Algorithmic Randomness with Probabilistic Strategies and Intermediate Success

Preliminary draft, comments appreciated

Sam Buss*
Department of Mathematics
University of California, San Diego
La Jolla, CA 92093-0112, USA
sbuss@math.ucsd.edu

Mia Minnes†
Department of Mathematics
University of California, San Diego
La Jolla, CA 92093-0112, USA
minnes@math.ucsd.edu

July 2, 2013

Abstract

Probabilistic betting strategies provide a method of characterizing algorithmically random sequences, including Martin-Löf random sequences and (partial) computably random sequences. We generalize probabilistic betting strategies to allow probabilistic moves at every step, not just at betting steps; we also consider restricting randomness to binary choices (or, "coin flips"). We prove that these modifications do not change the strength of probabilistic betting strategies. We introduce probabilistic strategies with intermediate success probabilities, and prove some simulation results. We give a new proof of the separation of Martin-Löf randomness and partial computable randomness using probabilistic betting strategies.

1 Introduction

An algorithmically random real is a binary sequence which appears random to any computable process. The central notions of algorithmic randomness can be defined in several equivalent ways by using Martin-Löf tests, Kolmogorov compressibility, or algorithmic martingales. (See [4, 7] for background on algorithmic randomness.) The authors recently gave new char-

*Supported in part by NSF grants DMS-1101228 and CCR-1213151.
†Supported in part by NSF grant DMS-1060351.

acterizations of algorithmic randomness using probabilistic betting strategies [3]. A probabilistic betting strategy works similarly to a martingale in that the algorithm places bets on the bits of an infinite sequence X. However, a probabilistic betting strategy can make a randomized choice each time it is about to bet by computing a rational probability that controls whether the bet is actually placed or not. Thus, probabilistic betting strategies differ essentially from martingales, since they are (probabilistic) algorithms and do not compute fixed functions.

A probabilistic betting strategy is deemed "successful" against X provided it accumulates unbounded capital when betting against X. There are two distinct ways of defining "success". First, a probabilistic betting strategy is "Ex-successful" provided it accumulates unbounded capital in *expectation*, where the expectation is computed in terms of expected capital after n bets. Second, a probabilistic betting strategy is "P1-successful" provided it accumulates unbounded capital with probability one. (Precise definitions are given in Section 2.) Our earlier paper [3] proved that Ex-randomness characterizes Martin-Löf randomness. It was also proved that P1-randomness characterizes computable randomness or partial computable randomness, depending on whether or not the probabilistic betting strategy always continues betting with probability one.

The probabilistic betting strategies of [3] are constrained to used randomization only for the purpose of deciding whether to place a bet or to wait to place a (possibly different) bet later. In Section 3 below, we introduce a more general notion of probabilistic strategies, called "R-strategies", which are allowed to use randomization at any point in the computation. In other words, an R-strategy can make a probabilistic choice without having to risk making a bet right away. These R-strategies are seen to be equivalent in strength to probabilistic betting strategies but have the advantage of being easier to design and describe than the probabilistic betting strategies of [3]. We also show that, with no loss in computational power, R-strategies may be restricted to dyadic randomness, namely, restricted to making probabilistic decisions among two *equally likely* choices.

Section 4 defines a Pα-strategy (for fixed $0 < \alpha \leq 1$) against a sequence X to be an (R-)strategy which accumulates unbounded capital with probability at least α. We prove that Pα-strategies are equivalent to Pβ-strategies, for all $0 < \alpha < \beta < 1$. A strategy is "locally weak" for X if it always continues to bet against X with probability one. We show that there exists a locally weak Pα-strategy against X iff there is a P1-strategy against X. By [3], this is equivalent to X being computably random.

Section 5 proves that every Martin-Lof random sequence is Pα-random,

for all $\alpha \in (0,1]$. The question of whether every $\mathsf{P}\alpha$-random sequence is Martin-Lof random is still open. Section 6 concludes the paper by giving a new exposition of the proof that Martin-Löf randomness is stronger than partial computable randomness. This new proof uses the characterization of Martin-Löf randomness in terms of Ex-R-strategies, and gives an example of how the use of R-strategies can streamline proofs in algorithmic randomness.

2 Background

Definition Let Σ be a finite set of symbols. The set of finite strings over Σ is denoted Σ^*; the set of (one-way) infinite strings is Σ^∞. The length of a finite string σ is denoted by $|\sigma|$. We write λ for the empty string. The number of occurrences of a symbol a in σ is denoted $|\sigma|_a$. Given $\sigma \in \Sigma^*$ and $x \in \Sigma^* \cup \Sigma^\infty$, we write $\sigma \sqsubset x$ if σ is a proper initial prefix of x. The set of infinite extensions of $\sigma \in \Sigma^*$ is denoted by $[\sigma] \subseteq \Sigma^\infty$. Concatenation of two strings is represented by writing them one next to another. Given $X \in \Sigma^\infty$, its restriction to the first n letters is denoted $X\restriction n$. The i-th symbol of X is denoted $X(i)$.

For the purposes of studying random strings, we restrict to infinite binary strings $X \in \{0,1\}^\infty$. We now recall the definitions used in characterizing algorithmic randomness via betting strategies or martingales.

Definition A function $d : \{0,1\}^* \to \mathbb{R}^{\geq 0}$ is a *martingale* if, for each $\sigma \in \{0,1\}^*$,

$$d(\sigma) = \frac{d(\sigma 0) + d(\sigma 1)}{2}. \tag{1}$$

The martingale d *succeeds* on $X \in \{0,1\}^\infty$ if $\limsup_n d(X\restriction n) = \infty$ [1].

A martingale can be viewed as equaling the capital of a betting strategy as it bets on successive bits of an infinite string. The martingale property (1) ensures that bets are fair.

The motivation behind many definitions of algorithmic randomness is that a random object is unpredictable and, therefore, an effective betting strategy should not succeed against it. Varying the effectiveness requirement leads to different notions of randomness. Hence, we recall some basic definitions of computable real-valued functions.

[1] It can be shown that for all the notions of algorithmic randomness discussed below, the "lim sup" may be replaced by "lim".

Definition Let D be a computable set. A function $f : D \to \mathbb{Q}^{\geq 0}$ is *computable* if there is a Turing machine which, on input $x \in D$, halts in finite time and outputs $f(x)$. The function $f : D \to \mathbb{R}^{\geq 0}$ is *computable* if there is a rational-valued computable function $g : D \times \mathbb{Q}^{>0} \to \mathbb{Q}^{\geq 0}$ such that, for all $x \in D$ and $\epsilon > 0$, $|f(x) - g(x, \epsilon)| < \epsilon$. A function $f : D \to \mathbb{R}^{\geq 0}$ is *computably enumerable* (or c.e.) if there is a rational-valued computable function $h : D \times \mathbb{N} \to \mathbb{Q}^{\geq 0}$ such that, for all $x \in D$, $\lim_i h(x, i) = f(x)$ and for all $i \in \mathbb{N}$, $h(x, i) < h(x, i+1) < f(x)$.

Definition A function $f : D \to \mathbb{R}^{\geq 0}$ is *partial* if its domain is a subset of D. We write $f(x)\downarrow$, resp. $f(x)\uparrow$, if $x \in D$ is, resp. is not, in the domain of f. A partial function f is a *partial martingale* provided that whenever either $f(\sigma 0)\downarrow$ or $f(\sigma 1)\downarrow$, then all three of $f(\sigma)$, $f(\sigma 0)$, and $f(\sigma 1)$ are defined and the martingale condition (1) holds.

Definition An infinite string $X \in \{0,1\}^\infty$ is Martin-Löf random, abbreviated as *ML-random*, if no c.e. martingale succeeds on it. It is *partial computably random* if no partial computable martingale succeeds on it. And, X is *computably random* if no computable martingale succeeds on it.

The relationships between these notions of randomness are well-known.

Theorem 1 [1, 6, 8] *Every ML-random is partial computably random; every partial computably random is computably random. Moreover, the inclusions are strict.*

In [3], the authors introduced probabilistic strategies and showed that they characterize each of these classes of randomness. The rest of this section briefly reviews the definitions and results from [3].

A probabilistic betting strategy A is an algorithm which, at each step of its computation, makes a probabilistic choice to either bet (b) on the next symbol of X, or wait (w) and not bet on that symbol yet. The history of bet/wait choices so far is coded by a string $\pi \in \{\mathsf{b}, \mathsf{w}\}^*$, and the history of the bet-upon bits of X is coded by a $\sigma \in \{0,1\}^*$. If $|\sigma| = |\pi|_\mathsf{b}$, then the pair (π, σ) is called a *valid A-configuration*, and fully specifies the history of A's computation after $|\pi|$ steps and $|\pi|_\mathsf{b}$ bets.

Definition [3] A *probabilistic betting strategy* A is specified by a pair of (total) computable functions $p_A : \{\mathsf{b}, \mathsf{w}\}^* \times \{0,1\}^* \to \mathbb{Q} \cap [0,1]$ and $q_A : \{\mathsf{b}, \mathsf{w}\}^* \times \{0,1\}^* \to \mathbb{Q} \cap [0,2]$.

If (π, σ) is a valid A-configuration, $p_A(\pi, \sigma)$ specifies the probability of A placing a bet. In the event a bet is placed, $q_A(\pi, \sigma)$ specifies the stake function value for the bet.

The success of a probabilistic betting strategy depends on the probabilistic choices that are made. For this, there are two inductively defined auxiliary functions. The notation $\mathrm{Prob}[\,\cdot \mid \cdot\,]$ denotes conditional probability.

Definition The *cumulative probability function* P_A is defined as

$$P_A(\pi, \sigma) \;=\; \mathrm{Prob}[\pi \text{ codes the initial bet/wait moves of } A \mid \sigma \sqsubset X]$$

for (π, σ) a valid A-configuration and $X \in \{0,1\}^\infty$ the infinite binary string being played against. Thus, $P_A(\lambda, \lambda) = 1$, $P_A(\pi\mathsf{b}, \sigma x) = P_A(\pi, \sigma)p_A(\pi, \sigma)$, and $P_A(\pi\mathsf{w}, \sigma) = P_A(\pi, \sigma)(1 - p_A(\pi, \sigma))$ where $x \in \{0,1\}$. The *capital function* C_A is defined as $C_A(\lambda, \lambda) = 1$, $C_A(\pi\mathsf{w}, \sigma) = C_A(\pi, \sigma)$, and

$$\begin{aligned} C_A(\pi\mathsf{b}, \sigma 0) &= C_A(\pi, \sigma)(2 - q_A(\pi, \sigma)) \\ C_A(\pi\mathsf{b}, \sigma 1) &= C_A(\pi, \sigma)q_A(\pi, \sigma) \end{aligned}$$

for (π, σ) a valid A-configuration. We let $P_A^X(\pi)$ and $C_A^X(\pi)$ abbreviate $P_A(\pi, X\restriction|\pi|_\mathsf{b})$ and $C_A(\pi, X\restriction|\pi|_\mathsf{b})$, respectively.

Definition The probability measure μ_A^X induced on $\{\mathsf{b}, \mathsf{w}\}^\infty$ by the strategy A playing against an infinite string $X \in \{0,1\}^\infty$ is defined on basic open sets by

$$\mu_A^X([\pi]) = P_A^X(\pi).$$

A probabilistic strategy succeeds when playing against $X \in \{0,1\}^\infty$ along a fixed infinite path $\Pi \in \{\mathsf{b}, \mathsf{w}\}^\infty$ provided it accumulates unbounded capital during its play. The measure of the paths along which A succeeds when playing against X leads to one definition of success for probabilistic strategies:

Definition [3] The *probability of success* of a probabilistic betting strategy A when playing against $X \in \{0,1\}^\infty$ is

$$Pr_A^X(Succ) \;=\; \mu_A^X\left(\{\Pi \in \{\mathsf{b}, \mathsf{w}\}^\infty : \lim_n C_A^X(\Pi\restriction n) = \infty\}\right).$$

Then, A is a $\mathsf{P1}$-*strategy* for X if $Pr_A^X(Succ) = 1$. Replacing the "lim" with "lim sup" gives the (equivalent) notion of a *limsup* $\mathsf{P1}$-*strategy* for X.

Definition [3] A probabilistic betting strategy A *eventually bets on* X *with probability one* provided that for all $\pi \in \{\mathsf{b}, \mathsf{w}\}^*$

$$P_A^X(\pi) \cdot \prod_{i \in \mathbb{N}} (1 - p_A^X(\pi \mathsf{w}^i)) \;=\; 0.$$

In this case, A is called *locally weak* with respect to X. If A is locally weak for all sequences X, then we say that A is *weak* or that A *always eventually bets with probability one.*

An alternate notion of success for a probabilistic betting strategy considers the expected capital of the strategy after n bets have been placed:

Definition We denote by $R(n)$ the set of possible b/w histories up through the n-th bet. That is, for $n \in \mathbb{N}$, define $R(0) = \{\lambda\}$ and $R(n+1) = \{\pi \mathsf{b} \in \{\mathsf{b}, \mathsf{w}\}^* : |\pi|_{\mathsf{b}} = n\}$.

If A is a probabilistic betting strategy, $X \in \{0,1\}^\infty$, and $n \in \mathbb{N}$, then

$$\mathsf{Ex}_A^X(n) = \sum_{\pi \in R(n)} P_A^X(\pi) C_A^X(\pi).$$

This is exactly the expected value for the capital of A after n bets if we define the value of "the capital of A after n bets" to equal zero in the event that A never makes n bets.

We say that A is an Ex-*strategy* for X if $\lim_n \mathsf{Ex}_A^X(n) = \infty$. As before, we can replace "lim" by "lim sup": A is a *limsup* Ex-*strategy* for X if $\limsup_n \mathsf{Ex}_A^X(n) = \infty$.

Lemma 2 [3] *Let $X \in \{0,1\}^\infty$. There is a $\mathsf{P1}$-strategy (resp. weak $\mathsf{P1}$-strategy) for X if and only if there is a limsup $\mathsf{P1}$-strategy (resp. weak limsup $\mathsf{P1}$strategy) for X. Similarly, there is an Ex-strategy for X if and only if there is a limsup Ex-strategy for X.*

Definition [3] An infinite string $X \in \{0,1\}^\infty$ is $\mathsf{P1}$-*random* (resp. *weak* $\mathsf{P1}$-*random*) if no probabilistic betting strategy is a $\mathsf{P1}$-strategy (resp. weak $\mathsf{P1}$-strategy) for X. And, X is Ex-*random* if no probabilistic betting strategy is an Ex-strategy for it.

Theorem 3 [3] *An infinite string is ML-random if and only if it is Ex-random. It is partial computably random if and only if it is $\mathsf{P1}$-random. It is computably random if and only if it is weak $\mathsf{P1}$-random.*

It was also shown in [3] that X is partial computably random (resp. computably random) if and only there is a locally weak (resp. weak) Ex-strategy for X.

3 Equivalent notions of strategies

The definition of probabilistic betting strategy in [3] is based on special probabilistic algorithms in which randomness is used only to decide whether or not to place a bet. The complexity theory literature, on the other hand, uses a more general and more convenient model of probabilistic computation, given by probabilistic Turing machines that allow randomness at every step of the computation.

Definition (Def. 7.1 in [2]) A *probabilistic Turing machine* (PTM) is a Turing machine with two transition functions δ_0 and δ_1. Each computation step of a PTM randomly chooses to use either δ_0 or δ_1; selecting each δ_i with probability $\frac{1}{2}$. The choice of transition function at each step is made independently of all previous choices.

We define alternate versions of probabilistic betting strategies called "R-strategies" (for "random strategies") which use the PTM model of computation:

Definition An R-*strategy with rational randomness* B is specified by a computable *stake* function $q_B : \{0,1\}^* \times \{0,1\}^* \to \{\mathsf{w}\} \cup (\mathbb{Q} \cap [0,2])$ and a computable *bias* function $p_B : \{0,1\}^* \times \{0,1\}^* \to \mathbb{Q} \cap [0,1]$. As described below, if B so far has bet on bits $\sigma \in \{0,1\}^*$ and has randomly selected bits $r \in \{0,1\}^*$, then $q_B(\sigma, r)$ controls whether and how much B bets next, and $p_B(\sigma, r)$ controls the bias in selecting the next random bit.

If the bias function is the constant function $p_B(\sigma, r) = \frac{1}{2}$, then we call B an R-*strategy with dyadic randomness*.

Suppose that B has carried out its strategy for n steps, working against $X \in \{0,1\}^*$. Let σ be the sequence of bits of X that have been bet upon so far, and thus are known to B. Also let the string r be the sequence of random bits selected so far; since exactly one random bit is chosen per step, $|r| = n$. We interpret $q_B(\sigma, r) = \mathsf{w}$ to mean that B does not make a bet at this stage. Otherwise, $q_B(\sigma, r) \in \mathbb{Q} \cap [0,2]$, and B bets with a stake value of $q_B(\sigma, r)$. We interpret the value $p_B(\sigma, r)$ as the probability that the next selected random bit is equal to 1.

The following definitions formalize the action of an R-strategy. First we define when σ and r code a feasible valid configuration for B.

Definition Let B be an R-strategy. The set of *valid* B*-configurations* is inductively defined by

a. (λ, λ) is a valid B-configuration.

b. If (σ, r) is a valid B-configuration, $q_B(\sigma, r) = \mathsf{w}$, and $y \in \{0,1\}$, then (σ, ry) is a valid B-configuration.

c. If (σ, r) is a valid B-configuration, $q_B(\sigma, r) \neq \mathsf{w}$, and $x, y \in \{0,1\}$, then $(\sigma x, ry)$ is a valid B-configuration.

The intuition is that y is the next random bit, and x is the bet-upon bit of X.

When B is working against X, and (σ, r) is a valid B-configuration with $\sigma \sqsubset X$, we define $P_B(\sigma, r)$ to be the probability that B reaches configuration (σ, r), and $C_B(\sigma, r)$ to be B's capital upon reaching (σ, r):

Definition Let B be an R-strategy. The probability function P_B and the capital value function C_B are defined as follows. Initially, $P_B(\lambda, \lambda) = 1$ and $C_B(\lambda, \lambda) = 1$. Suppose that (σ, r) and $(\sigma x, ry)$ are both valid B-configurations, with $y \in \{0,1\}$, and $x \in \{0, 1, \lambda\}$. Then

$$P_B(\sigma x, ry) = \begin{cases} (1 - p_B(\sigma, r)) \cdot P_B(\sigma, r) & \text{if } y = 0 \\ p_B(\sigma, r) \cdot P_B(\sigma, r) & \text{if } y = 1 \end{cases}$$

$$C_B(\sigma x, ry) = \begin{cases} C_B(\sigma, r) & \text{if } x = \lambda \\ C_B(\sigma, r) \cdot (2 - q_B(\sigma, r)) & \text{if } x = 0 \\ C_B(\sigma, r) \cdot q_B(\sigma, r) & \text{if } x = 1 \end{cases}$$

Note that $x = \lambda$ iff $q_B(\sigma, r) = \mathsf{w}$, so the definition is well-formed.

Definition Let $r \in \{0,1\}^n$. We write $P_B^X(r)$ and $C_B^X(r)$ for the probability and capital values associated with the random bits r when playing against X for $n = |r|$ steps. Namely, let $\sigma \sqsubset X$ be the unique initial substring of X such that (σ, r) is a valid B-configuration. Then $P_B^X(r) = P_B(\sigma, r)$ and $C_B^X(r) = C_B(\sigma, r)$. We define $p_B^X(r)$ and $q_B^X(r)$ similarly.

The measure μ_B^X is defined as before so that $\mu_B^X([r]) = P_B^X(r)$. Note that if B uses dyadic randomness, then μ_B^X is just the usual (Lebesgue) measure μ on Cantor space, with $\mu([r]) = 2^{-|r|}$.

Definition The *probability of success* of an R-strategy B when playing against $X \in \{0,1\}^\infty$ is

$$Pr_B^X(Succ) = \mu_B^X\left(\{R \in \{0,1\}^\infty : \lim_n C_B^X(R\restriction n) = \infty\}\right).$$

B is said to be P1-*successful* against X provided $Pr_B^X(Succ) = 1$. The sequence X is P1-R-*random* if no R-strategy is P1-successful against X.

The *limsup probability of success*, *limsup* $\Pr_B^X(Succ)$, and the notion of *limsup* P1-*successful* are defined similarly with "lim sup" in place of "lim".

We also need to define the expected capital after n bets, $\mathsf{Ex}_B^X(n)$. For probabilistic strategies, this definition was made using the set $R(n)$. For R-strategies, it is a little more complicated as the random bits r do not by themselves describe how many bets have been placed. Instead, define $R_{B,X}(n)$ to be the set of valid B-configurations (σ, r) such that $\sigma \sqsubset X$ and $|\sigma| = n$ and such that B has placed a bet (its n-th bet) during the step leading to the configuation (σ, r).

Definition The *expected capital after n bets* of an R-strategy B when working against $X \in \{0,1\}^\infty$ is defined as

$$\mathsf{Ex}_B^X(n) \;=\; \sum_{(\sigma,r)\in R_{B,X}(n)} P_B(\sigma,r)C_B(\sigma,r) \;=\; \sum_{(\sigma,r)\in R_{B,X}(n)} P_B^X(r)C_B^X(r).$$

B is said to be Ex-*successful* against X provided $\lim_n \mathsf{Ex}_B^X(n) = \infty$. The sequence X is Ex-R-*random* if no R-strategy is Ex-successful against X.

The notion of *limsup* Ex-*successful* is defined similarly with "lim sup" in place of "lim".

Definition An R-strategy B *eventually bets on X with probability one* if

$$\mathrm{Prob}[\forall m\, \exists n{>}m\ \left(q_B^X(R\restriction n) \neq \mathsf{w}\right)] \;=\; 1.$$

with the probability taken over all infinite binary strings R under the probability measure μ_B^X. As before, we equivalently say that B is *locally weak against X*.

Definition An R-strategy is *weak* if it is locally weak against all X. The sequence X is *weak*-P1-R-*random* if there is no weak R-strategy which is P1-successful against X.

The above definitions of R-strategies, like the definitions of probabilistic betting strategies in [3], are "stateless". That is to say, although based on the intuition of a probabilistic Turing machine carrying out the R-strategy, the formal definition of R-strategy does not include any notion of the current state and tape contents of a Turing machine. The above definitions can be modified to define "stateful" R-strategies. For this, a configuration of a

stateful R-strategy would be a triple (σ, r, s) where σ and r are as before, and s is intended to encode the state information. For example, in the Turing machine model, s would encode the Turing machine's finite control state plus its tape contents and tape head positions. Formally, a stateful R-strategy is defined by letting the computable functions $p_B = p_B(\sigma, r, s)$ and $q_B = q_B(\sigma, r, s)$ now take the current state as an additional input, and adding a new computable function $Next_B = Next_B(\sigma, r, s)$ which computes the state value s for the next configuration of B. The rest of the definitions are easily modified to incorporate the presence of the state value s. The notion of stateful probabilistic betting strategy can be defined by modifying the definitions of [3] analogously.

It is easy to see that stateful strategies and stateless strategies have the same computational power: this follows from the fact we are working only with computable strategies, and not with resource-bounded algorithms. Indeed, any stateful strategy can be converted into an equivalent stateless strategy by the simple expedient of having the stateless strategy recompute all the state values s of the stateful strategy as necessary.

For the present paper, the advantage of using stateful strategies is that it makes it easier to informally describe strategies. Generally when we are proving that a certain strategy exists, it is convenient describe the strategy as a stateful strategy. Conversely, when proving limits on a strategy, or when simulating a strategy, it can be easier to assume the strategy is stateless. We expect it could also be useful to work with stateful strategies when using resource-bounded strategies (requiring, say, time- or space-bounded computation), but we do not consider resource-bounded computation in the present paper.

Definition Two betting (R-)strategies A and B are *strongly equivalent* if for all $X \in \{0,1\}^\infty$,

- $Pr_A^X(Succ) = Pr_B^X(Succ)$,
- $\mathsf{Ex}_A^X(n) = \mathsf{Ex}_B^X(n)$ for all $n \geq 0$.
- A eventually bets with probability 1 on X iff B eventually bets with probability 1 on X.

For example, it is not difficult to show that every stateful R-strategy A is strongly equivalent to a stateless R-strategy B. The next theorem states that R-strategies and probabilistic betting strategies are strongly equivalent. [2]

[2]It is possible to define an even stronger notion of equivalence, by stating that A and B are "very strongly equivalent" provided there is a measure-preserving correspondence be-

Theorem 4 *If A is a strategy of one of the following three kinds, then there are strategies of each of the other two kinds which are strongly equivalent to A.*

- R-*strategies with dyadic randomness.*
- R-*strategies with rational randomness.*
- *Probabilistic betting strategies.*

Theorem 4 allows us to work with R-strategies with rational or dyadic randomness, or with probabilistic betting strategies, choosing whichever one is most suitable for the task at hand. It also immediately implies that P1-R-randomness, weak-P1-R-randomness, and Ex-R-randomness are equivalent, respectively, to P1-randomness, weak-P1-randomness, and Ex-randomness.

Proof (Sketch.) Of course, an R-strategy with dyadic randomness is also an R-strategy with rational randomness. Conversely, suppose A is an R-strategy with rational randomness. We claim there is a strongly equivalent R-strategy B with dyadic randomness. The (stateful) strategy B simulates A directly: at each step where A chooses a random bit y with bias $p_A(\sigma, r) \neq \mathsf{w}$, B begins enumerating the base-two representation of $p_A(\sigma, r)$ as $0.x_1x_2x_3\ldots$, with each $x_i \in \{0, 1\}$. At the same time, B chooses random bits $z_1, z_2, z_3, \ldots$ uniformly at random (with dyadic probabilities). As soon as B finds an i such that $x_i \neq z_i$, B halts the enumerations and sets the random bit y equal to 1 if $z_i < x_i$ or equal to 0 if $z_i > x_i$. Then, B forgets the values z_i and resumes the simulation of A. Note that B will find some $z_i \neq x_i$ with probability 1. It is easy to verify that B faithfully simulates A.

Now we consider strong equivalences between probabilistic betting strategies and R-strategies. First, a probabilistic betting strategy A is clearly strongly equivalent to some R-strategy B. This is because A is restricted to using probabilities only when (possibly) placing a bet, where B may use randomness at every step. Second, let A be an R-strategy with dyadic randomness. (The argument would be very similar if A used rational probabilities.) We describe a stateful probabilistic strategy B which is strongly equivalent to A.

To describe the operation of B, we first describe how it decides to place its first bet, that is, what it does to bet on bit $X(0)$. For this, B enumerates

tween computations of A and B such that the corresponding computations of A and B make exactly the same bets with the same stake values. Theorem 4 holds also for "very strongly equivalent". The notion of "very strongly equivalent" is not needed for the results of the present paper however.

all bit strings $r \in \{0,1\}^*$, say in length-lexicographic order, such that (λ, r) is a valid A-configuration and such that $q_A(\lambda, r) \neq \mathsf{w}$. Let $r_1, r_2, r_3, \ldots$ denote the enumerated values of r. Note there may be either finitely or infinitely many r_i's. The r_i's are interpreted as the random bits chosen by A before placing the first bet on X: the condition $q_A(\lambda, r) \neq \mathsf{w}$ means that (λ, r) is a configuration where A places its first bet with stake value $q_A(\lambda, r)$. When the first value, r_1, is enumerated, B bets on $X(0)$ with probability $p_1 = 2^{-|r_1|}$ using the same stake value $q_A(\lambda, r_1)$ as A: the probability p_1 is just the probability that the random bits r_1 are chosen by A. If B does not bet, then it resumes enumerating the r_i's. For each subsequent r_i in the enumeration, B bets with probability

$$p_i \;=\; \frac{2^{-|r_i|}}{1 - \sum_{j<i} 2^{-|r_j|}}. \tag{2}$$

using the stake value $q_A(\lambda, r_i)$. It is easy to check that each bet of B corresponds faithfully to a bet of A. Furthermore, B bets on $X(0)$ with probability 1 iff A does. If and when B does place a bet based on a pair (λ, r_i), then B defines the strings $\sigma_1 := X(0)$ and $\overline{r}_1 := r_i$ to record the now-known bit of X and the random bits chosen by A so far.

The strategy B uses an analogous algorithm to decide how to bet on $X(m{+}1)$, the $(m+1)$-st bit of X. At this point, B has already set $\sigma_m \in \{0,1\}^m$ (the first m bits of X) and $\overline{r}_m$ (the random bits chosen by A so far). B now enumerates all values $r_1, r_2, r_3, \ldots$, such that $r_i \in \{0,1\}^*$ and such that $(\sigma_m, \overline{r}_m r_i)$ is a valid A-configuration with $q_A(\sigma_m, \overline{r}_m r_i) \neq \mathsf{w}$. For each r_i, B places a bet on $X(m+1)$ with probability p_i as given by (2) and with stake value $q_A(\sigma_m, \overline{r}_m r_i)$. Once a bet is placed, the values σ_{m+1} and $\overline{r}_{m+1}$ are set equal to $\sigma_m X(m+1)$ and $\overline{r}_m r_i$, respectively. We leave to the reader to verify that this a faithful simulation of A by B, and that B is strongly equivalent to A. □

Corollary 5 *There is a locally weak* P1*-strategy (resp.,* Ex*-strategy) against X iff there is a locally weak* P1-R*-strategy (resp.,* Ex-R*-strategy) against X.*

For the proof of Theorem 10 in the next section, it will be convenient to assume that R-strategies never use the stake value 0 or 2; so that the capital never falls to zero. This is shown by the next proposition. We cannot use strong equivalence for this however, since the stake values will change and this affects the values of $\mathsf{Ex}_A^X(n)$, which can change by as much as a constant factor. Instead, we rely on a weaker notion of "acceptance equivalence".

Definition Two R-strategies A and B are P1-*acceptance equivalent* provided that for all $X \in \{0,1\}^\infty$, A is P1-successful against X iff B is P1-successful against X. A similar definition is used for Ex-*acceptance equivalent*. In addition, A and B are *acceptance equivalent* provided they are both P1-acceptance equivalent and Ex-acceptance equivalent.

We make several observations: Any strongly equivalent R-strategies are also P1-acceptance equivalent, Ex-acceptance equivalent, and acceptance equivalent. Also, if two R-strategies are, in fact, deterministic, then they either succeed or fail on each infinite string, without any probabilistic considerations. Hence, they are P1-acceptance equivalent if and only if they are Ex-acceptance equivalent. However, there are P1-acceptance equivalent R-strategies that are not Ex-acceptance equivalent, and vice versa. For example, two R-strategies which have a nonzero probability of never betting would be P1-acceptance-equivalent (because they are nowhere P1-successful) but may not be Ex-acceptance-equivalent. Conversely, let A be an R-strategy which is P1-successful on some string X. Modify A to obtain a new R-strategy B by adding an initial step which, with some small nonzero probability, enters a loop and never places any bet. The strategy B is Ex-successful on exactly those strings for which A Ex-succeeds, but B is nowhere P1-successful. Thus A and B are Ex-acceptance equivalent, but not P1-acceptance equivalent.

Lemma 6 *Every* R*-strategy* B *is acceptance equivalent to an* R*-strategy* B' *such that* 0 *and* 2 *are not in the range of* $q_{B'}$.

Proof Informally, the strategy B' copies the strategy B except that any bets B makes with stake values 0 and 2 are changed by B' to bets that differ from 0 and 2 by increasingly small amounts. That is, consider the computable sequence $\epsilon_i = 1/2^{i+2}$, which has the property that $1/2 < \prod_{i=0}^{\infty}(1-\epsilon_i) < 1$. Set $p_{B'}(\sigma, r) = p_B(\sigma, r)$ and define $q_{B'}(\sigma, r)$ to equal

$$q_{B'}(\sigma,r) \;=\; \begin{cases} 2-\epsilon_{|r|} & \text{if } q_B(\sigma,r) > 2-\epsilon_{|r|} \\ \epsilon_{|r|} & \text{if } q_B(\sigma,r) < \epsilon_{|r|} \\ q_B(\sigma,r) & \text{otherwise.} \end{cases}$$

It is easy to verify that B' is acceptance equivalent to B. □

We end this section with a couple of simple observations about R-strategies. First, we note that there is no benefit to be obtained by letting the functions p_B and q_B be partial computable:

Definition A *partial* R*-strategy* B is defined similarly to an R*-strategy* except that p_A and q_A are allowed to be partial computable instead of (total) computable.

Lemma 7 *Every partial computable* R*-strategy* A *is strongly equivalent to a (total)* R*-strategy* B*.*

Proof (Sketch.) The idea is to insert extra wait states into the computations of $p_A(\sigma, r)$ and $q_A(\sigma, r)$, say by fixing a fixed constant $T > 0$, and interrupting the computations of $p_A(\sigma, r)$ and $q_A(\sigma, r)$ every T steps and waiting (w) without betting. This allows B to have total functions p_B and q_B. The extra random bits obtained by B at these interruptions are just ignored. □

Finally, the next theorem states in effect that randomness does not increase the power of total computable martingales.

Theorem 8 *Let* B *be an* R*-strategy such that* $q_B(\sigma, r) \neq \mathsf{w}$ *for all* (σ, r)*. Then* B *is* Ex*-acceptance equivalent to a probabilistic betting strategy* A *which always bets, that is, such that* $p_A(\pi, \sigma) = 1$ *for all* (π, σ)*.*

The condition that $p_A(\pi, \sigma)$ always equals 1 means that A is actually deterministic, not probabilistic.

Proof (Sketch.) Since A always bets and never waits, $p_A(\pi, \sigma)$ and $q_A(\pi, \sigma)$ need to be defined only for $\pi = \mathsf{b}^{|\sigma|}$. The idea of the proof is to let

$$q_A(\mathsf{b}^n, \sigma) \;=\; \frac{\sum_{r\in\{0,1\}^n} P_B(\sigma, r) C_B(\sigma, r) q_B(\sigma, r)}{\sum_{r\in\{0,1\}^n} P_B(\sigma, r) C_B(\sigma, r)}, \tag{3}$$

where $n = |\sigma|$. Note that w.l.o.g. by Lemma 6, the values $C_B(\sigma, r)$ are non-zero, so $q_A(\mathsf{b}^n, \sigma)$ is well-defined and equal to a weighted average of the values $q_B(\sigma, r)$. Under this definition, induction on n establishes that if $\sigma \sqsubset X$ then $C_A(\mathsf{b}^n, \sigma)$ is equal to $\mathsf{Ex}_B^X(n)$, namely the denominator of (3). The details of the proof are left to the reader. □

4 Intermediate success probabilities

A natural alternative criterion for the success of a probabilistic betting strategy is that the set of successful paths has positive measure, instead of measure one.

Definition Let $0 < \alpha \leq 1$. A probabilistic betting strategy A is a $\mathsf{P}\alpha$-*strategy against* X if A's probability of success on input $X \in \{0,1\}^\infty$ is at least α, namely if

$$Pr_A^X(Succ) \;=\; \mu_A^X\big(\{\Pi \in \{\mathsf{b},\mathsf{w}\}^\infty : \lim_n C_A^X(\Pi{\restriction}n) = \infty\}\big) \;\geq\; \alpha. \qquad (4)$$

The string X is $\mathsf{P}\alpha$-*random* if there is no $\mathsf{P}\alpha$-strategy against it. The notions of *limsup* $\mathsf{P}\alpha$-*strategy* and *limsup* $\mathsf{P}\alpha$-*random* are defined similarly by replacing "lim" with "lim sup" in (4).

The techniques from [3] carry over to prove that X is $\mathsf{P}\alpha$-random iff X is limsup $\mathsf{P}\alpha$-random. The key tool is the well-known "slow-but-sure" savings trick. We discuss and extend this trick in the proof of Theorem 12 below.

$\mathsf{P}\alpha$-strategies can also be formulated in terms of R-strategies.

Definition An R-strategy B is a $\mathsf{P}\alpha$-R-*strategy against* X if its probability of success on input $X \in \{0,1\}^\infty$ is at least α, namely if

$$Pr_B^X(Succ) \;=\; \mu_B^X\big(\{R \in \{0,1\}^\infty : \lim_n C_B^X(R{\restriction}n) = \infty\}\big) \;\geq\; \alpha. \qquad (5)$$

The string X is $\mathsf{P}\alpha$-R-*random* if there is no $\mathsf{P}\alpha$-R-strategy against it. The notions of *limsup* $\mathsf{P}\alpha$-R-*strategy* and *limsup* $\mathsf{P}\alpha$-R-*random* are defined similarly.

Lemma 9 *A sequence $X \in \{0,1\}^\infty$ is (limsup) $\mathsf{P}\alpha$-random if and only if it is (limsup) $\mathsf{P}\alpha$-R-random.*

Lemma 9 follows by the proof method of Theorem 4.

We next establish our first main result about $\mathsf{P}\alpha$-randomness; namely, that its definition is independent of the choice of the value $\alpha \in (0,1)$. For this we show that the measure of successful paths can be amplified arbitrarily close to 1.

Theorem 10 *Let $X \in \{0,1\}^\infty$ and let $\beta > 0$. If X is not $\mathsf{P}\beta$-random then, for each $0 < \alpha < 1$, X is not $\mathsf{P}\alpha$-random. Hence, for $\alpha, \beta \in (0,1)$, $\mathsf{P}\alpha$-randomness and $\mathsf{P}\beta$-randomness are equivalent.*

Proof Suppose A is a limsup $\mathsf{P}\alpha$-R-strategy against X that uses dyadic randomness. Let $\alpha < \beta < 1$. Without loss of generality, the limsup probability of success of A, as given by (5) with "lim sup" in place of "lim", is equal to exactly α.

We construct a $\mathsf{P}\beta$-R-strategy B against X. Let $\epsilon = 1 - \beta$. Fix rational numbers q_1, q_2 such that $0 < q_1 < \alpha < q_2 < 1$ and $q_2 < (1+\epsilon)q_1$. Let C_0 be a rational number such that

$$\text{Prob}[\exists n \left(C_A^X(R\restriction n) > C_0\right)] < q_2, \tag{6}$$

where the probability is taken over all choices of $R \in \{0,1\}^\infty$. Such a C_0 exists because the probability that A succeeds on X is $\alpha < q_2$, and since success is defined in terms of capital having limsup equal to ∞. Of course, the left hand side of (6) is $\geq \alpha$. The R-strategy B needs to know the values q_1 and C_0, but not q_2 or α.

The first phase of B's computation is devoted to finding an initial finite sequence of random bits that leads to success with probability $\geq \beta$. For this, B simulates the initial computation of A on all possible finite random sequences r. Namely, for successive values $m = 1, 2, 3, \ldots$, the strategy B simulates all possible runs of A against X for m steps, using all 2^m different sequences of random bits in $\{0,1\}^m$. As these simulations are performed, B queries the first m bits of X, by betting evenly with stake value 1 on each bit of X. Although these m bits of X might not all be needed, they are enough to carry out the simulations of A, as A can place at most one bet per step. The simulation of A continues until a value m is reached where the left hand side of the inequality in (6) is seen to have value $> q_1$; namely, until $|\mathcal{R}_0| > q_1 2^m$ where

$$\mathcal{R}_0 \;=\; \{r \in \{0,1\}^m : \exists n{\leq}m \left(C_A^X(r\restriction n) > C_0\right)\}.$$

The strategy B now enters its second phase. For this, B chooses uniformly at random a member r_0 of $\mathcal{R}_0$ and switches to emulating A using r_0 as the first m many random bits for A, and then using randomly chosen bits R for the subsequent steps of A.[3] For the first m queries made by A, the corresponding bits of X are already known: after those queries, B emulates A by betting on the bits of X using the same stake value as A would have.

We need to prove that B has unbounded limsup capital against X with probability $\geq \beta = 1{-}\epsilon$. Let $R \in \{0,1\}^\infty$, and let $r_0 R$ denote the concatenation of the (finite) sequence r_0 and the (infinite) sequence R. We claim that B's capital converges to infinity in the limsup sense whenever

[3]For the second phase of B, we use the terminology "B emulates A" instead of "B simulates A". The point is that B "emulates" A by placing bets with stake values determined by A's algorithm, rather than merely betting evenly with stake value 1 as was done while "simulating" A in phase one. This distinction will be more crucial in the proof of Theorem 12.

$\limsup_n C_A^X((r_0R)\restriction n) = \infty$. To prove this claim, it is enough to assume that A makes infinitely many queries to X, since otherwise $C_A^X((r_0R)\restriction n)$ is constant in the limit. As B emulates A using r_0, it initially uses stake values equal to 1 until m bits of X have been queried. After that, B uses the same stake values as A. This means that B's capital is proportional to $C_A^X((r_0R)\restriction n)$, and therefore implies the claim.

We write $R \sqsupset \mathcal{R}_0$ to mean that $r_0 \sqsubset R$ for some $r_0 \in \mathcal{R}_0$. Since B chooses an $r_0 \in \mathcal{R}_0$ at random and then emulates A using a random $R \sqsupset r_0$, the probability that B has unbounded capital against X can be bounded below as follows.

$$\begin{aligned}
&\mathrm{Prob}[\limsup_n C_B^X(R\restriction n) = \infty] \\
&\quad \geq \quad \mathrm{Prob}[\limsup_n C_A^X(R\restriction n) = \infty \mid R \sqsupset \mathcal{R}_0] \\
&\quad = \quad 1 - \frac{\mathrm{Prob}[R \sqsupset \mathcal{R}_0 \ \&\ \limsup_n C_A^X(R\restriction n) < \infty]}{\mathrm{Prob}[R \sqsupset \mathcal{R}_0]} \\
&\quad \geq \quad 1 - \frac{\mathrm{Prob}[\sup_n C_A^X(R\restriction n) > C_0 \ \&\ \limsup_n C_A^X(R\restriction n) < \infty]}{\mathrm{Prob}[R \sqsupset \mathcal{R}_0]} \\
&\quad = \quad 1 - \frac{\mathrm{Prob}[\sup_n C_A^X(R\restriction n) > C_0] - \mathrm{Prob}[\limsup_n C_A^X(R\restriction n) = \infty]}{\mathrm{Prob}[R \sqsupset \mathcal{R}_0]} \\
&\quad \geq \quad 1 - \frac{q_2 - \alpha}{q_1} \ \geq \ 1 - \frac{q_2 - q_1}{q_1} \ \geq \ 1 - \epsilon.
\end{aligned}$$

This proves Theorem 10. □

The above proof can be iterated to prove Theorem 12, the second main theorem of this section. This will give a stronger conclusion than Theorem 10, with $\mathsf{P1}$-random in place of $\mathsf{P}\beta$-random, but under the additional assumption of local weakness.

Lemma 11 *There is a locally weak $\mathsf{P}\alpha$-strategy for X iff there is a locally weak $\mathsf{P}\alpha$-R-strategy for X.*

Lemma 11 follows by the proof method of Theorem 4.

Theorem 12 *Let $X \in \{0,1\}^\infty$. If there is a locally weak $\mathsf{P}\alpha$-strategy for X then there is a $\mathsf{P1}$-strategy for X.*

(Recall that $\mathsf{P1}$-strategies for X must be locally weak against X, since otherwise there is a non-zero probability of making only finitely many bets.) The two previous theorems imply that if X is $\mathsf{P1}$-random, then X is also

locally weak Pα-random for every $\alpha > 0$. Conversely, if X is locally weak Pα-random, then trivially X is P1-random.

It is an open question whether the assumption of of local weakness can be removed from Theorem 12:

Question 13 *Is every* P1*-random string* Pα*-random? In other words, is every partial computable random string* Pα*-random?*

In preparation for proving Theorem 12, we discuss slow-but-sure (SBS) simulations. Recall that slow-but-sure simulations are used for converting a strategy which succeeds in the "limsup" sense into one which succeeds in the "lim" sense (see, for example, [4, Proposition 6.3.8]). An SBS-strategy splits its current capital C into two portions, the saved capital S and the working capital W. The stake values for bets put the working capital at risk, and thus the working capital W can increase and decrease; however, the saved capital S never decreases. Whenever the working capital exceeds a fixed threshold, part of the working capital is transferred to the saved capital. We adopt the convention that whenever the working capital exceeds 2 units, then 1 unit of capital is transferred to the saved capital.

More formally, if A is a betting strategy or R-strategy, then the *SBS-emulation* of A is the strategy B which operates as follows: Initially, B's saved capital is $S = 0$ and its working capital is $W = 1$. B then emulates A but modifies the stake values so that when A uses stake value q, B uses stake value

$$q_B \;=\; q \cdot \frac{W}{S+W}. \tag{7}$$

The result of the bet transforms the working capital W to $W' = W \pm (1-q)W$ and leaves the saved capital unchanged as $S' = S$. If $W' < 2$, then B's working capital is updated by setting $W := W'$ and the saved capital remains unchanged. However, if $W' \geq 2$, then B moves 1 unit of capital into its saved capital, namely B sets $W := W' - 1$ and $S := S' + 1$.

We use the terminology "B *SBS-simulates* A", to mean that B keeps track of what its saved capital and working capital would have been equal to had it been SBS-emulating A, but that B does not base its stake values on the stake values computed by the simulation of A. The point is that B can SBS-simulate many executions of A, but can SBS-emulate only one execution of A at a time.

Proof (of Theorem 12) Let A be a locally weak limsup Pα-R-strategy for X that uses dyadic randomness, so that $\limsup_n C_A^X(R\restriction n) = \infty$ holds with probability α for random $R \in \{0,1\}^\infty$. By Theorem 10, we may assume

$\alpha > \frac{1}{2}$. Without loss of generality, by Lemma 6, A never uses stake values 0 or 2. We must define a P1-R-strategy B against X. The strategy B has an infinite sequence of phases, each of which is somewhat like the second phase of B from the proof of Theorem 10 in that B will SBS-emulate a particular computation of A. However, at the same time, B will also SBS-simulate all other possible computations of A. At the end of each phase, B halts its SBS-emulation, and makes a new selection of the initial random bits of A for the next phase.

In more detail, the strategy B works as follows: B will be an SBS-type strategy and maintains separate values for its working capital W and saved capital S. The initial capital values are $W_0 = 1$ and $S_0 = 0$. We also set $T_0 = 0$, since no bits of X have been bet upon yet.

Phase i: (for $i > 0$.) At the beginning of Phase i, B has already bet on (and thus knows) T_{i-1} many bits of X, and it has saved capital S_{i-1} and working capital $W_{i-1} > 0$. Now B begins SBS-emulating A using a sequence of randomly selected bits which we denote r_i. The first T_{i-1} many bets placed by the (SBS-)emulation are already known and cannot be bet upon again. However, once the emulation of A makes more than T_{i-1} many bets, B begins SBS-emulating A using equation (7) to set its stake values. The SBS-emulation of A keeps track of the values m and t, where m is the total number of steps performed by the entire execution of A, and $t = t(m)$ is the total number of bets that have been placed by A. (The count t includes the first T_{i-1} bets placed against the bits of X that are already known from earlier phases.) The hope is that the SBS-emulation of A will cause B to increase its savings value to at least $S_{i-1} + 1$ before the end of Phase i.

As this SBS-emulation is carried out, B also SBS-*simulates* all possible executions of A that use $\leq m$ many steps and that make $\leq t$ many bets (using all possible choices of random bits r). As B SBS-emulates A and the values m and t increase, B computes a prefix-free set $\mathcal{P} = \mathcal{P}(m,t)$ of finite strings of choices of random bits for which A has been SBS-simulated. Namely, $\mathcal{P}(m,t)$ is the set of those r such that either (a) $|r| = m$ and A run with random bits r on input X makes $\leq t$ bets in its first m moves, or (b) $|r| < m$ and A run with random bits r on input X makes its t-th bet in its $|r|$-th move.

For each $r \in \mathcal{P}(m,t)$, B also computes the associated saved capital and working capital values; the intuition being that these are the capital values that B would have achieved had it used r instead of r_i in the SBS-emulation described two paragraphs earlier. More precisely, B first uses a $r \in \mathcal{P}(m,t)$ and the known bits of X to simulate A until the simulation makes T_{i-1} bets

on bits of X. As the SBS-simulation continues, B then computes the capital values that would have been achieved by B starting with saved capital S_{i-1} and working capital W_{i-1} and using (7) to compute stake values. We define $r \in \mathcal{P}(m,t)$ to be *savings-successful* if the calculation indicates that B would have achieved a saved capital value $\geq S_{i-1}+1$. Let $\mathcal{R}(m,t)$ be the set

$$\mathcal{R}(m,t) \;=\; \{r \in \mathcal{P}(m,t) : r \text{ is savings-successful}\}.$$

Phase i ends once $\mu(\mathcal{R}(m,t)) > \frac{1}{2}$. At this point, the SBS-emulation of A is halted. T_i is set equal to t, namely the number of bits of X that are known. We define S_i and W_i to be the saved capital and working capital achieved by B at the end of the SBS-emulation using r_i. Since A never uses stake values 0 or 2, we have $W_i \neq 0$.

This completes the description of the algorithm for B. To finish the proof of Theorem 12, we will show that Phase i ends with $S_i > S_{i-1}$ with probability $\geq 1/2$, where the probability is taken over the random choice of r_i. In other words, we need to show that, at the end of Phase i, the SBS-emulation of A corresponds to one of the savings-successful computations of $\mathcal{R}(m,t)$ with probability $\geq 1/2$.

We first claim that each Phase i halts with probability 1. Recall that m and $t = t(m)$ measure the number of steps and the number of bets carried out by the SBS-emulation of A using r_i. If Phase i does not halt, then m increases without bound, and since A is locally weak against X, the value of t also increases unboundedly with probability 1. Therefore, with probability one, if Phase i does not halt, the SBS-simulations with all possible r are carried out indefinitely (without being permanently halted at any finite stage by the value of t). Let $\mathcal{R}_{succ}$ be the set of random choices for which A is limsup successful against X:

$$\mathcal{R}_{succ} \;=\; \{R \in \{0,1\}^\infty : \limsup_n C_A^X(R\restriction n) = \infty\}.$$

Of course,

$$\mu(\mathcal{R}_{succ}) \;=\; \alpha \;>\; \tfrac{1}{2}.$$

For $R \in \mathcal{R}_{succ}$, the Phase i SBS-simulation of A using R will be savings-successful, unless it is limited by the values of m and $t(m)$ from the Phase i SBS-emulation. If follows that, with probability one, $\mu(\mathcal{R}(m,t))$ will surpass $\frac{1}{2}$. Thus, with probability one, Phase i will eventually halt.

We must also compute a lower bound for the probability that $S_i > S_{i-1}$. The intuition is that the SBS-simulations of Phase i are run until more than half of them are savings-successful, and that since the SBS-emulation was

carried out for a randomly chosen r_i, it will also be savings-successful and thus $S_i > S_{i-1}$ with probability $\geq \frac{1}{2}$. This intuition is accurate enough, but there is a minor complication caused by the fact that the SBS-simulations are controlled by the values of $t(m)$, and these depend on the SBS-emulation. To handle the complication, we think of B as selecting an *infinite* sequence of random bits r_i before running the SBS-emulation. (Of course, in actuality, B randomly selects additional bits of r_i only as they are needed.) Define m_{r_i} and t_{r_i} to be the first values of m and $t(m)$ at which the SBS-emulation of A using r_i causes the saved capital of B to increase from S_{i-1}; except, if no such values exist, set $m_{r_i} = t_{r_i} = \infty$.

Then $S_i > S_{i-1}$ holds at the end of Phase $i+1$ precisely when

$$\mu(\mathcal{R}(m_{r_i} - 1, t_{r_i} - 1)) \leq \frac{1}{2}.$$

An infinite sequence R is in $\mathcal{R}(m_{r_i} - 1, t_{r_i} - 1)$ iff both $m_R \leq m_{r_i} - 1$ and $t_R \leq t_{r_i} - 1$. We write $R \prec r_i$ for this condition, namely,

$$R \prec r_i \quad \Leftrightarrow \quad m_R < m_{r_i} \text{ and } t_R < t_{r_i}.$$

The $\prec$ relation is a partial order. Thus Phase i successfully causes S_i to be greater than S_{i-1} provided r_i satisfies

$$\mu(\{R : R \prec r_i\}) \ \leq \ \frac{1}{2}.$$

It follows immediately that a randomly chosen r_i causes $S_i > S_{i-1}$ with probability $\geq \frac{1}{2}$.

Therefore, with probability one, $S_i > S_{i-1}$ for infinitely many values of i. Since the saving capital is always an integer, it follows that $\sup_i S_i = \infty$ with probability one. This proves Theorem 12. □

5 Every ML-random is Pα-random

The next theorem is a strengthening of the fact that every Martin-Löf random sequence is partial computably random.

Theorem 14 *Let $0 < \alpha \leq 1$. If there is a (limsup) Pα-strategy against a sequence X then X is not ML-random.*

Corollary 15 *Every ML-random sequence is Pα-random for all $0 < \alpha \leq 1$.*

Question 16 *Is every Pα-random sequence an ML-random sequence?*

Proof Let the probabilistic betting strategy A be $\mathsf{P}\alpha$-successful against X. Let T be the set of valid A-configurations (π, σ) satisfying $\pi \in R(|\sigma|)$. We view T as an infinitely branching tree by letting the infinitely many children of (π, σ) be the elements of the set

$$\{(\pi \mathsf{w}^j \mathsf{b}, \sigma 0), (\pi \mathsf{w}^j \mathsf{b}, \sigma 1) : j \in \mathbb{N}\}.$$

The nodes (π, σ) and (π', σ') are *incomparable*, provided they are distinct and neither is an ancestor of the other in T.

For $i > 0$, let S_i equal the set of pairs $(\pi, \sigma) \in T$ such that (1) $C_A(\pi, \sigma) \geq 2^i$, and (2) no ancestor (π', σ') of (π, σ) has capital $C_A(\pi', \sigma') \geq 2^i$. The S_i's are uniformly computable because the capital function C_A is computable. Note that the sets S_i are disjoint since A's capital value can at most double in a single bet. Furthermore, for each i, the members of S_i are pairwise incomparable. For $(\pi, \sigma) \in T$ and $\tau \in \{0,1\}^*$, define

$$d_{\pi,\sigma}(\tau) \;=\; \begin{cases} P_A(\pi, \sigma) & \text{if } \tau \sqsupseteq \sigma \\ 2^{(|\tau| - |\sigma|)} P_A(\pi, \sigma) & \text{if } \tau \sqsubset \sigma \\ 0 & \text{otherwise.} \end{cases}$$

It is clear from the definition that $d_{\pi,\sigma}$ satisfies the martingale property (1). Define d_i as

$$d_i(\tau) \;=\; \sum\nolimits_{(\pi,\sigma) \in S_i} d_{\pi,\sigma}(\tau).$$

Finally, define $D(\tau) = \sum_{i>0} d_i$. We will prove that D is a computably enumerable martingale and that $\lim_n D(X\upharpoonright n) = \infty$. This will imply that X is not ML-random.

The function D is clearly computably enumerable (left-c.e.) provided it converges to a finite value. Moreover, whenever it converges, D is a martingale since it is a sum of martingales. By the martingale property (1), it is sufficient to prove that $D(\lambda)\downarrow$ in order to show $D(\tau)\downarrow$ for all τ. For this, we prove that for each i, $d_i(\lambda) \leq 2^{-i}$ and thus that $D(\lambda) \leq 1$. The following variant of Kolmogorov's inequality is useful.

Lemma 17 *Let $(\pi_0, \sigma_0) \in T$ and let S be a subset of T such that the members of S are pairwise incomparable and such that $\pi_0 \sqsubseteq \pi$ and $\sigma_0 \sqsubseteq \sigma$ for all $(\pi, \sigma) \in S$. Further suppose that $i > 0$ and $C_A(\pi, \sigma) \geq 2^i$ for all $(\pi, \sigma) \in S$. Then,*

$$\sum_{(\pi,\sigma) \in S} 2^{-|\sigma|} 2^i P_A(\pi, \sigma) \;\leq\; 2^{-|\sigma_0|} C_A(\pi_0, \sigma_0) P_A(\pi_0, \sigma_0). \tag{8}$$

Proof It suffices to prove the lemma for finite sets S, since (8) is true for S if it is true for every finite subset of S. Of course, the lemma is trivially true when S is empty. For non-empty S, the proof is by induction on

$$k_S \;=\; \max_{(\pi,\sigma)\in S}\{|\sigma| - |\sigma_0|\}.$$

In the base case, $k_S = 0$ and $S = \{(\pi_0, \sigma_0)\}$. By hypothesis, $C_A(\pi_0, \sigma_0) \geq 2^i$, and this implies that (8) holds.

For the induction case, fix $k > 0$, and assume the lemma holds for all S with $k_S < k$. We now prove the lemma for S such that $k_S = k$. The assumption of pairwise incomparability implies that $(\pi_0, \sigma_0) \notin S$. Partition S into subsets $S_{j,x}$, where $j \geq 0$ and $x \in \{0, 1\}$ and where

$$S_{j,x} \;=\; \{(\pi, \sigma) \in S : \pi \sqsupseteq \pi_0 \mathsf{w}^j \mathsf{b} \text{ and } \sigma \sqsupseteq \sigma_0 x\}.$$

The induction hypothesis applied to $S_{j,x}$ over the base point $(\pi_0\mathsf{w}^j\mathsf{b}, \sigma_0 x)$ gives

$$\sum_{(\sigma,\pi)\in S_{j,x}} 2^{-|\sigma|}2^i P_A(\pi, \sigma) \;\leq\; 2^{-(|\sigma_0|+1)} C_A(\pi_0\mathsf{w}^j\mathsf{b}, \sigma_0 x) P_A(\pi_0\mathsf{w}^j\mathsf{b}, \sigma_0 x).$$

From this we get

$$\begin{aligned}
&2^{-|\sigma_0|} C_A(\pi_0, \sigma_0) P_A(\pi_0, \sigma_0) \\
&\quad\geq\; 2^{-|\sigma_0|} C_A(\pi_0, \sigma_0) \sum_{j\in\mathbb{N}} P_A(\pi_0\mathsf{w}^j\mathsf{b}, \sigma_0 0) \\
&\quad=\; 2^{-|\sigma_0|} \sum_{j\in\mathbb{N}} C_A(\pi_0\mathsf{w}^j, \sigma_0) P_A(\pi_0\mathsf{w}^j\mathsf{b}, \sigma_0 0) \\
&\quad=\; 2^{-|\sigma_0|} \sum_{j\in\mathbb{N}} \frac{1}{2}[C_A(\pi_0\mathsf{w}^j\mathsf{b}, \sigma_0 0) + C_A(\pi_0\mathsf{w}^j\mathsf{b}, \sigma_0 1)] P_A(\pi_0\mathsf{w}^j\mathsf{b}, \sigma_0 0) \\
&\quad=\; \sum_{j\in\mathbb{N}} \sum_{x\in\{0,1\}} 2^{-(|\sigma_0|+1)} C_A(\pi_0\mathsf{w}^j\mathsf{b}, \sigma_0 x) P_A(\pi_0\mathsf{w}^j\mathsf{b}, \sigma_0 x) \\
&\quad\geq\; \sum_{j\in\mathbb{N}} \sum_{x\in\{0,1\}} \sum_{(\sigma,\pi)\in S_{j,x}} 2^{-|\sigma|}2^i P_A(\pi, \sigma), \quad \text{by the induction hypothesis} \\
&\quad=\; \sum_{(\sigma,\pi)\in S} 2^{-|\sigma|}2^i P_A(\pi, \sigma).
\end{aligned}$$

This proves the induction step, and Lemma 17. □

Applying Lemma 17 with $S = S_i$ and $\pi_0 = \sigma_0 = \lambda$ gives

$$d_i(\lambda) \;=\; \sum_{(\pi,\sigma)\in S} 2^{-|\sigma|} P_A(\pi,\sigma) \;\leq\; 2^{-i},$$

as required to prove that D is a c.e. martingale.

We complete the proof of Theorem 14 by showing that $\lim_n D(X\upharpoonright n) = \infty$. By assumption on A,

$$\mu_A^X(\{\Pi \in \{\mathsf{b},\mathsf{w}\}^\infty : \lim_n C_A^X(\Pi\upharpoonright n) = \infty\}) \;\geq\; \alpha.$$

Therefore, for each capital threshold 2^i, there is a number n_i such that, if

$$\Pi_i \;=\; \{\Pi \in \{\mathsf{b},\mathsf{w}\}^\infty : \forall n \geq n_i (C_A^X(\Pi\upharpoonright n) \geq 2^i)\}$$

then $\mu_A^X(\Pi_i) \geq \alpha/2$. Note that if $\Pi \in \Pi_i$ and thus $C_A^X(\Pi\upharpoonright n_i) \geq 2^i$, then there is a $(\pi,\sigma) \in S_i$ such that $\pi \sqsubseteq \Pi\upharpoonright n_i$ and $\sigma \sqsubseteq X$. Thus, for $n \geq n_i$,

$$d_i(X\upharpoonright n) \;\geq \sum_{\substack{(\pi,\sigma)\in S_i \\ \sigma\sqsubseteq X\upharpoonright n_i}} P_A(\pi,\sigma) \;\geq\; \mu_A^X(\Pi_i) \;\geq\; \frac{\alpha}{2}.$$

Hence, for each i, $\lim_n d_i(X\upharpoonright n) \geq \alpha/2$. (In fact, $\alpha/2$ can be replaced with α.) Since $\alpha > 0$, it follows that $\lim_n D(X\upharpoonright n) = \infty$, and this completes the proof of Theorem 14. □

6 A probabilistic strategy separation proof

This section presents a proof of the separation between partial computable randomness and Martin-Löf randomness. There are at least two existing proofs of this separation. The first proof, described in [4] and [7], exploits the equivalence between ML-randomness and incompressibility with respect to Kolomogorov complexity. The second proof is one ingredient in Kastermans and Lempp's result [5] separating ML-randomness from a restricted version of randomness with respect to nonmonotonic strategies. We present a translation of this second proof to the language of probabilistic strategies, thereby illustrating how probabilistic strategies can be used to obtain simple and intuitive separation proofs.

Theorem 18 *There is a string $X \in \{0,1\}^\infty$ which is partial computably random but not Martin-Löf random.*

Proof Let $\{d_i\}_{i>0}$ be a computable enumeration of all partial computable martingales. In particular, for each i and each $\sigma \in \{0,1\}^*$, if either $d_i(\sigma 0)$ and $d_i(\sigma 1)$ are defined, then all three of $d_i(\sigma)$, $d_i(\sigma 0)$ and $d_i(\sigma 1)$ are defined and the martingale property (1) holds. For a detailed justification of the enumerability of this set of functions, see [5]. We will define an R-strategy A and a string $X \in \{0,1\}^\infty$ such that $\limsup_n \mathsf{Ex}_A^X(n) = \infty$ while $\limsup_n d_i(X\upharpoonright n) < \infty$ for each i. The string X is partial computably random but, by Theorem 3, not Martin-Löf random.

The string X is constructed in stages, each stage working to defeat some partial computable strategy. The R-strategy A models these stages in the construction of X, and at each stage i, A makes a probabilistic guess about the i-th stage of the construction of X. The i-th stage in the construction of X has the following data:

a. *Data summarizing the martingales that are being worked against.* There are non-negative rational constants $\alpha_1, \ldots, \alpha_i$, and D_i is the martingale defined by

$$D_i(\sigma) \;=\; \alpha_1 d_1(\sigma) + \alpha_2 d_2(\sigma) + \cdots + \alpha_i d_i(\sigma)$$

for all σ. Some of the α_i's are equal to zero, and in this case the term $\alpha_i d_i(\sigma)$ is interpreted as equaling zero, whether or not $d_i(\sigma)\downarrow$.

Once the value α_j has been chosen (during stage j), it remains fixed for all later stages.

b. *Initial segment of X that has already been specified.* There is a string $\sigma_i \in \{0,1\}^*$. We have $\sigma_{i-1} \sqsubset \sigma_i$ for all i.

The constructed string X is the limit of the σ_i's, namely, the X such that $\sigma_i \sqsubset X$ for all i. The following conditions hold for all i:

i. $D_i(\sigma_i) < 2$.

ii. If $\alpha_i = 0$, then $d_i(\sigma_i)\uparrow$.

iii. If $\alpha_i > 0$, then $d_i(\sigma_j)\downarrow$ for all j.

These conditions imply that X is not partial computably random. To prove this, note that if the i-th partial computable martingale d_i succeeds against X, it must be that $d_i(\sigma_j)\downarrow$ for all j, so $\alpha_i > 0$. Then, for all $j \geq i$,

$$d_i(\sigma_j) \;\leq\; D_j(\sigma_j)/\alpha_i \;<\; 2/\alpha_i,$$

which means that $\limsup_n d_i(X\upharpoonright n) < \infty$, so d_i does not succeed against X.

Definition Suppose $\tau \in \{0,1\}^*$ and $D_i(\tau)\downarrow$. The *length k D_i-decreasing extension of τ* is the lexicographically first string $\sigma = \tau x_1 x_2 \cdots x_k$ such that each $x_i \in \{0,1\}$ and

$$D_i(\tau) \;\geq\; D_i(\tau x_1) \;\geq\; D_i(\tau x_1 x_2) \;\geq\; \cdots \;\geq\; D_i(\tau x_1 x_2 \cdots x_k).$$

The intuition is that σ is the string that causes D_i to repeatedly lose bets. Note that σ is well-defined provided that $D_i(\tau x_1 \ldots x_\ell x_{\ell+1})\downarrow$ whenever $D_i(\tau x_1 \cdots x_\ell) \leq D_i(\tau)$ for all $\ell < k$.

To prove that X is not Martin-Löf random, we describe the details of the stages in the construction of the D_i's and σ_i's, and then describe the R-strategy A which is Ex-successful against X. Let σ_0 be the empty string. Assume that D_i, $\alpha_1, \ldots, \alpha_i$, and σ_i have already been defined, with $D_i(\sigma_i) < 2$. The definition of α_{i+1} and σ_{i+1} splits into two cases:

Case (1): If there is a $\tau \sqsupseteq \sigma_i$ such that $D_i(\tau) < 2$ and $d_{i+1}(\tau)\uparrow$, then set τ_i to equal the lexicographically first shortest such τ. Let $k_i = |\tau_i| - |\sigma_i|$ and define σ_{i+1} be the length $k_i{+}3$ D_i-decreasing extension of τ_i. In this case, $d_{i+1}(\sigma_{i+1})\uparrow$, so α_{i+1} is defined to equal 0.

Case (2): Otherwise, let σ_{i+1} be the length 2 D_i-decreasing extension of $\tau_i = \sigma_i$. Note $D_i(\sigma_{i+1}) \leq D_i(\sigma_i) < 2$. Set

$$\alpha_{i+1} \;=\; \frac{2 - D_i(\sigma_{i+1})}{2 d_{i+1}(\sigma_{i+1})} \tag{9}$$

if $d_{i+1}(\sigma_{i+1}) \neq 0$, and set $\alpha_{i+1} = 1$ otherwise. Note that $d_{i+1}(\sigma_{i+1})\downarrow$ and hence $D_{i+1}(\sigma_{i+1})\downarrow$, since otherwise Case (1) would apply. By (9), $\alpha_{i+1} > 0$ and $D_{i+1}(\sigma_{i+1}) < 2$.

It is not hard to verify that the necessary D_i-decreasing extensions required for the two cases exist. The only way for a D_i-extension to not exist is for there to be an $\alpha_j > 0$ such that some $d_j(\tau_i x_1 \cdots x_\ell)$ fails to converge during the process of the forming the D_i-extension. But, $\alpha_j > 0$ means that that Case (1) did not apply in stage j, and this means that the values $d_j(\tau_i x_1 \cdots x_\ell)$ needed for defining the D_i-extension all converge.

The R-strategy A makes probabilistic guesses about the values of k_i and σ_i, and then bets all-or-nothing (stake value equal to 0 or 2) that its guess is correct. Initially, of course, σ_0 is the empty string. Suppose that, at the beginning of the i-th stage, A has already picked the correct values for σ_i and $\alpha_1, \ldots, \alpha_i$. Also, suppose that $\sigma_i \sqsubseteq X$ and that A has already

bet all-or-nothing that the first $|\sigma_i|$ many bits of X equal the bits of σ_i. The R-strategy A then uses a random bit to decide whether Case (1) or (2) holds. With probability $\frac{1}{2}$, A decides that Case (2) occurred: in this case, A computes $\sigma_{i+1} = \sigma_i x_1 x_2$ to equal the length 2 D_i-decreasing extension of σ_i. A does this by calculating $D_i(\sigma 0)$ and $D_i(\sigma 1)$ to determine x_1 and then calculating $D_i(\sigma x_1 0)$ and $D_i(\sigma x_2 1)$ to determine x_2. This is possible since A knows the correct values for all the α_j's. A then bets all-or-nothing that the next two bits of X are equal to $x_1 x_2$. If it is correct that Case (2) holds, this increases A's capital by a factor of 4.

Alternatively, also with probability $\frac{1}{2}$, A decides that Case (1) holds. A then chooses a value $k_i \geq 0$ by choosing bits at random until a bit 1 is chosen, and letting k_i be the number of 0's obtained before that first 1. A chooses k_i more random bits $y_1 y_2 \cdots y_{k_i}$, and sets $\tau_i = \sigma_i y_1 \cdots y_{k_i}$. Finally, A sets $\sigma_{i+1} = \tau_i x_1 \cdots x_{k_i+3}$ to be the length k_i+3 D_i-decreasing extension of τ_i. We have $|\sigma_{i+1}| = |\sigma_i| + 2k_i + 3$; so A then bets that the next $2k_i + 3$ many bits of X are as specified by σ_{i+1}. If A has correctly chosen k_i and τ_i, then this increases A's capital by a factor of 2^{2k_i+3}.

If Case (2) holds, then A correctly decides this with probability $\frac{1}{2}$ and increases its capital by a factor of $2^2 = 4$. If Case (1) holds, then A chooses the correct values for k_i and σ_{i+1} with probability $2^{-(2k_i+2)}$, namely by using one bit to decide Case (1) holds, $k_i + 1$ many bits to determine k_i, and k_i more bits to determine τ_i. In this case, A then multiplies its capital by a factor of 2^{2k+3}. Therefore, in both Cases (1) and (2), A increases its expected capital by a factor of 2. By induction on i, we have $\mathsf{Ex}_A^X(\sigma_{i+1}) \geq 2^{i+1}$. Therefore, $\limsup_n \mathsf{Ex}_A^X(\sigma_{i+1}) = \infty$. This establishes that X is not Martin-Löf random, and completes the proof of Theorem 18. □

References

[1] K. Ambos-Spies, *Algorithmic randomness revisited*, in Language, Logic, and Formalization of Knowledge, Coimbra Lecture and Proceedings of a Symposition held in Siena in September 1997, Bibliotheca, 1998, pp. 33–52.

[2] S. Arora and B. Barak, *Computational Complexity: A Modern Approach*, Cambridge University Press, 2009.

[3] S. R. Buss and M. Minnes, *Probabilistic algorithmic randomness*, Journal of Symbolic Logic, 78 (2013), pp. 579–601.

[4] R. G. Downey and D. Hirschfeldt, *Algorithmic Randomness and Complexity*, Springer, 2010.

[5] B. Kastermans and S. Lempp, *Comparing notions of randomness*, Theoretical Computer Science, 411 (2010), pp. 602–616.

[6] A. A. Muchnik, A. L. Semenov, and V. Uspensky, *Mathematical metaphysics of randomness*, Theoretical Computer Science, 207 (1998), pp. 263–317.

[7] A. Nies, *Computability and Randomness*, Oxford University Press, 2009.

[8] C. P. Schnorr, *Process complexity and effective random tests*, Journal of Computer and System Sciences, 5 (1973), pp. 378–388.