CDMTCS
Research
Report
Series

Lowness for Computable and Partial Computable Randomness

André Nies
University of Auckland

CDMTCS-363
April 2009

Centre for Discrete Mathematics and
Theoretical Computer Science

Lowness for computable and partial computable randomness

André Nies

Abstract. Ambos-Spies and Kucera [**1**, Problem 4.8] asked whether there is a non-computable set which is low for the computably random sets. We show that no such set exists. The same result holds for partial computable randomness. Each tally language that is low for polynomial randomness is on a polynomial time tree of bounded width.

This research was done in 2003 but has not been published so far.

1. Introduction

Consider a (relativizable) class $\mathcal{C}$ of sets. An oracle set A is called *low for* $\mathcal{C}$ if $\mathcal{C}^A = \mathcal{C}$. For instance, if $\mathcal{C}$ is the class of Δ^0_2 sets, then lowness for $\mathcal{C}$ coincides with the usual lowness $A' \le_T \emptyset'$.

The case where $\mathcal{C}$ is a randomness property is in the focus of current interest. A *martingale* is a function $M : 2^{<\omega} \mapsto \mathbb{R}^+_0$ such that $M(\lambda) \le 1$, and M has the martingale property $M(x0) + M(x1) = 2M(x)$.

We first study lowness for the computably random sets, namely

$$\mathsf{CR} = \{X : \textit{no computable martingale succeeds on } X\}.$$

We refer to [**1, 3**] for motivation and terminology. Using A-computable martingales gives a generally smaller class CR^A. A is low for computably random (Low(CR)) if CR^A is as large as possible, namely $\mathsf{CR}^A = \mathsf{CR}$.

All martingales can be assumed $\mathbb{Q}$-valued, which is no restriction as far as the randomness notions are concerned.

Theorem 1.1. *Each low for computably random set is computable.*

Now we consider time bounded martingales. Consider a time class $\mathcal{D}$ which is closed downward under polynomial time Turing reducibility $\le^p_T$. Then $\mathcal{D}-\mathsf{Rand}$ is the randomness notion given by martingales in $\mathcal{D}$. Note that Low($\mathcal{D}-\mathsf{Rand}$) is closed downward as well.

A language $A \subseteq \{0^*\}$ is called tally.

Theorem 1.2. *Let $\mathcal{D} = \bigcup_j \textsc{Dtime}(g_j)$, where $(g_j)_{j\in\mathbb{N}}$ is a family of time bounds containing the polynomials and closed under composition and multiplication (for instance, $\mathcal{D} = \textsc{Ptime}$). If the language A is tally and low for $\mathcal{D}$-random, then A is on a tree in $\mathcal{D}$ of width bounded by a constant.*

This does not imply that A is itself in $\mathcal{D}$. For instance, the construction of a supersparse set A puts A on a polynomial time tree of width bounded by 2, but A is only in Dext. However, we have:

Corollary 1.3. *If B is in Low($\mathcal{D}-\mathsf{Rand}$) then B is computable.*

Proof. This follows because the tally set $\{0^n : n \in B\}$ is Turing equivalent to B and computable being an isolated path on a computable tree. $\square$

We don't know whether, say, a language in Low(Ptime$-\mathsf{Rand}$) is already in Ptime, even when the language is tally.

2. Preliminaries

Given v, let

$$\widehat{M}(y) = \widehat{M}(y) = \max\{M(y') : v \preceq y' \preceq y \ \&\ M(y')\text{defined}\}.$$

Kolmogorov: for $M(v) < b$,

$$\mu\{z \succeq v : \widehat{M}(v) \geq b\} \leq M(v)/b2^{-|v|} \tag{1}$$

The following is fairly trivial but very useful.

Lemma 2.1 (Non-ascending path trick, NAPT). *Suppose M is a martingale which is computable in running time τ. Then, for each string z and each $u > |z|$ we can compute in time $u\tau(u)$ a string $w \succ z$, $|w| = u$, such that $M(w \upharpoonright q+1) \leq M(w \upharpoonright q)$ for each q, $|z| \leq q < u$.*

We say that a martingale B has the "savings property" if

$$x \prec y \Rightarrow B(y) \geq B(x) - 2. \tag{2}$$

It is known (see [**3**]) that the relevant randomness notions may be defined in terms of $\mathbb{Q}$-valued martingales with the savings property.

A *martingale operator* is a Turing functional L such that, for each oracle X, L^X is a total martingale. For a string γ, we write $L^\gamma(x) = p$ if this oracle computation converges with all oracle questions less than γ. To prove Theorems 1.1 and 1.2 we will define a martingale operator L (which can be computed in quadratic time). We will apply the following purely combinatorial Lemma to $N = L^A$ and the family (B_i) of martingales with the savings property characterizing the randomness notion in question. It says that for some positive linear combination M of the martingales B_i, and for some d, $N(w) \geq 2^d$ implies $M(w) \geq 2$ in an interval $[v]$, while $M(v) < 2$.

Lemma 2.2. *Let N be any martingale such that $N(\lambda) \leq 1$. Let $(B_i)_{i\in\mathbb{N}}$ be some family of martingales with the savings property (2). Assume that*

$$S(N) \subseteq \textstyle\bigcup_i S(B_i).$$

Then there are $v \in 2^{<\omega}$ and $d \in \mathbb{N}$ and a martingale M which is a finite linear combination $\sum_{i=0}^n q_i B_i$ with rational positive coefficients such that $M(v) < 2$ and

$$\forall x \succ v[\ N(x) \geq 2^d \Rightarrow M(x) \geq 2]. \tag{3}$$

Proof. If the Lemma fails, then for each linear combination $M = \sum_{i=0}^n q_i B_i$, $q_i \in \mathbb{Q}^+$

$$\forall v \forall d\ [M(v) < 2 \Rightarrow \exists w \succ v\ (N(w) \geq 2^d \ \&\ M(w) < 2)]. \tag{4}$$

We define a sequence of strings $v_0 \prec v_1 \prec \ldots$ and rationals $q_i > 0$ such that

$$N(v_n) \geq 2^n - 1 \;\&\; \sum_{i=0}^{n} q_i B_i(v_n) < 2. \tag{5}$$

Let $v_0 = \lambda$ and $q_0 = 1$, so that (5) holds for $n = 0$. Now suppose that $n > 0$ and v_{n-1}, q_{n-1} have been defined. Let

$$p_n = \tfrac{1}{2} 2^{-|v_{n-1}|}(2 - \textstyle\sum_{i=0}^{n-1} q_i B_i(v_{n-1})),$$

so that $M(v_{n-1}) < 2$ where $M = \sum_{i=0}^{n} q_i B_i$ (note that $0 < q_n \leq 1$). Applying (4) to $v = v_{n-1}, d = n$ there is $v_n = w \succ v$ such that $N(w) \geq 2^n$ and $M(w) < 2$.

If $Z = \bigcup_n v_n$, then N succeeds on Z (interestingly, not necessarily in the effective sense of Schnorr). On the other hand, for each $n \geq i$, $q_i B_i(v_n) < 2$. Since B_i has the savings property (2), $\limsup_n B_i(Z \restriction n) \leq 2 + 2/q_i$, so B_i does not succeed on Z. $\Box$

A *partial computable martingale* is a partial computable function $M : 2^{<\omega} \mapsto \mathbb{Q}$ such that $\mathrm{dom}(M)$ is $2^{<\omega}$, or $2^{\leq n}$ for some n, $M(\lambda) \leq 1$, and M has the martingale property $M(x0) + M(x1) = 2M(x)$ whenever $x0, x1$ are in the domain. Clearly there is an effective listing $(M_e)_{e \in \mathbb{N}}$ of partial computable martingales with range included in $[1/2, \infty)$. We let τ_e be the partial computable function such that $\tau_e(n) \sim$ the maximum running time of $M_e(w)$ for any w of length n (includes the linear slow down since we need an effective listing).

3. Proof of Theorem 1.1

Remarks in brackets [...] refer to later adaptation of the proof to the time-bounded case, and can be ignored at first reading. Fix an effective listing $(\eta_m)_{m \geq 1}$ of all triples

$$\eta_m = \langle e, v, d \rangle \tag{6}$$

where v is a string, and $e, d \in \mathbb{N}$ (e is an index for a partial computable martingale M_e). We think of η_m as a witness as in Lemma 2.2, where (B_i) is the family of all (total) computable martingales with the savings property (not an effective listing).

We will independently build martingale operators L_m for each $m \geq 1$ which have value 2^{-m} on any input of length $\leq m$. L_m is computable in linear time, for a fixed constant. Then $L = \sum_{m \geq 1} L_m$ is a martingale operator (L is $\mathbb{Q}$-valued since the contributions of L_m, $m > |w|$, add up to $2^{-|w|}$), and L is computable in quadratic time.

We define L in order to ensure that for each A, if $N = L^A$ and $S(N) \subseteq \bigcup_i S(B_i)$ (which is the case if A is low for computably random), then we can compute A. The computation procedure for A is based on a witness $\eta_m = \langle e, v, d \rangle$ given by Lemma 2.2, so M_e is total. Since we cannot determine the witness effectively, to make L a martingale operator we need to consider all η_m together, including those where M_e is partial.

The idea how to compute A is this. Once L is defined, if η_m is a witness for Lemma 2.2 where $N = L^A$, let $M = M_e$ and consider the tree

$$T_m = \{\gamma : \forall w \succeq v (L_m^\gamma(w) \geq 2^d \Rightarrow M(w) \geq 2)\}.$$

Since η_m is a witness and $L^A \geq L_m^A$, A is a path of T_m. Let $k = 2^{d+m}$, and let $\mathcal{S}_k$ denote the set of k-element sets of strings of the same length. Let α, β range over elements of $\mathcal{S}_k$. We write α_r for the r-th element in lexicographical order

$(0 \leq r < k)$, and identify α with the string $\alpha_0\alpha_1 \ldots \alpha_{k-1}$. For each α, we ensure that $\alpha \not\subseteq T_m$, in an effective way: given α, we are able to find $s < k$ such that $\alpha_s \notin T_m$. This will allow us to determine a tree $R \supseteq T_m$ such that for each j, the j-th level $R^{(j)}$ has size $< k$ and we can compute that level Then we can compute A: fix j_0 sufficiently large so that only one extension of $A \upharpoonright j_0$ exists in $R^{(j)}$, for each $j \geq j_0$. This extension must be $A \upharpoonright j$ since A is a path of T_m and $T_m \subseteq R$. So, given input $p \geq j_0$ to compute $A(p)$ we output the last bit of that extension for $j = p + 1$.

Let $z_0, \ldots, z_{k-1}$ be the strings of length $d+m$ in lexicographical order. We describe in more detail the strategy which, given α, produces an s such that $\alpha_s \notin T_m$. Suppose $w \succ v$ is a string such that $M(w) < 2$, and no value $L^\gamma(w')$ has been declared for any $w' \preceq w$ (we call w an *α-destroyer*). In this case we may define $L_m(w) = 2^{-m}$ regardless of the oracle. For each $s < k$, we ensure $L_m^{\alpha_s}(wz_s) = 2^d$, by betting all the capital along z_s from the end of w on. Since $M(w) < 2$, by the NAPT (2.1) we can compute s such that $M(wz_s) < 2$. So $x = wz_s$ is a counterexample to (3), so that $\alpha_s \notin T_m$.

We want to carry out this strategy independently for different α. To do so we assign to each α a string y_α. Given η_m, let

$$\widehat{M}(y) = \max\{M(y') : v \preceq y' \preceq y\}.$$

The assignment function $G_m : \mathcal{S}_k \mapsto \{0,1\}^*$ mapping α to y_α (which only is defined when $M_e(v) < 2$) will satisfy the following.

- (G1) The range of G_m is an antichain of strings y such that $\widehat{M}(y) < 2$
- (G2) G_m and G_m^{-1} are computable in the sense that there is an algorithm to decide if the function is defined and in that case returns the correct value.

We cannot apply the strategy above with $w = y_\alpha$, since we first would need to recover α from w, which may take long or even forever (depending on M_e which may be partial), but also we want L_m to be total, and in fact to be computable in quadratic time. Instead we use the "looking-back" technique. Let $h_m(\alpha)$ be the number of steps required to check that $M_e(v) < 2$, $G_m^{-1}(y_\alpha \upharpoonright p)$ is undefined for $p = 0, \ldots, |y_\alpha| - 1$, and to compute $\alpha = G_m^{-1}(y_\alpha)$. Each $w \succ y_\alpha$ of length $h_m(\alpha)$ is a potential α-destroyer. Now we can recover α from w in linear time, and then define $L_m^{\alpha_r}$ above w according to the strategy above.

Given α, to find the actual α-destroyer w, first compute y_α, then $h_m(\alpha)$, and now use the NAPT to find $w \succ y_\alpha$ of length $h(\alpha)$ such that $M(w) < 2$. As explained above, use w to determine which α_r is not on T_m.

The actual choice of the G_m is irrelevant so long as (G1) and (G2) hold. So we defer defining the G_m. Note that the time to compute G_m and G_m^{-1} will be closely related to the running time τ_M of M, since we need to find strings y such that $\widehat{M}(y) < 2$. The following procedure will be used to define L_m and to compute $h_m(\alpha)$.

```
Procedure $P_m$ ($\eta_m = \langle e, v, d\rangle$)
Input $x$
    1. Let $p = 0$
    2. $y = x \upharpoonright p$
    3. Attempt to compute $\alpha = G^{-1}(y)$. If defined, output $\alpha$ and $h =$ the number
       of steps used so far.
```

```
4. $p \leftarrow p+1$. If $p < |x|$ Goto 2.
```

Construction of L_m. We define L_m by declaring axioms of the form $L_m^\gamma(w) = p$, in such a way that

(a) $|\gamma| \leq |w|$ and one can determine in time $O(|w|)$ whether an axiom $L^\gamma(w) = p$ has been declared, and
(b) whenever distinct axioms $L^\gamma(w) = p$ and $L^\delta(w) = q$ are declared then γ, δ are incompatible.

Then we let $L_m^X(w) = p$ if some axiom $L_m^\gamma(w) = p$ has been declared for $\gamma \subseteq X$, or else if p is the "default value" 2^{-m}. Clearly $L_m^X(w)$ can be determined in time $O(|w|)$ using oracle X.

Given a string x, we declare no axiom for x unless in $|x|$ steps we can determine that $\eta_m = \langle e, v, d\rangle$, and that $M_e(v)$ converges in $< |x|$ steps with value < 2. If so, run at most $|x|$ steps of procedure $P_m(x)$. If there is an output α, h, then let $w = x \upharpoonright h$ and declare axioms as follows (implementing the strategy outlined above): Let $x = wz$. For each $s < k$, let $L_m^{\alpha_s}(x) = 0$ unless z is compatible with z_s. In that case, declare $L_m^{\alpha_s}(x) = 2^{-m+|z|}$ if $z \preceq z_s$, and $L_m^{\alpha_s}(x) = 2^d$ if $z_s \preceq z$. *End of construction.*

Clearly (a) holds. Moreover (b) is satisfied since the strings (y_α) form an antichain, we only declare axioms $L_m^\gamma(x) = p$, $y_\alpha \preceq x$ if $\gamma \in \alpha$, and the individual strings within α are incompatible. Finally L^X is a martingale for each oracle X.

Suppose (B_i) is the family of all (total) computable martingales with the savings property (2). If A is Low(CR), then $S(L^A) \subseteq \bigcup_i S(B_i)$. The linear combination M obtained in Lemma 2.2 is computable. So the following lemma suffices to compute A, since, as explained above, the existence of R implies that A is computable.

Lemma 3.1. *[Computing a thin tree] Suppose $\eta_m = \langle e, v, d\rangle$, where $M = M_e$ is total and M, v, d is a witness for (3) in Lemma 2.2 where $N = L^A$. Then there is a tree $R \supseteq T_m$ such that for each j, the j-th level $R^{(j)}$ has size $< k = 2^{d+m}$ and we can compute that level from j [here j is given in unary].*

Proof. Let $R^{(0))} = \{\lambda\}$. Suppose $j > 0$ and we have determined $R^{(j-1)}$. Carry out the following to determine $R^{(j)}$:

```
1. Let $F$ be the set of strings of length $j$ that extend strings in $R^{(j-1)}$ (so $|F| = 2|R^{(j-1)}|$) .
2. While $|F| \geq k$: Let $\alpha$ be the lexicographically leftmost size $k$ subset of $F$.
    (a) Compute $y = G(\alpha)$.
    (b) Apply procedure $P_m$ to $y$ to compute $h = h(\alpha)$.
    (c) By NAPT find $w \succ y_\alpha$ of length $h$ such that $\widehat{M}(w) < 2$.
    (d) Search for $r < k$ such that $\widehat{M}(wz_r) < 2$. Remove $\alpha_r$ from $F$.
3. Let $R^{(j)} = F$.
```

□

To conclude the recursion theoretic case it remains to define the G_m. We prove a lemma which will be useful in the time-bounded case as well. Recall that $\widehat{M}(y) = \max\{M(y') : v \preceq y' \preceq y\}$. We use the following instance of Kolmogorov's inequality: for $M(v) < b$,

$$\mu(\{z \succeq v : \widehat{M}(z) \geq b\}|v) \leq M(v)/b, \tag{7}$$

where $\mu(X|v)$ stands

Lemma 3.2 (The assignment function, recursion theoretic case). *Given η_m, suppose that $M(v) < b$, $b \in \mathbb{Q}$. Let*

$$P = \{y \succeq v : \widehat{M}(y) < b\},$$

and let $r \in \mathbb{N}$ be such that $2^{-r} \leq 1 - M_e(v)/b$. Then given i we can compute $y^{(i)}$ of length $i+r+1$ such that $y^{(i)} \in P$ and the strings $y^{(i)}$ form an antichain. If M is partial, we can compute $y^{(i)}$ for each i such that M is defined for strings of length up to $i+r+1$. [The computation takes time $O(i^2)\tau_M(|v|+i+r)$ where i is given in unary.]

Proof. Suppose inductively $y^{(q)}$ has been computed for $q < i$. Since $\sum_{q<i} 2^{-|y^{(q)}|} = 2^{-r}(1-2^{-i})$ and $2^{-r} \leq \mu(P|v)$ by Kolmogorov's inequality, one can compute $y \in P$ such that $|y| = i+r+1$ and $y_q \not\preceq y$ for all $q < i$. Let $y_i = y$. [To compute such a y efficiently, search for the least $u < i+1$ such that some $z = y^{(q)} \upharpoonright u$, $q < i$, has an extension $z\widehat{\ }h \in P$ ($h \in \{0,1\}$ which is not on any y_l ($l < i$). This needs at most $O(i^2)$ many computations $M(w)$, for strings w of length $< i+r+1$. Now extend $z\widehat{\ }h$ to a string y of length $i+r+1$ such that $M(y) < 2$, using the NAPT.] □

In the recursion theoretic case, let $b = 2$, and let n_α be a number greater than the length of each string in α, assigned to α in an effective 1-1 way. Let $G_m(\alpha) = y^{(n_\alpha)}$. Clearly (G1) and (G2) hold.

4. Proof of Theorem 1.2

A tally language A can be viewed as the set $\{n : 0^n \in A\}$. So to prove Theorem 1.2 we can adapt the previous proof. All relevant oracle queries in the definition of L_m are now in $\{0\}^*$. We will modify the definition of the assignment functions G_m. Recall that the martingale operator L is in quadratic time no matter how we specify G_m as long as (G1) and (G2) hold. Let $\{B_i\}$ be some list of all martingales in $\mathcal{D}$ with the savings property. Then a set which fails to be $\mathcal{D}$-random is already in $S(B_i)$ for some i. So if A is Low($\mathcal{D}$-random) then $S(L^A) \subseteq \bigcup_i S(B_i)$, and Lemma 2.2 yields a martingale $M = M_e$ and v, d such that (3) holds and $M \in \mathcal{D}$, since M is a linear combination of martingales in $\mathcal{D}$. Thus the running time τ_M is bounded by a function g_j from the list of time bounds determining $\mathcal{D}$.

We want to argue that $A \in \mathcal{D}$. The problem is that, with the current choice of G, the algorithm in the proof of Lemma 3.1 takes too long. We need a function G such that $|G(\alpha)| = O(|\alpha|)$, for in that case we can compute A with running time sufficiently close to τ_M so that $A \in \mathcal{D}$.

As usual we work in the context of the witness $\eta_m = \langle e, v, d\rangle$, and let $k = 2^{d+m}$. Recall that $\mathcal{S}_k$ denotes the set of k-element sets of strings of the same length. First we replace a string $\alpha = \alpha_0 \ldots \alpha_{k-1}$ in $\mathcal{S}_k$ by the pair $\langle p, j\rangle$, where $j = k|\alpha_0|$ and $p < 2^j$ is the lexicographical position of α. Clearly it suffices to define a map $\widetilde{G}$ with the properties (G1) and (G2) on all pairs $\langle p, j\rangle$ where $p < 2^j$, instead of on $\mathcal{S}_k$.

Suppose $r \in \mathbb{N}$ is such that $2^{-(r-1)} < 1 - M(v)/2$. Let $b = (M(v)+2)/2$, and as in Lemma 3.2 let $P = \{y \succeq v : \widehat{M}(y) < b\}$, so that we can compute on input j a string $y^j \in P$ of length $j+r+1$ such that the strings y^j form an antichain. The string $\widetilde{G}(p,j)$ will be an extension of y^j. Fix j and consider the martingale D given by

$$D(x) = M(vy^jx).$$

Then $2^{-r} < 1 - D(\lambda)/2$ since $D(\lambda) < b$. Let $Q = \{x : \forall z \preceq x\ D(z) < 2\}$.
Let

$$\beta(x) = 1 - \tfrac{D(x)}{2},$$

so that $\beta(x0) + \beta(x1) = 2\beta(x)$. Note that

$$\beta(x) \leq \mu(Q|x), \tag{8}$$

(thus $\beta(x)$ tells us how many extensions of x are suitable values $\widetilde{G}(p,j)$).

To each x of length at most j we will assign an interval I_x of the form $[p,q)$, where $p,q \in \mathbb{Q}$ and $0 \leq p \leq q \leq 1$ (thus I_x may be empty). Write $|I_x|$ for the length $q-p$. For each $i \leq j$ the nonempty intervals I_x, $|x| = i$, partition $[0,1)$ and are arranged according to the lexicographical order of x. Moreover, if $I_x \neq \emptyset$, then

$$|I_x| \leq \beta(x)2^{r-|x|}. \tag{9}$$

Let $I_\lambda = [0,1)$ so that (9) holds for $x = \lambda$. If I_x has been defined and $|x| < j$, distinguish three cases.

1.) $M(x1) \geq 2$ (hence $\beta(x1) \leq 0$). Let $I_{x0} = I_x$ and $I_{x1} = \emptyset$.
2.) $M(x0) \geq 2$ (hence $\beta(x0) \leq 0$). Let $I_{x1} = I_x$ and $I_{x0} = \emptyset$.
3.) Otherwise. Then split I_x in the proportion $\beta(x0) : \beta(x1)$. That is, let $s_x = \beta(x0)/(\beta(x0) + \beta(x1))$, let I_x be the left half open subinterval of I_x of length $s_x|I_x|$, and let $I_{x1} = I_x - I_{x0}$.

We check (9) by induction on $|x|$. In Case 1, $\beta(x0) \geq 2\beta(x)$, so (9) is true for $x0$ (and trivial for $x1$). Case 2 is similar. In Case 3 multiply (9) by s_x to obtain the inequality for $x0$, and multiply (9) by $1 - s_x$ to obtain it for $x1$ (using that $\beta_{x0} = \beta_x s_x$ and $\beta_{x1} = \beta_x(1 - s_x)$). We are now ready to give the procedure for $\widetilde{G}$.

To compute $\widetilde{G}(p,j)$ $(p < 2^j)$

1. Determine the unique x (and also the end points of I_x) such that $|x| = j$ and $p2^{-j} \in I_x$
2. Compute $p_0 \in \mathbb{N}$ least such that $p_0 2^{-j} \in I_x$. Let $q = p - p_0$
3. Let z be the $q{+}1$-st string in lexicographical order such that $x \prec z$, $|z| = i{+}r$ and $z \in Q$. Let $\widetilde{G}(p,j) = y^j z$

We first verify that z in step 3. exists. By (9) and (8),

$$2^i|I_x| \leq 2^r \mu(Q|x).$$

The quantity on the right bounds from below the number of extension z of x as in 3. But $p - p_0 < 2^j|I_x|$, since both p_0^{-j} and $p2^{-j}$ are in this half-open interval. So there are at least $p - p_0 + 1$ possible extensions z.

We check that $\widetilde{G}$ and its inverse are computable within the allowed the time bounds.

Claim 4.1. *Let* $b(i) = |v| + 2(i + r) + 1$

(i) *The computation of* $\widetilde{G}(p,i)$ *takes time*

$$O(i^2)\tau_M(b(i))$$

(ii) *The computation of $\widetilde{G}^{-1}$, on relevant inputs w of length $|w| = b(i)$ also takes time $O(i^2)\tau_M(b(i))$*

Proof. Given i, an *admissible M-computation (AMC)* is a computation $M(w)$ where $|w| \leq b(i)$.
(i) We show that $O(i^2)$ AMC are sufficient. Step 1 needs i evaluations $D(x)$, $|x| \leq i$, which means i AMC. Step 2 is trivial in terms of complexity, and Step 3 requires the constant amount of 2^r AMC. Finally, by Lemma 3.2, computing y^i requires i^2 AMC.
(ii) Given w, reject unless $|w| = b(i)$. Now use $O(i^2)$ AMC to compute y^i. Reject unless $y^i \prec w$. In that case write $y^i u = w$, and let $x = u \upharpoonright i$. Compute I_x using i AMC. Reject unless I_x is non-empty. In that case by (9), $|I_x| \leq 2^r 2^{-i}$, so we may within our time bound compute $\widetilde{G}(p, i)$ for all p such that $p2^{-i} \in I_x$ by (i) to see if the value is w. If not reject, else output $\langle p, i \rangle$.
To show A is in $\mathcal{D}$, it suffices to prove

Claim 4.2. *The function $0^j \mapsto R^{(j)}$ in Lemma 3.1 is in $\mathcal{D}$.*

For in that case, can compute A as before (fix j_0 sufficiently large so that only one extension of $A \upharpoonright j_0$ exists in $R^{(j)}$, for each $j \geq j_0$. This extension must be $A \upharpoonright j$ since A is a path of R. So, given input 0^p, $p \geq j_0$ to compute $A(0^p)$ we output the last bit of that extension for $j = p + 1$.)

Proof. Recall that (g_l) is the list of time bounds, called admissible. Let g, g' etc. denote admissible time bounds, and let $f(i)$ be the admissible bound which is the maximum of the bounds obtained in (i) and (ii) above. The highest cost is the computation of M values of long strings in Steps 2(c) and 2(d).

1. Let F be the set of strings of length j that extend strings in $R^{(j-1)}$.
2. While $|F| \geq k$: Let α be the lexicographically leftmost size k subset of F. *This loop is carried out for a constant number of times.*
   (a) Compute $y = G(\alpha)$. *Replacing α by p, i is polynomial time. Computing $y = \widetilde{G}(p, i)$ takes time $f(i)$, where $|y| = b(i)$.*
   (b) Apply procedure P_m to y to compute $h = h(\alpha)$. *This take one computation $G^{-1}(z)$ for each $z \preceq y$, and hence time $b(i)^2 f(i)$. Thus $h(\alpha)$, which is the number of steps taken, is bounded by an admissible bound $g(i)$. Let $u = g(i)$.*
   (c) By NAPT find $w \succ y_\alpha$ of length h such that $M(w) < 2$. *This needs u many M computations on strings of length $\leq u$, hence bounded by $u\tau_M(u)$*
   (d) Search for $r < k$ such that $M(wz_r) < 2$. Remove α_r from F. *Takes a constant number of M computations on strings of length $u + k$.*
3. Let $R^{(j)} = F$.

The running time to compute $0^j \mapsto R^{(j)}$ is therefore admissible.

5. Lowness for partial computable randomness

In this section we consider lowness properties defined in terms of two randomness notions $\mathcal{C}, \mathcal{D}$. Let Low$(\mathcal{C}, \mathcal{D})$ denote the class of oracles A such that $\mathcal{C} \subseteq \mathcal{D}^A$.

Theorem 5.1. *Each Low(PrecRand,CRand) set is computable.*

Lemma 5.2. *Let N be any total martingale such that $S(N) \subseteq$ Non-PrecRand. Then there are $v \in 2^{<\omega}$ and $d \in \mathbb{N}$ and p.c. martingale M such that $M(v) < 2$ and*

(a) $\{x \succeq v : \widehat{M}(x) < 2\}$ *is computable,*
(b) $\forall x \succeq v[\ N(x) \geq 2^d \Rightarrow \widehat{M}(x) \geq 2]$,

where $\widehat{M}(y) = \max\{M(y') : v \preceq y' \preceq y \ \&\ M(y') \text{defined}\}$.

We define the martingale operator L as before (the construction allowed for partial computable martingales M_e anyway). But we now must argue that A is computable based on a witness $\eta_m = \langle e, v, d\rangle$ for the weaker Lemma 5.2 where $M = M_e$.

Notice that the Kolmogorov inequality (1) is valid for partial martingales M. Moreover, by (a) of the Lemma and that inequality, we have the following version of the NAPT:

Lemma 5.3 (weak NAPT). *Given z, m such that $v \preceq z$, $\widehat{M}(z) < 2$ and $m \geq |z|$, one can compute $w \succeq z$ of length m such that $\widehat{M}(w) < 2$.*

□

By (b), A is on the tree

$$\widehat{T}_m = \{\gamma : \forall w \succeq v(L^{\gamma}_m(w) \geq 2^d \Rightarrow \widehat{M}(w) \geq 2)\}.$$

So as before it suffices to find a computable tree $R \supseteq \widehat{T}_m$ of width at most $k = 2^{d+m}$. The assignment function G_m defined after Lemma 3.2 is total, since the proof only relies on the fact that P is computable. The algorithm in Lemma 3.1 to compute R works in the new setting, by the weak NAPT and (a).

Proof of Lemma 5.2. We apply the following:

Claim 5.4. *There is a sequence $C_0, \ldots, C_n$ of p.c. martingales such that, if we also define $C_{-1} = 0$, there are v, d such that $M(v) < 2$ and*

(a*) $\forall w \succeq v \forall j \leq n[C_{j-1}(w) < 2\mathbb{R}AC_j(w)\downarrow]$
(b*) $\forall x \succeq v[\ N(x) \geq 2^d \Rightarrow \neg C_n(x) < 2]$.

Note that (a*) implies $C_0(w)\downarrow$ for all $w \succeq v$. We first check that the Claim suffices to prove the Lemma. Let $M = C_n$. Given input $w \succeq v$, consider the following procedure to decide whether $\widehat{M}(w) < 2$.

```
1. For $l = |v|$ to $|w|$: let $x = w \upharpoonright l$
2.     For $j = 0$ to $n$
3.     If $C_j(x) \geq 2$ output No; end.
4.     Next $j$
5. Next $l$
6. Output Yes
```

By (a*), $C_j(x)$ is defined in line 3. Thus the procedure decides correctly, which shows (a). For (b), note that if $N(x) \geq 2^d$, then by (b*), it is not the case that $M(x)$ is defined and less than 2. Hence we cannot reach line 6. This proves the Lemma.

We prove the claim. Assume for a contradiction that N is a total martingale such that $S(N) \subseteq$ Non-PrecRand, but the claim fails. Let $(B_i)_{i \in \mathbb{N}}$ be a list of the p.c. martingales B with the savings property

$$x \prec y \ \&\ B(y)\downarrow \Rightarrow B(y) \geq B(x) - 2),$$

so that (as in the case of computable randomness) $\mathsf{Non\text{-}PrecRand} = \bigcup_i S(B_i)$.

As before, we define a sequence of strings $v_0 \prec v_1 \prec \ldots$ such that N succeeds on $Z = \bigcup_n v_n$ but $Z \notin \bigcup_i S(B_i)$. We define p.c. martingales C_n of the form $\sum_{j \leq n} q_j B_{s_j}$ ($q_j \in \mathbb{Q}^+$). For each n, (a*) holds (so that (b* fails).

At step s, B_s is either included in the martingale C_n (a linear combination of B_i's) or B_s is made partial along Z.

We define v_s and the MG $C_s = \sum_{j \leq s} q_j B_{r_j}$ in such a way that $C_s(v_s) < 2$, and $B_t(v_t)$ is undefined for each $t \leq s$ not of the form r_j.

Let $n_{-1} = 0$, $v_{-1} = \lambda$ and $C_{-1} = 0$.

Step $s \geq 0$.

1. If there is $w \succeq v_{s-1}$ such that $C_{n_{s-1}}(w) < 2$ and $B_s(w)$ is undefined, then let $v_s = w$, $n_s = n_{s-1}$.
2. Else let $n = n_s = n_{s-1} + 1$, and choose a rational $q_n > 0$ such that, where $C_{n_s} = C_{n_{s-1}} + q_n B_s$, $C_{n_s}(v_{s-1}) < 2$.
3. Now if $v = v_{s-1}$, (a*) holds (in the new case $j = n$, we use that case 1. did not apply). So (b*) fails. Thus we may choose $w = v_s \succeq v_{s-1}$ so that $N(w) \geq 2^s$ and $C_n(w) < 2$.

No B_s succeeds on Z, since either $B_s(w)$ is undefined for $v_s \preceq w \preceq Z$, or $B_s(v_t) < 2/q_n$ for all $t \geq s$, where q_n is the rational chosen at stage s. In the latter case B_s is bounded along Z. □

References

[1] Klaus Ambos-Spies and Antonin Kucera. Randomness in Computability Theory. P. Cholak ea, eds. Proceedings of a 1999 AMS conference, Contemporary Math 257: 1-14, AMS, 2000
[2] A. Nies. Lowness properties and randomness. *Adv. in Math.*, 197:274–305, 2005.
[3] A. Nies. *Computability and Randomness.* Oxford University Press, 2009. Oxford Logic Guides, xv + 443 pages.
[4] Sebastiaan A. Terwijn and Domenico Zambella. Computational Randomness and Lowness. The Journal of Symbolic Logic, 66(3):1188-1205, 2001.

Department of Computer Science, University of Auckland,, Web site http://www.cs.auckland.ac.nz/~nies
E-mail address: `andre@cs.auckland.ac.nz`