(k)) + n + m) \;=\; B((\sum_{k<n} p(k)) + m) \;=\; Z(2(\sum_{k<n} p(k) + m));\\
&Q((\sum_{k\le n} p(k)) + n) \;=\; Z(2n+1) = A(Z \upharpoonright 2n+1).
\end{aligned}$$

This mapping is given by a permutation S so that $Q(r) = Z(S(r))$ 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 $r = (\sum_{k\le n} p(k)) + n$ 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 Q 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 it bets 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, the capital in the 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 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 $O(2^{3n'})$ and so the martingale cannot be successful on B by the choice of B as being random for such martingales.

Thus, 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 and thus B is not $2^{10\cdot n}$ random. So, for almost all n, the block of $p(n)$ random bits in Q before $A(Z(0)Z(1)\ldots Z(2n))$ permits to compute this value correctly.

Now this property will be used to construct a polynomial time martingale H which succeeds on Q. Let $\tilde{A}(n)$ denote $A(Z \restriction 2n+1)$. Given $p(n)$ random bits from B preceding $\tilde{A}(n)$ in Q, 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 Q 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 Q is not polynomial time random. □

Corollary 3.3. *Assume that A is in* BPP *and B is* EXP*-random relative to A. Then A is polynomial time computable relative to B and thus not polynomial time random relative to B.*

Proof. The proof of Theorem 3.2 can be adjusted to get the corollary. So assume that A is in BPP and B is EXP-random relative to A. There is a polynomial time algorithm and a polynomial p such that the algorithm uses $p(n)$ random bits to compute $A(n)$ (more precisely, $A(0^n)$) 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 $0^{q(n)}, 0^{q(n)+1}, \ldots, 0^{q(n)+p(n)-1}$ to compute $A(0^n)$. 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 3.4. In the proof of Theorem 3.2, $Z = \tilde{A} \oplus B$ is polynomial time random; however, $\tilde{A}$ is not polynomial time random relative to B, as the reorganisation with

S shows. Note that van Lambalgen's Theorem [9] says that in a recursion-theoretic setting that $\tilde{A} \oplus B$ is random iff (a) B is random and (b) $\tilde{A}$ is random relative to B. Thus, under the assumption that $\mathsf{BPP} = \mathsf{EXP}$, one of the directions of the van Lambalgen Theorem does not hold for polynomial time randomness. Furthermore, the corollary shows that one can choose, under the assumption that $\mathsf{BPP} = \mathsf{EXP}$, sets A, B such that A is polynomial time random, B is polynomial time random relative to A and A is computable relative to B. This would imply that A is a basis of polynomial time randomness although A is polynomial time random itself; this stands in contrast to the world of randomness in recursion theory where a basis of randomness has to be trivial and cannot be random [4, 7].

4. If $\mathsf{P} = \mathsf{PSPACE}$ Then Closure Holds

We say that $Z \subseteq \mathbb{N}$ is PSPACE *random* if no PSPACE computable martingale succeeds on Z. In this section we show that PSPACE randomness is closed under fully polynomial time computable permuations in the sense of Definition 3.1. 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 (adaptively specifying an order in which bits are read). We modify the argument given by Buhrman, van Melkebeek, Regan, Sivakumar and Strauss [2, Section 4.1], which was not concerned with PSPACE-randomness, but rather geared for the context of Lutz's theory of resource bounded measure. In that theory martingales bet on strings in some non-unary alphabet. These can of course be suitably encoded by natural numbers; however, the resource bounds will change when one does so.

Definition 4.1. 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 $i < |\alpha|$. Given V, we will call a string α a *run of* V, thinking of α as a sequence of answers to oracle queries. In this context 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 $V(Z) \subseteq \mathbb{N}$ be the set Y such that $Y(i) = Z(V(Y \upharpoonright i))$ for each i. In the following we review some key technical concepts of Buhrman, van Melkebeek, Regan, Sivakumar and Strauss [2, Section 4.1], somewhat changing the terminology in order to make it compatible with the one of Nies [8, Section 7.5] where non-monotonic randomness notions are studied.

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 $g(n)$,

$$\forall r < n\, \exists i\, V(\alpha \upharpoonright i) = r.$$

Definition 4.2. 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 $\lim_n B(V(Z) \upharpoonright n) = \infty$.

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* PSPACE *random* if no such betting strategy succeeds on Z.

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

Definition 4.3. 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 $w(x) = \alpha(j)$. By $m(\alpha, w)$ we denote the number of such queries that are answered by w.

Claim 4.4. 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 any query q with $q < |w|$ has to be asked before stage $t(|w|)$ by the definition of the function t.

Theorem 4.5. *Let V be a* PSPACE *scanning function that is g-filling for a polynomial bound g. If Z is* PSPACE *random, then so is $V(Z)$.*

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

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

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

By the claim above and since B is a martingale, this definition is independent of t. Also, if we let $t = g(|w|)$ be the least allowed value, by the hypotheses that G can be computed in polynomial space and that g is a polynomial, D can be computed in polynomial space.

The rest of the argument very closely follows Buhrman, van Melkebeek, Regan, Sivakumar and Strauss [2]; we include it to verify that it works in the new notational setting.

Claim 4.6. D is a martingale.

Fix a string w of length n. To show that $2D(w) = D(w0) + D(w1)$, we partition the strings of length $t(n+1)$ into three sets. Let P_0 be the set of strings α of length $t(n+1)$ such that $\alpha \sim_V w0$ and $\alpha \not\sim_V w1$. Let P_1 be the set of strings α of length $t(n+1)$ such that $\alpha \sim_V w1$ and $\alpha \not\sim_V w0$. Let P be the set of strings α of length $t(n+1)$ such that $\alpha \sim_V w0$ and $\alpha \sim_V w1$. For $\alpha \in P_0$ we have $m(\alpha, w0) = m(\alpha, w) + 1$, and similarly for $\alpha \in P_1$. If $\alpha \in P$ then $m(\alpha, w0) = m(\alpha, w1) = m(\alpha, w)$. Now $D(w0) + D(w1)$ is given by

$$\begin{aligned}
&= \sum_{\alpha \in P_0} B(\alpha) 2^{m(\alpha,w0)-t(n+1)} + \sum_{\alpha \in P_1} B(\alpha) 2^{m(\alpha,w1)-t(n+1)} \\
&\quad + 2 \sum_{\alpha \in P} B(\alpha) 2^{m(\alpha,w)-t(n+1)} \\
&= 2 \sum_{\alpha \in P_0} B(\alpha) 2^{m(\alpha,w)-t(n+1)} + 2 \sum_{\alpha \in P_1} B(\alpha) 2^{m(\alpha,w)-t(n+1)} \\
&\quad + 2 \sum_{\alpha \in P} B(\alpha) 2^{m(\alpha,w)-t(n+1)} \\
&= 2D(w)
\end{aligned}$$

where the last equality is due to the fact that the definition of D above is independent of the parameter t.

Claim 4.7. D succeeds on Z.

We may assume that $B(x) > 0$ for each x. The Savings Lemma (see e.g. Nies [8, 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 $V(Z)$ such that

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

By definition of $V(Z)$ 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 4.8. *Let S be a polynomial time computable permutation of $\{0\}^*$ such that S^{-1} is polynomially bounded. If Z is* PSPACE *random, then so is $Z \circ S$.*

Proof. The permutation S can be viewed as a scanning function V that only looks at the length of the input: let $V(\alpha) = S(|\alpha|)$. We have $Z \circ S = V(Z)$. By hypothesis on S, the scanning function V is polynomially filling. So $Z \circ S$ is PSPACE-random by the theorem. □

Acknowledgments. The authors would like to thank Alexander Galicki and Wolfgang Merkle for discussions and comments.

References

[1] Vasco Brattka, Joseph S. Miller and André Nies. Randomness and differentiability. *Transactions of the American Mathematical Society*, 368:581–605, 2016. See also technical report on http://arxiv.org/abs/1104.4465.

[2] Harry Buhrman, Dieter Van Melkebeek, Kenneth W. Regan, D. Sivakumar and Martin Strauss. A generalization of resource-bounded measure, with application to the BPP vs. EXP problem. *SIAM Journal on Computing*, 30(2):576–601, 2000.

[3] Santiago Figueira and André Nies. Feasible analysis, randomness, and base invariance. *Theory of Computing Systems*, 56(3):439–464, 2015, DOI 10.1007/s00224-013-9507-7.

[4] Denis R. Hirschfeldt, André Nies and Frank Stephan. Using random sets as oracles. *Journal of the London Mathematical Society*, 75(3):610–622, 2007.

[5] Bjørn Kjos-Hanssen, Paul Kim Long V. Nguyen and Jason M. Rute. Algorithmic randomness for Doob's martingale convergence theorem in continuous time. *Logical Methods in Computer Science*, 10(4:12):1–35, 2014.

[6] Per Martin-Löf. The definition of random sequences. *Information and Control*, 9:602–619, 1966.

[7] André Nies. Lowness properties and randomness. *Advances in Mathematics*, 197(1):274–305, 2005.

[8] André Nies. *Computability and Randomness*, volume 51 of *Oxford Logic Guides*. Oxford University Press, Oxford, 2009. 444 pages. Paperback version 2011.

[9] Michiel van Lambalgen. The axiomatization of randomness. *The Journal of Symbolic Logic*, 55(3):1143–1167, 1990.

(A. Nies) Department of Computer Science, The University of Auckland, Private Bag 92019, Auckland, New Zealand

E-mail address: andre@cs.auckland.ac.nz

(F. 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

E-mail address: fstephan@comp.nus.edu.sg