Closure of resource bounded randomness notions under polynomial time permutations

André Nies, University of Auckland
Joint work with Frank Stephan, NUS

Aspects of Computation, IMS, Singapore, 2017

ABSTRACT. It is well known that computable 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.

On the one hand, 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 are in P.

On the other hand, we show 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.

A hierarchy of randomness notions

Formal randomness notions for an infinite bit sequence $Z \in \{0,1\}^{\mathbb{N}}$ can be introduced via algorithmic tests.

- Martin-Löf (1966) defined algorithmic tests as uniformly Σ^0_1 sequences $\langle G_m \rangle_{m \in \mathbb{N}}$, where $G_m \subseteq \{0,1\}^{\mathbb{N}}$ and $\lambda G_m \leq 2^{-m}$; Z is Martin-Löf-random if $Z \notin \bigcap G_m$ for each such test.
- Schnorr (1971) used a more restrictive notions of tests: he required that the measure λG_m be computable uniformly in m. This yields the weaker notion of Schnorr randomness.
- He also introduced computable randomness of Z: no computable betting strategy succeeds when betting along Z.

Martin-Löf rd. $\Rightarrow$ computably random $\Rightarrow$ Schnorr random

Criteria for formal randomness notions

1. A reasonable notion should imply intuitive properties of random sequences: e.g. Borel normal; no infinite "part" (bits in a recursive set of positions) is computable; parts are independent.

The first two properties hold for any Schnorr random Z. However, there is a computably random Z such that the even-positioned bits are Turing equivalent to the odd-positioned bits (take Z in a high minimal degree). This is no longer possible for a ML-random Z.

2. A reasonable randomness notion should be invariant under computable measure-preserving operations on $\{0,1\}^{\mathbb{N}}$.
E.g. a computable permutation of the bits.

It follows from the definitions that ML- and Schnorr randomness are invariant under computable permutations. For computable randomness, it's also true, but requires more work.

Preliminary:

resource–bounded

randomness notions

Algorithmic tests in the resource–bounded setting

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 $x \in \{0,1\}^*$.

M succeeds on a set Z if $\limsup_n M(Z \upharpoonright n) = \infty$.

- A martingale M is polynomial time computable if on input x one can determine the rational $M(x)$ in polynomial time.
- Here a positive rational number q is presented by the pair $\langle k, n \rangle$ of natural numbers (written in binary) such that $q = k/n$ in lowest terms.
- Similarly, define when a martingale is polynomial space computable.

Polynomial time and polynomial space randomness

- $Z \in \{0,1\}^{\mathbb{N}}$ is polynomial time random if no polynomial time computable martingale succeeds on Z.
- This was briefly defined by Schnorr (1971). Lutz (1990) studied resource bounded martingales. Ambos-Spies and Mayordomo looked at the associated randomness notions (for sets of strings, rather than bit sequences). Polynomial randomness was studied in more explicit form in Yongge Wang's 1996 thesis (Uni Heidelberg)[a].
- In a similar way one defines polynomial space randomness.

[a]See the 1996 survey "Resource bounded measure and randomness" by Ambos-Spies and Mayordomo for background and references.

Tally languages

From now on we identify a bit sequence $Z \in \{0,1\}^{\mathbb{N}}$ with a subset of $\{0\}^*$ (a tally language):

$$Z \in \{0,1\}^{\mathbb{N}} \text{ is seen as } \{0^k \colon Z(k) = 1\}.$$

In this way we can apply the notions of complexity theory to Z.

Polytime randoms exist in all superpolynomial time classes

Recall that a function h is time constructible if the function $0^n \to h(n)$ can be computed in time $O(h(n))$.

Existence result. Suppose the function $h(n)$ is time constructible and eventually dominates each polynomial (e.g. $h(n) = n^{\log n}$). There is polynomial time random Z computable in time $O(h(n))$.

Basic idea (Schnorr, essentially): Let $\langle M^e \rangle_{e \in \mathbb{N}}$ be an effective list of all polytime martingales with initial capital 1, and let

$$L = \sum 2^{-e} M_e.$$

Z is a bit sequence along which L does not increase. To get $L \in \mathsf{DTIME}(h)$, let M_e "kick in" only on sufficiently long strings.

Base invariance of polynomial time randomness

- Polynomial time martingales and polytime randomness in base $b > 2$ are defined similar to the above.
- Polynomial time randomness can be seen as a property of real numbers, rather than of sequences of digits for a fixed base:

Theorem (Figueira, N 2013) For a real number $r \in [0, 1]$,

- let Z be the binary expansion of r
- let Y be the expansion of r in base b.

Z is polynomial time random $\iff$ Y is polynomial time random.

A similar result was previously known for computable randomness. In both cases, one uses a characterization of the randomness notion for reals via differentiability of certain effectively computable functions.

Polynomial time functions $g\colon (0,1) \to \mathbb{R}$

- A sequence of rationals $(p_i)_{i\in\mathbb{N}}$ is called a Cauchy name if $\forall k > i\,|p_i - p_k| \leq 2^{-i}$
- In the efficient setting, one uses a compact set of Cauchy names to represent reals.
- A sequence $(a_i)_{i\in\mathbb{N}}$, where $a_i \in \{-1, 0, 1\}$, $a_0 = 0, a_1 = 1$, determines the real $\sum_{i\in\mathbb{N}} a_i 2^{-i} \in (0,1)$.
- A function $g\colon (0,1) \to \mathbb{R}$ is called polynomial time computable if there is a polynomial time oracle Turing machine turning every such Cauchy name for x into a Cauchy name for $g(x)$.

Functions such as e^x, x^2, $\sin x$ are polynomial time computable.

Characterising polynomial time randomness via differentiability

Theorem (N., STACS 2014)

A real z is polynomial time random $\iff$
$g'(z)$ exists for every nondecreasing polynomial time function g.

Permutations with polynomially unbounded inverse

Proposition. Let S be a polynomial time computable 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 $Z \subseteq \{0\}^*$ (computable in time $2^{O(n)}$) such that $Z \circ S$ is not polynomial time random.

Idea:

- build polytime random Z as above: let $L = \sum 2^{-e} M_e$; Z is a bit sequence along which L does not increase.
- If $S(n)$ is much smaller than n, we can predict $Z(S(n))$ in time polynomial in n. So $Z \circ S$ is not polytime random.

A polytime computable permutation with unbounded inverse

A permutation S as in the Proposition exists:

For the k-th polynomial p_k, if $n = \langle k, i \rangle$ is least such that $p_k(\langle k, 0 \rangle) \leq n$, put a cycle into S as follows:

$$n \to \langle k, 0 \rangle \to \langle k, 1 \rangle \to \ldots \to \langle k, i-1 \rangle \to n.$$

Hence one should only look for closure under polynomial time computable permutation S such that S^{-1} is polynomially bounded.

Part 1: If $\mathsf{P} = \mathsf{PSPACE}$ then closure holds

Based on the arguments in Buhrman, Melkebeek, Regan, Sivakumar and Strauss (2000), we show that polynomial space randomness is closed under polynomial time computable permutations S with polynomially bounded inverse.

So if $\mathsf{P} = \mathsf{PSPACE}$, this closure property applies to polynomial time randomness as well.

- Let $g(n)$ be a polynomial bound for S^{-1}.
- Suppose that a polynomial space martingale B succeeds on $Z \circ S$. We may assume that B has the savings property:

$$\text{if } \sigma \preceq \tau \text{ then } B(\tau) \geq B(\sigma) - 2.$$

The "savings – gale" $B \in \mathsf{PSPACE}$ succeeds on $Z \circ S$.
We define a martingale $D \in \mathsf{PSPACE}$ that succeeds on Z.

For bit strings α, w such that $|\alpha| = g(|w|)$, we write $\alpha \sim_S w$ if $\alpha = w \circ S$ whenever both sides are defined, namely

$$k < |\alpha| \wedge S(k) < |w| \Rightarrow \alpha(k) = w(S(k)).$$

There are $2^{g(|w|)-|w|}$ many α's of length $g(|w|)$ such that $\alpha \sim_S w$.
To define $D(w)$ we take their average:

$$D(w) = 2^{|w|-g(|w|)} \sum B(\alpha) \, [\![|\alpha| = g(|w|) \ \wedge \ \alpha \sim_S w]\!]$$

D is a martingale because if $|\alpha| = g(n+1)$ and $\alpha \sim_S w$, then either $\alpha \sim_S w0$ or $\alpha \sim_S w1$. Since g is a polynomial, D is in PSPACE. B succeeds on $Z \circ S$, so for each c we can find $w \prec Z$ such that $B(\alpha) \geq c$ for each $\alpha \sim_S w$. So D succeeds on Z.

Extension to scanning functions

A scanning function is a function $V\colon \{0,1\}^* \to \{0\}^*$ such that $V(\alpha) \neq V(\alpha \upharpoonright i)$ for each bit string α and each $i < |\alpha|$. 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.

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

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 $w(x) = \alpha(j)$.
The argument given for permutations actually shows the following.

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

Part 2: If BPP is large then closure fails

The class BPP

- BPP is short for “bounded-error probabilistic polynomial time”. (But BEPP is too long.)
- BPP contains the languages for which there is a polynomial time algorithm that uses random bits and obtains the answer with error probability $< 1/2 - \epsilon$ for some fixed ϵ.
- By repeating runs independently, we can ensure that the error probability on inputs of length n is at most $2^{-q(n)}$ for a given polynomial q.

Example of a problem in BPP that is not known to be in P: polynomial identity testing. Polynomials are given as arithmetical circuits, have to test whether they are equal.

If BPP contains a time class larger than P then closure fails

Theorem. Assume that $\mathsf{DTIME}(h) \subseteq \mathsf{BPP}$ for some time constructible function h that eventually dominates each polynomial, e.g. $n^{\log n}$.
Then there is a polynomial time random set $Z \in \mathsf{DTIME}(2^{3 \cdot n})$ and a permutation S with $S, S^{-1} \in \mathsf{P}$ such that

$$Z \circ S \text{ is not polynomial time random.}$$

We may assume that $h(n) \leq n^{\log n}$.

Let M be a martingale in $\mathsf{DTIME}(h)$ that bets only on odd positions, and dominates up to a multiplicative constant all polynomial time martingales that bet only on odd positions. Let

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

A is computable in time h and hence by assumption in BPP. Let $B \subseteq \{0\}^*$ be a set in $\mathsf{DTIME}(2^{5\cdot n})$ so that no martingale in $\mathsf{DTIME}(2^{4\cdot n})$ succeeds along B. Define $Z \subseteq \mathbb{N}$ as follows:

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

$$Z = \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|c|c}\hline \mathrm{B} & \mathrm{A} & \mathrm{B} & \mathrm{A} & \mathrm{B} & \mathrm{A} & \mathrm{B} & \mathrm{A} & \mathrm{B} & \mathrm{A} & \mathrm{B} & \mathrm{A} & \mathrm{B} & \dots \\ \hline \end{array}$$

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

Z is polynomial time random

$$Z = \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|c|c} \hline \mathrm{B} & \mathrm{A} & \mathrm{B} & \mathrm{A} & \mathrm{B} & \mathrm{A} & \mathrm{B} & \mathrm{A} & \mathrm{B} & \mathrm{A} & \mathrm{B} & \mathrm{A} & \mathrm{B} & \dots \\ \hline \end{array}$$

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

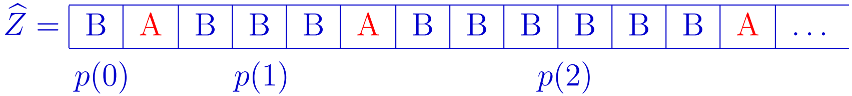
Assume that a polynomial time martingale L succeeds on Z. We may suppose L bets only on odd positions (the A's), or only on even positions (the B's).

- The case "odd positions" cannot happen: by definition of A, the capital of the universal M, and hence of L, is bounded along Z.
- In the case "even positions", we define a martingale $N \in \mathsf{DTIME}(nh(n) + h(n))$ that succeeds on B, contradiction.
- N computes the values of Z at the even positions using that $A \in \mathsf{DTIME}(h)$. It doesn't need to bet on these values; they are only needed to determine the bets of L at the odd positions.

A reshuffling $\widehat{Z}$ of Z

Since $A \in \mathsf{BPP}$, there is a randomised algorithm $\mathcal{R}$ that computes $A(x)$ on input $x \in \{0,1\}^{2n+1}$ with error probability 2^{-3n-2}. This takes time $p(n)$ for some polynomial p, and hence uses at most $p(n)$ random bits.

Let $\widehat{Z}$ consisting for $n = 0, 1, \ldots$ of $p(n)$ bits taken from B, followed by the bit $A(Z \upharpoonright 2n+1)$.



Then $\widehat{Z} = Z \circ S$ for the right permutation S, and $S, S^{-1} \in \mathsf{P}$.

The reshuffling $\widehat{Z}$ is not polynomial time random

$\widehat{Z} =$	B	A	B	B	B	A	B	B	B	B	B	B	A	$\ldots$
	$p(0)$			$p(1)$						$p(2)$				

Recall algorithm $\mathcal{R}$ computes $A(x)$ on input $x \in \{0,1\}^{2n+1}$ with error probability 2^{-3n-2}.

Idea: use B as a reservoir of random bits.

The probability that a string y of length $p(n)$ miscomputes $A(x)$ for some x of length $2n+1$ is at most $2^{2n+1} \cdot 2^{-3n-2} = 2^{-n-1}$. Whether y miscomputes some x can be determined in time $O(2^{3n})$.

Starting with 2^{-n-1} of the initial capital set aside for this purpose, we can bet on the set of miscomputing y. On each such y we gain SGD 1. If the portion of B's of length $p(n)$ is a miscomputing y i.o. then the martingale succeeds on the sequence B.

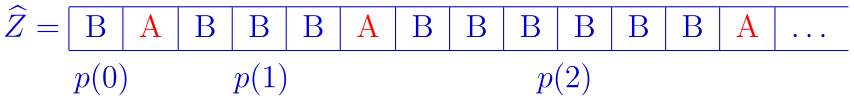
The reshuffling $\widehat{Z}$ is not polynomial time random

$$\widehat{Z} = \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|c|c}\hline \mathrm{B} & \mathrm{A} & \mathrm{B} & \mathrm{B} & \mathrm{B} & \mathrm{A} & \mathrm{B} & \mathrm{B} & \mathrm{B} & \mathrm{B} & \mathrm{B} & \mathrm{B} & \mathrm{A} & \ldots \\ \hline \multicolumn{1}{c}{p(0)} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{p(1)} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{p(2)} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{} & \end{array}$$

- The martingale explained above can be computed in time $O(2^{4k})$.
- This uses that $p(n+1) - p(n) \leq p(n)$ for a.e. n. For $k = p(n)$ we have to average over at most 2^k many y's satisfying a condition that can be checked in time $O(2^{3n})$. This yields the exponent 4.

Since B is $O(2^{4n})$-random, almost every such portion of B's computes the value of $\widehat{Z}$ at the "A" that comes after correctly.

The reshuffling $\widehat{Z}$ is not polynomial time random



To repeat, almost every such portion of B's of length $p(n)$ computes the value of $\widehat{Z}$ at the "A" that comes after correctly. We use this to define a polynomial time martingale that succeeds on $\widehat{Z}$. To make bet at the position r_n of the n-th "A":

- From $\widehat{Z} \upharpoonright r_n$ determine $x = Z \upharpoonright 2n + 1$
- let $y =$ the portion of B's preceding the n-th "A"
- run the algorithm $\mathcal{R}(x)$ with random bits y, and bet half of the capital on the value predicted by $\mathcal{R}$

For almost every n, this martingale gains by a factor 1.5 at r_n.

Question

PP denotes probabilistic polynomial time: w is in the language if the majority of computations of a polynomial time NTM on input w is accepting. Clearly $\mathsf{PP} \subseteq \mathsf{PSPACE}$.

Is the assumption $\mathsf{P} = \mathsf{PP}$ sufficient for closure of polynomial time randomness under permutations S such that $S, S^{-1} \in \mathsf{P}$?

Reference: upcoming paper “Closure of resource bounded randomness notions under polynomial time permutations”, available on Nies’ web site.