Theoretical Computer Science 452 (2012) 47–55

Contents lists available at SciVerse ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

# Process and truth-table characterisations of randomness

Adam R. Day *

*University of California, Berkeley, United States*

ARTICLE INFO

*Article history:*
Received 22 September 2010
Received in revised form 15 May 2012
Accepted 23 May 2012
Communicated by B. Durand

ABSTRACT

This paper uses quick process machines to provide characterisations of computable randomness, Schnorr randomness and weak randomness. The quick process machine is a type of process machine first considered in work of Levin and Zvonkin. A new technique for building process machines and quick process machines is presented. This technique is similar to the KC theorem for prefix-free machines. Using this technique, a method of translating computable martingales to quick process machines is given. This translation forms the basis for these new randomness characterisations. Quick process machines are also used to provide characterisations of computable randomness, Schnorr randomness, and weak randomness in terms of truth-table reducibility.



## 1. Introduction

Underlying the study of randomness is our intuition about how a random real number should behave. This intuition can be expressed through different perspectives such as:

- Compressibility — a real number is random if it is incompressible.
- Betting — a real number is random if a gambler could not make money betting on its bits.
- Test — a real number is random if it has no rare properties.

The first two perspectives use that fact that a real number can be represented by an infinite binary sequence. An intriguing aspect of the study of randomness, is that these perspectives, once formalised, often lead to equivalent definitions.

The betting perspective is typically formalised using martingales. This use of martingales has found widespread application. For example, Lutz pioneered the use of martingales to study the exponential time complexity classes [13]. In this paper, we show how to change between the betting perspective and the compressibility perspective by translating back and forth between martingales and a variant of process machines. This allows us to provide consistent compressibility based definitions of several types of randomness. The main theorems of this paper provide new characterisations of computable randomness, Schnorr randomness and weak randomness. The quick process machines that we will use come from work of Levin and Zvonkin [11].

**Definition.** A *martingale* is a function $d : 2^{<\omega} \to \mathbb{R}^{\geq 0}$ (where $2^{<\omega}$ is the set of all finite binary strings) such that for all $\sigma \in 2^{<\omega}$ we have

$$d(\sigma) = \frac{d(\sigma 0) + d(\sigma 1)}{2}. \tag{1.1}$$

* Tel.: +1 43898007.
*E-mail addresses:* adam.r.day@gmail.com, adam.day@math.berkeley.edu.

0304-3975/$ – see front matter 
doi:10.1016/j.tcs.2012.05.028

A martingale is a strategy for betting of the bits of a real number. The martingale condition (1.1) ensures that the betting is fair. We say that a martingale *succeeds* on a real $\alpha$ if $\lim_{n\to\infty} d(\alpha \upharpoonright n) = +\infty$ (where $\alpha \upharpoonright n$ is the first $n$ bits of some binary representation of $\alpha$).

The idea is that a real number is not random if a gambler could make an infinite amount of money betting on the bits of the real number. For this to work, we need to place some effectivity constraints on the martingales, and potentially the speed at which the gambler makes money. By changing the constraints we get different notions of randomness.

There are a number of different randomness notions that we will investigate in this paper: Martin-Löf randomness, computable randomness, Schnorr randomness and weak randomness. These notions were originally defined in very different ways. However, they can all be characterised in terms of martingales. For simplicity, we will take these martingale characterisations as our definitions.

**Definition.** A function $h : \mathbb{N} \to \mathbb{N}$ is called an *order function* if it is computable, non-decreasing, and unbounded.

**Definition.** (1) A real $\alpha$ is *Martin-Löf random* if no computably enumerable martingale succeeds on $\alpha$.
(2) A real $\alpha$ is *computably random* if no computable martingale succeeds on $\alpha$.
(3) A real $\alpha$ is *Schnorr random* if for all computable martingales $d$, for all orders $h$, for almost all $n$, $d(\alpha \upharpoonright n) < h(n)$.
(4) A real $\alpha$ is *weakly random* if for all computable martingales $d$, for all orders $h$, there exists an $n$, $d(\alpha \upharpoonright n) < h(n)$.

Martin-Löf randomness and Schnorr randomness are both named after the people who first defined them [14,19]. The characterisation of Martin-Löf randomness in terms of martingales is due to Schnorr [19]. In the same paper Schnorr defined computable randomness. Schnorr argued that the effectivity requirements in the definition of computable randomness were insufficient. He suggested that to make the martingale truly effective, the gambler should be able to identify when they were winning. Weak randomness was first defined by Kurtz [9]. The characterisation in terms of martingales was established by Wang [23]. For a full discussion of the early history of this field see Downey and Hirschfeldt, or Li and Vitányi [5,12].

We can weaken the notion of martingale success as follows. We can say that a martingale $d$ succeeds on a real $\alpha$ if $\lim_{\sup} d(\alpha \upharpoonright n) = +\infty$. Changing the definition of martingale success in this way does not alter the underlying notion of randomness. This is because given a $d$ that succeeds, in this weakened sense, on a real $\alpha$, another martingale $\widehat{d}$ can be defined such that $\lim_{n\to\infty} \widehat{d}(\alpha \upharpoonright n) = +\infty$. The definition of $\widehat{d}$ uses a technique known as the savings trick. Details of the savings trick can be found in Downey and Hirschfeldt, or Nies [5,16].

Initial work by Solomonoff, Kolmogorov and Chaitin showed that compressibility could be used to describe randomness for finite strings [2,8,22]. Following this, Levin and Schnorr showed that compressibility could be used to characterise Martin-Löf randomness [10,21]. Schnorr provided a characterisation in terms of process machines. In order to give this characterisation, first let us denote the set of all binary strings of length $n$, the set of all finite binary strings, and the set of all infinite binary strings, by $\{0,1\}^n$, $2^{<\omega}$, and $2^\omega$ respectively. The empty string will be represented by $\lambda$. The relation $\preceq$ on $2^{<\omega} \times (2^{<\omega} \cup 2^\omega)$ is defined by $\sigma \preceq \tau$ if $\sigma$ is an initial segment of $\tau$. We say $\sigma \prec \tau$ if $\sigma \preceq \tau$ and $\sigma \neq \tau$. The relations $\succeq$ and $\succ$ are defined to be the inverse relations of $\preceq$ and $\prec$ respectively. If $\sigma$ is a finite string and $\tau$ a finite or infinite string, then we will write $\sigma\tau$ to represent the concatenation of $\sigma$ and $\tau$.

**Definition.** A *process machine* is a partial computable function $M : 2^{<\omega} \to 2^{<\omega}$ such that if $\tau \in \mathrm{dom}(M)$, and $\tau' \preceq \tau$, then $\tau' \in \mathrm{dom}(M)$ and $M(\tau') \preceq M(\tau)$.

This definition of a process machine was given by Levin and Zvonkin [11]. Independently to Levin and Zvonkin, Schnorr defined a very similar notion of a process machine [21]. In the Schnorr definition, it is not necessary for the domain of the process machine to be closed downwards i.e. a Schnorr process machine is a partial computable function $M : 2^{<\omega} \to 2^{<\omega}$ such that if $\tau, \tau' \in \mathrm{dom}(M)$, and $\tau' \preceq \tau$, then $M(\tau') \preceq M(\tau)$.

Given a process machine $P$, we call $\tau$ a $P$-description of $\sigma$ if $P(\tau) = \sigma$. We define the complexity of a string $\sigma$ with respect to $P$ in terms of its shortest $P$-description:

$$C^P(\sigma) = \begin{cases} \min\{|\tau| : P(\tau) = \sigma\} & \text{if } \exists \tau \in 2^{<\omega}, P(\tau) = \sigma \\ \infty & \text{otherwise.} \end{cases}$$

Schnorr showed that a real $\alpha$ is Martin-Löf random if and only if for all process machines $P$, $C^P(\alpha \upharpoonright n) \geq n - O(1)$ [21]. Note that this theorem holds for both definitions of process machine. (For a discussion of the differences between the complexities induced by the different definitions of process machine see Day [3].) Levin also provided a similar characterisation using monotonic complexity [10].

Levin and Zvonkin defined a process machine $P$ as being *applicable* to a real number $\alpha$, if there was an order $h$ such that $|P(\alpha \upharpoonright n)| \geq h(n)$. We suggest the following name for a process machine that is applicable to all real numbers.

**Definition.** A process machine $P$ is a *quick process machine*, if it is total and there is an order function $h$ such that for all $\tau \in 2^{<\omega}$, $|P(\tau)| \geq h(|\tau|)$.

By compactness, it is equivalent to require that $P$ is defined and unbounded in length, on the prefixes of any real. Given a quick process machine $P$, there is a simple procedure to determine the complexity of any string $\sigma$ with respect to $P$ i.e. $C^P(\sigma)$ is a computable function. This is due to a combination of the facts that $P$ is total and $h$ is an order function. Because $h$ is an order function, there are only finitely many strings in the domain of $P$ that could map to $\sigma$. Hence we can compute $P(\tau)$ for all strings $\tau$ such that $h(|\tau|) \leq |\sigma|$. If $\sigma$ has a $P$-description, it must be one of these strings, and thus we can compute the minimum length of a description.

In Theorem 2.13, a characterisation of computable randomness, Schnorr randomness, and weak randomness in terms of quick process machines is made. These are not the first compressibility characterisations of these types of randomness. Downey and Griffiths have characterised Schnorr randomness in terms of computable measure machines [6]. Mihailović provided a machine characterisation of computable randomness in terms of bounded measure machines [5]. Downey, Griffiths and Reid characterised weak randomness in terms of computably layered machines [7]. The value of these new characterisations lies in their simplicity and consistency.

A *prefix-free machine*, is a partial computable function $M : 2^{<\omega} \to 2^{<\omega}$ such that the domain of $M$ forms an anti-chain with respect to $\preceq$. One reason that prefix-free machines have been widely used in the study of randomness is the existence of the KC theorem [5].[1] The KC theorem provides a means of building prefix-free machines via requests. Given a computable sequence $S = (\sigma_0, n_0), (\sigma_1, n_1), \ldots$, such that $\sum_{i\in\mathbb{N}} 2^{-n_i} \leq 1$ (this sum is the weight of the requests), the KC theorem establishes the existence of a prefix-free machine $M$ such that for all $i \in \mathbb{N}$, there is some string $\tau_i$ of length $n_i$ such that $M(\tau_i) = \sigma_i$. In fact we can regard our sequence $S$ as returning a sequence $\tau_0, \tau_1, \ldots$ of pairwise incomparable descriptions where $|\tau_i| = n_i$. The KC theorem greatly simplifies the construction of prefix-free machines. Underlying Theorem 2.13, is a new technique for building process machines. This technique can be thought of as a KC theorem for process machines. It allows process machines to be built by listing the strings that need descriptions, the description length and the relationship between described strings. This technique is presented in Theorems 2.2 and 2.3.

Demuth showed that there is a link between truth-table reducibility and randomness [4]. For an updated treatment of Demuth's theorem see Bienvenu and Porter [1]. We provide further evidence for this link. There is a close relationship between quick process machines and truth-table functionals. We use this relationship to provide truth-table reducibility characterisations of computable randomness, Schnorr randomness and weak randomness in Theorem 3.1.

## 2. Quick process machines and randomness

Our goal is to show that quick process machines can be used to characterise computable randomness, Schnorr randomness and weak randomness. This is due to the fact that quick process machines are very similar to martingales.

Proposition 2.1 shows how to construct a martingale from a quick process machine. This proof is essentially due to Levin (note that the relevant theorems in the paper of Levin and Zvonkin are attributed solely to Levin).

**Proposition 2.1** (*Levin [11]*)**.** *For any quick process machine $P$, there is a computable martingale $d$ such that for all $\sigma$, $d(\sigma) \geq 2^{|\sigma|-C^P(\sigma)}$.*

**Proof.** Let $P$ be a quick process machine with associated order function $h$. Define $g(n) = \min\{x : h(x) > n\}$. If $\tau \in \{0,1\}^{g(n)}$ then $|P(\tau)| \geq h(|\tau|) = h(g(n)) > n$. We define a computable martingale $d$ as follows. First for any string $\sigma$ define $E_\sigma = \{\tau \in \{0,1\}^{g(|\sigma|)} : P(\tau) \succ \sigma\}$. Now define $d$ by

$$d(\sigma) = \frac{|E_\sigma|}{2^{g(|\sigma|)-|\sigma|}}.$$

The function $d$ is computable because $P$ is total so $E_\sigma$ is computable for all $\sigma$. We will now show that $d$ is a martingale. First note that $d(\lambda) = 1$ because $P$ is total. For any $\sigma$, $E_{\sigma 0}$ and $E_{\sigma 1}$ are disjoint because the strings $\sigma 0$ and $\sigma 1$ are incomparable and so no element of the range of $P$ can extend both of them. If $\tau \in E_{\sigma 0}$ we have that $\tau \upharpoonright g(|\sigma|) \in E_\sigma$ because $P(\tau \upharpoonright g(|\sigma|)) \preceq P(\tau)$, $\sigma \preceq P(\tau)$ and $|\sigma| < |P(\tau \upharpoonright g(|\sigma|))|$ so $\sigma \prec P(\tau \upharpoonright g(|\sigma|))$. Similarly if $\tau \in E_{\sigma 1}$, then $\tau \upharpoonright g(|\sigma|) \in E_\sigma$.

Now if $\tau \in E_\sigma$, and $\tau' \succeq \tau$ with $|\tau'| = g(|\sigma|+1)$ then it must be that $|P(\tau')| \geq h(|\tau'|) = h(g(|\sigma|+1)) > |\sigma|+1$. Hence $\tau' \in E_{\sigma 0} \cup E_{\sigma 1}$. Thus we have that $(|E_{\sigma 0}| + |E_{\sigma 1}|)2^{g(|\sigma|)-g(|\sigma+1|)} = |E_\sigma|$. This gives us that

$$d(\sigma) = \frac{|E_\sigma|}{2^{g(|\sigma|)-|\sigma|}} = \frac{|E_{\sigma 0}| + |E_{\sigma 1}|}{2^{g(|\sigma+1|)-|\sigma|}} = \frac{1}{2}\frac{|E_{\sigma 0}| + |E_{\sigma 1}|}{2^{g(|\sigma+1|)-(|\sigma|+1)}} = \frac{d(\sigma 0) + d(\sigma 1)}{2}.$$

Assume $C^P(\sigma) = |\sigma| - c$, and $|\sigma| = n$. If so, for some $\tau$ with $|\tau| = n - c$, we have that $P(\tau) = \sigma$. This means that $h(|\tau|) \leq n$ and consequently that $g(n) > |\tau|$. If $\tau' \succeq \tau$ with $|\tau'| = g(n)$, then $P(\tau') \succ \sigma$ and so $\tau' \in E_\sigma$. Hence $|E_\sigma| \geq 2^{g(n)-(n-c)}$. Thus

$$d(\sigma) = \frac{|E_\sigma|}{2^{g(n)-n}} \geq 2^c = 2^{|\sigma|-C^P(\sigma)}. \quad \square$$

[1] This theorem is known by the names Kraft–Chaitin and Kraft Computable.

A similar result holds, by essentially the same proof, for process machines and left-c.e. martingales. Our next objective is to build a quick process machine from a martingale. While Levin showed how to build a process machine from a computable measure [11], Levin's objective was to show that any computable measure could be obtained using similar techniques to Proposition 2.1. The process machine Levin built was not total but instead applicable to a set of uniform measure 1. Further, Levin did not relate the complexity of a string $\sigma$ with respect to the process machine created, to the measure of the cylinder $\{\sigma\alpha : \alpha \in 2^\omega\}$.

Before showing how to build a quick process machine from a martingale, we will present a new technique for building process machines. The benefit of this technique is that it allows us to build process machines without worrying about which descriptions to use. Instead, like the KC theorem, we request a description length.

**Definition.** We call a partial computable function $f : \omega^{<\omega} \to 2^{<\omega} \times \mathbb{N}$ a *process request function*, if:

(1) The domain of $f$ is closed downwards;
(2) For all $\rho \in \mathrm{dom}(f)$, $2^{-f_2(\rho)} \geq \sum_{\rho x \in \mathrm{dom}(f)} 2^{-f_2(\rho x)}$; and
(3) If $\rho_1, \rho_2 \in \mathrm{dom}(f)$ and $\rho_1 \prec \rho_2$, then $f_1(\rho_1) \preceq f_1(\rho_2)$, and $f_2(\rho_1) < f_2(\rho_2)$.

In this definition, $f_1$ and $f_2$ are the co-ordinate functions of $f$, and $\omega^{<\omega}$ is the set of all finite strings of elements of $\mathbb{N}$.

Item (1) means closure under prefixes. It is not necessary that $f(\rho 0)$ is defined if $f(\rho 1)$ is defined.

The idea behind this definition is the following. We want to represent the essential combinatorics of a process machine by a tree. Each node of the tree maps to a pair $(\sigma, n)$ where $\sigma \in 2^{<\omega}$ and $n \in \mathbb{N}$. When we turn this tree into a process machine, this node generates a description of $\sigma$ of length $n$. Suppose we have two nodes $\rho_1, \rho_2$. If $\rho_1$ is an initial segment of $\rho_2$, then the description generated by $\rho_1$ will be an initial segment of the description generated by $\rho_2$. We need the second condition so that the combined weight of the descriptions generated by the children of a node, does not exceed the weight of the description generated by the node itself. We need $f_1(\rho_1) \preceq f_1(\rho_2)$ in order to make a process machine. We will use the fact that $f_2(\rho_1) < f_2(\rho_2)$ in the proof of Theorem 2.2. Essentially this ensures that the mapping from elements in the domain of the process request function, to elements in the domain of the process machine is one-to-one.

**Definition.** A process machine $P$ *implements* a process request function $f$ if for all $\sigma \in 2^{<\omega}$, $C^P(\sigma) = \min\{n : (\sigma, n) \in \mathrm{rng}(f)\}$.

**Theorem 2.2.** *Any process request function is implemented by some process machine, and any process machine implements some process request function.*

**Proof.** Given a process machine $P$, there is a natural process request function $f$ that $P$ implements. We define $f$ by $f(\tau) = (P(\tau), |\tau|)$. In this case we have $\mathrm{dom}(f) = \mathrm{dom}(P) \subseteq 2^{<\omega}$.

Given a process request function $f$, each node $\rho \in \mathrm{dom}(f)$ defines a prefix-free machine $M_\rho$ via the KC theorem in the following manner. Let $(\upsilon, n) = f(\rho)$. For any $x \in \mathbb{N}$, if $f(\rho x)$ halts, then let $(\upsilon\sigma_x, m_x + n) = f(\rho x)$ where $m_x > 0$. If this occurs we add $(\sigma_x, m_x)$ to our KC request sequence. Condition (2) for $f$ to be a process request function ensures that the weight of the requests does not exceed 1 because

$$1 \geq \sum_{\rho x \in \mathrm{dom}(f)} 2^{-(f_2(\rho x) - f_2(\rho))} = \sum_{\rho x \in \mathrm{dom}(f)} 2^{-(m_x + n - n)}.$$

Given $\tau$ in the domain of $M_\rho$, we can determine the node $\rho x$ that made the request that returned $\tau$. We will call $\rho x$ the node associated with $\tau$ and $M_\rho$. In the verification we will make use of the fact that $f_1(\rho)M_\rho(\tau) = \upsilon\sigma_x = f_1(\rho x)$.

We will now build a process machine $P$ that implements $f$. If $f$ is the empty function, then so is $P$. Otherwise $f(\lambda)$ halts and so we can define $\tau_0 = 0^{f_2(\lambda)}$, and $P(\tau') = f_1(\lambda)$ for any $\tau' \preceq \tau_0$.

We set $P(\tau) = \sigma$ if at any stage we find a decomposition of $\tau = \tau_0\tau_1 \ldots \tau_n$, and a node $\rho \in \omega^{<\omega}$ of length $n - 1$ with the following properties:

(1) For all $i \in \{0, \ldots, n-1\}$, $\tau_{i+1} \in \mathrm{dom}(M_{\rho\restriction i})$,
(2) For all $i \in \{0, \ldots, n-2\}$, $\rho \restriction (i+1)$ is the node associated with $\tau_{i+1}$ and $M_{\rho\restriction i}$, and
(3) $\sigma = f_1(\lambda)M_{\rho\restriction 0}(\tau_1) \ldots M_{\rho\restriction(n-1)}(\tau_n)$.

Note that because $f_2$ is strictly increasing, for all $i \in \{1, \ldots, n\}$, $\tau_i \neq \lambda$.

To make $P$ a process machine, we need to close the domain of $P$ downwards. To achieve this, we also define $P(\tau') = P(\tau_0\tau_1 \ldots \tau_{n-1})$ for all $\tau'$ such that $\tau_0 \ldots \tau_{n-1} \prec \tau' \prec \tau$.

If $P$ fails to be a process machine, then there must be some $\tau, \pi$ added to the domain of $P$ with $\tau \preceq \pi$ such that $P(\tau) \not\preceq P(\pi)$. Further we can assume that $\tau$ was not added to the domain of $P$ in order to close the domain downwards. Let us take the decompositions used by the construction to be $\tau = \tau_0\tau_1 \ldots \tau_m$ and $\pi \succeq \pi_0\pi_1 \ldots \pi_n$, where $P(\pi) = P(\pi_0\pi_1 \ldots \pi_n)$.

By construction, $\tau_0 = \pi_0 = 0^{f_2(\lambda)}$. Further, $\tau_1 = \pi_1$ because $M_\lambda$ is a prefix-free machine. This also means that the node $\rho_1$ associated with $\tau_1$ and $M_\lambda$, is the same node associated with $\pi_1$ and $M_\lambda$. Hence $\tau_2 = \pi_2$ because $M_{\rho_1}$ is a prefix-free machine. By repeating this argument we establish that $\pi \succeq \tau_0 \ldots \tau_m\pi_{m+1} \ldots \pi_n$. This implies that

$$P(\pi) \succeq f_1(\lambda)M_{\rho_0}(\tau_1) \ldots M_{\rho_{n-1}}(\tau_n) = P(\tau)$$

contracting our assumption that $P$ was not a process machine. Note that because for all $i > 0$, $\pi_i \neq \lambda$, the decomposition of $\sigma$ is unique. This means that $P(\sigma)$ is defined at most once during the construction.

We will now verify that $P$ implements $f$. Let $\tau_0 = 0^{f_2(\lambda)}$. Let $\rho \in \mathrm{dom}(f)$, and let $(\sigma, n) = f(\rho)$. For $i \in \mathbb{N}$, $0 \leq i < |\rho|$, for each machine $M_{\rho|i}$ let $\tau_{i+1}$ be the element of the domain of this machine such that the node $\rho \upharpoonright (i+1)$ is associated with $\tau_{i+1}$ and $M_{\rho|i}$. For all such $i$, $f_1(\rho \upharpoonright i)M_{\rho|i}(\tau_{i+1}) = f_1(\rho \upharpoonright (i+1))$. Hence we have that $f_1(\lambda)M_{\rho|0}(\tau_1)\ldots M_{\rho|(|\rho|-1)}(\tau_{|\rho|}) = f_1(\rho) = \sigma$. Thus $P(\tau_0\tau_1\ldots\tau_n) = \sigma$, and

$$\sum_{i=0}^{|\rho|} |\tau_i| = f_2(\lambda) + \sum_{i=1}^{|\rho|} (f_2(\rho \upharpoonright i) - f_2(\rho \upharpoonright (i-1))) = f_2(\rho) = n. \quad \square$$

The proof shows that the implementation is uniform; given an index for a process request function $f$, we can compute an index for a process machine $P$ that implements $f$.

We can do a similar thing for quick process machines. The first two conditions below ensure that the quick process machine we create is total. The third condition ensures that once we build a process machine there is some order function $h$ such that for all $\tau \in 2^{<\omega}$, $|P(\tau)| \geq h(|\tau|)$.

**Definition.** A process request function $f$ is a *quick process request function* if:

(1) $\mathrm{dom}(f)$ is finitely branching,
(2) For all $n \in \mathbb{N}$, $\sum_{\rho \in \mathrm{dom}(f), |\rho|=n} 2^{-f_2(\rho)} = 1$,
(3) For some order $h$, for all $\rho \in \mathrm{dom}(f)$, $|f_1(\rho)| \geq h(|\rho|)$.

**Theorem 2.3.** *Any quick process request function is implemented by a quick process machine, and any quick process machine implements a quick process request function.*

**Proof.** Given a quick process machine $P$, the natural quick process request function that $P$ implements is again defined by $f(\tau) = (P(\tau), |\tau|)$.

Given a quick process request function $f$, we use Theorem 2.2 to construct a process machine $P$. The additional conditions on $f$, that $\mathrm{dom}(f)$ is finitely branching, and for all $n \in \mathbb{N}$, $\sum_{\rho \in \mathrm{dom}(f), |\rho|=n} 2^{-f_2(\rho)} = 1$ ensure that $P$ is total.

Let $l(x) = \max\{f_2(\rho) : |\rho| = x \text{ and } \rho \in \mathrm{dom}(f)\}$. This function is computable because the domain of $f$ is computable and finitely branching. If $|\tau| \geq l(x)$, then for some $\tau' \preceq \tau$ we have that $P(\tau') = f_1(\rho)$ for some $\rho$ of length $x$. This follows from condition 2 for $f$ to be a quick process request function. This means that $|P(\tau)| \geq |f_1(\rho)| \geq h(x)$.

We can define another order function $h'$ by $h'(n) = \max\{x : l(x) \leq n\}$. As $|\tau| \geq l(h'(|\tau|))$, we have that $|P(\tau)| \geq h(h'(|\tau|))$ and because $h \circ h'$ is also an order function, $P$ is a quick process machine. $\square$

We will now present a method for translating from computable martingales to quick process machines. One difficulty is that, when running a computable martingale, the value of $d(\sigma)$ can increase at any stage, albeit by a very small amount. However, we can avoid this issue by using the following result of Schnorr [20].

**Proposition 2.4** (*Schnorr*)**.** *If $d$ is a computable martingale, then there exists a computable martingale $\widehat{d}$ such that for all $\sigma \in 2^{<\omega}$:*

(1) *$\underline{d}(\sigma) \leq \widehat{d}(\sigma) \leq d(\sigma) + 2$; and*
(2) *$\widehat{d}(\sigma)$ is a computable dyadic rational i.e. there is an integer pair $(n, z)$ uniformly computable from $\sigma$ such that $\widehat{d}(\sigma) = n2^z$.*

From now on we will assume that all our martingales $d$ are dyadic rational valued, and further that $d(\lambda) = 1$. Given a martingale, we can define its *measure precision function* $r : \mathbb{N} \to \mathbb{N}$ by

$$r(n) = \min\{m \in \mathbb{N} : \forall \sigma \in 2^{<\omega}, |\sigma| = n \Rightarrow d(\sigma) \cdot 2^{-|\sigma|} \text{ is an integer multiple of } 2^{-m}\}.$$

We call this the measure precision function because there is a standard transformation from a martingale to a measure on $2^\omega$ by taking, for any finite string $\sigma$, the measure of the basic open set $\{\alpha \in 2^\omega : \sigma \prec \alpha\}$ to be $d(\sigma) \cdot 2^{-|\sigma|}$. The measure precision function of a computable martingale is computable. We will also make use of the following lemma.

**Lemma 2.5.** *Let $(a_1, a_2, \ldots, a_n)$ be a tuple of positive integers and $m \in \mathbb{N}$ such that $m \leq a_i$ for all $1 \leq i \leq n$. If $\sum_{i=1}^n 2^{-a_i} \geq 2^{-m}$, then for some $J \subseteq \{1, \ldots, n\}$ we have that $\sum_{i \in J} 2^{-a_i} = 2^{-m}$.*

**Proof.** We can assume that the elements of the tuple are non-decreasing. Consider the partial sum $S_k = \sum_{i=1}^k 2^{-a_i}$. $S_1 \leq 2^{-m}$. If $S_k < 2^{-m}$, it must be that $S_k \leq 2^{-m} - 2^{-a_k}$ (as $S_k$ is some integer multiple of $2^{-a_k}$). As $a_{k+1} \geq a_k$, so $S_{k+1} \leq 2^{-m}$. Hence for the least $k$ such that $S_k \geq 2^{-m}$, it must be that $S_k = 2^{-m}$. $\square$

Given a computable martingale $d$, it may not be possible to find a quick process machine $P$, such that for all strings $\sigma$, $C^P(\sigma) =^+ |\sigma| - \lfloor \log d(\sigma) \rfloor$. For example, consider the martingale defined by $d(\sigma) = 2^n$ if $\sigma = 1^n$ for some $n$, and $d(\sigma) = 0$ otherwise. In this example, for any $n$, $|1^n| - \lfloor \log d(1^n) \rfloor = 0$ and a process machine has only one description of length 0. The problem with this example is that the martingale wins too quickly on the sequence $1^\omega$. We will restrict ourselves to martingales that do not win "too quickly" on any sequence.

Our objective is to determine an infinite set $M$, and build a quick process machine $P$ such that for all strings $\sigma$ such that $|\sigma| \in M$, $C^P(\sigma) = |\sigma| - \lfloor \log d(\sigma) \rfloor$. We will take $M$ to be the range of a strictly increasing computable function. Given a

computable martingale $d$ with measure precision function $r$, we will say that a strictly increasing computable function $m$ is a *selection function* for $d$ if:

(1) For all $n \in \mathbb{N}$, $m(n+1) \geq r(m(n)) + n + 2$; and
(2) For all $n \in \mathbb{N}$, for all $\sigma \in 2^{<\omega}$, $|\sigma| = m(n) \Rightarrow d(\sigma) \leq 2^n$.

**Proposition 2.6.** *Let $d$ be a computable martingale and let $m$ be a selection function for $d$. There exists a quick process machine $P$ such that for all $\sigma \in 2^{<\omega}$ with $|\sigma| \in \mathrm{rng}(m)$, $C^P(\sigma) = |\sigma| - \lfloor \log d(\sigma) \rfloor$.*

**Proof.** We will assume that $m(0) = 0$ as this simplifies the exposition of the proof, and this is all we will need later. We construct $P$ using a quick process request function $f$. Let $r$ be the measure precision function for $d$. At stage $s$ in the construction we will define $f$ on all strings $\rho$ in the domain of $f$ of length $s$. We will prove that the function $f$ has the property that if $f(\rho) = (\sigma, x)$ with $|\rho| = n+1$, then $|\sigma| = m(n+1)$ and $r(m(n)) < x \leq r(m(n+1))$. As $m(n+1)$ is defined to be larger than $r(m(n))$, this allows us to request short descriptions of strings of length $m(n+1)$ if necessary.

At stage 0, we set $f(\lambda) = (\lambda, 0)$. At stage $s+1$, for all $\sigma \in 2^{<\omega}$ such that $|\sigma| = m(s)$, we let $E_\sigma = \{\rho \in \omega^s : f_1(\rho) = \sigma\}$. We make the following inductive assumptions on the construction:

(1) For all $\rho \in E_\sigma, f_2(\rho) \leq r(m(s))$; and
(2) $\sum_{\rho \in E_\sigma} 2^{-f_2(\rho)} = d(\sigma) 2^{-|\sigma|}$.

If $s = 0$, then $E_\lambda = \{\lambda\}$ and the inductive assumptions hold.

Choose $\sigma \in \{0,1\}^{m(s)}$. Let $\sigma_0, \ldots, \sigma_n$ be all extensions of $\sigma$ of length $m(s+1)$. Now as $d$ is a dyadic rational valued martingale, for each $i$, there is some finite set of integers $A_i$ such that

$$\sum_{a \in A_i} 2^{-a} = d(\sigma_i) 2^{-|\sigma_i|}. \tag{2.1}$$

Hence by applying the martingale condition and the second inductive assumption:

$$\sum_{i=0}^{n} \sum_{a \in A_i} 2^{-a} = \sum_{i=0}^{n} d(\sigma_i) 2^{-|\sigma_i|} = d(\sigma) 2^{-|\sigma|} = \sum_{\rho \in E_\sigma} 2^{-|f_2(\rho)|}.$$

Fix an $i$, and fix any $a \in A_i$. We want to show that $r(m(s)) < a \leq r(m(s+1))$. First $a \leq r(m(s+1))$ because $r$ is a measure precision function for $d$. Hence $d(\sigma_i) 2^{-|\sigma_i|}$ is an integer multiple of $2^{-r(m(s+1))}$ and thus $a \leq r(m(s+1))$. By (2.1), $2^{-a} \leq d(\sigma_i) 2^{-|\sigma_i|}$. Further as $d(\sigma_i) \leq 2^{s+1}$ (the second condition for $m$ to be a selection function), we have that $2^{-a} \leq 2^{s+1-|\sigma_i|}$. Hence $a \geq |\sigma_i| - s - 1 = m(s+1) - s - 1 > r(m(s))$ (the last inequality follows from the first condition for $m$ to be a selection function). Let $I = \{(i,a) : 0 \leq i \leq n, a \in A_i\}$ be an index set for the finite sets $A_i$. If $\rho \in E_\sigma$, then $f_2(\rho) \leq r(m(s)) < \min A_i$. Hence by Lemma 2.5, we can partition $I$ into subsets $S_\rho$ for each $\rho \in E_\sigma$, such that

$$\sum_{(i,a) \in S_\rho} 2^{-a} = 2^{-f_2(\rho)}. \tag{2.2}$$

Now for each such $\rho$, we define $f(\rho\langle i, a\rangle) = (\sigma_i, a)$ if $(i,a) \in S_\rho$ (where $\langle \_, \_ \rangle$ is a computable bijection between $\mathbb{N} \times \mathbb{N}$ and $\mathbb{N}$). We repeat this step for all $\sigma$ of length $m(s)$. Once this has been done for all such $\sigma$, we move to the next stage. Note that our first construction assumption is maintained because $f_2(\rho x) \leq \max\{a : (i,a) \in I\} \leq r(m(s+1))$. Our second construction assumption is maintained because of (2.1) and that every element of $A_i$ generates a new node in the domain of $f$.

This completes the definition of $f$. First we need to verify that $f$ is a quick process request function. This construction ensures that the domain of $f$ is closed downwards, and that each node is finitely branching (as the index set $I$ is always finite). Additionally $f_1$ is strictly increasing because $m$ is strictly increasing, and $f_2$ is strictly increasing because $f_2(\rho) \leq r(m(|\rho|)) < f_2(\rho x)$ for any $x \in \mathbb{N}$ with $\rho x \in \mathrm{dom}(f)$. Eq. (2.2) ensures that we meet condition 2 for $f$ to be a process request function and this along with the fact that $f_2(\lambda) = 0$ implies we meet condition 2 for $f$ to be a strict process request function. From $f$, we can construct a quick process machine $P$ using Theorem 2.3.

Now $P$ has the desired property because for any $n$, take any $\sigma$ with $|\sigma| = m(n)$. Let $A$ be the set of integers used for $\sigma$ in (2.1). If $a \in A$, then there is some description $\tau$ such that $P(\tau) = \sigma$ and $|\tau| = a$. For (2.1) to hold, it must be that $|\sigma| - \lfloor \log d(\sigma) \rfloor \in A$. □

The above construction has an additional property that will be useful when we consider truth-table reductions.

**Lemma 2.7.** *Let $\alpha \in 2^\omega$, let $d$ be a computable martingale and let $m$ be a selection function for $d$. Let $P$ be the quick process machine created by an application of Proposition 2.6. Then the set $\{\tau : \exists n, c \in \mathbb{N}\, P(\tau) = \alpha \upharpoonright m(n)$ and $d(\alpha \upharpoonright m(n)) = 2^c\}$ is a chain with respect to $\preceq$.*

**Proof.** Assume that there exists $\tau_1, \tau_2$ such that $P(\tau_1) = \alpha \upharpoonright m(n_1)$ and $P(\tau_2) = \alpha \upharpoonright m(n_2)$ with $n_1 \leq n_2$ and $d(\alpha \upharpoonright m(n_1)) = 2^c$ for some $c \in \mathbb{N}$.

Now by construction, there is some initial segment $\tau \preceq \tau_2$ such that $|P(\tau)| = m(n_1)$ and hence because $P$ is a process machine we have that $P(\tau) = \alpha \upharpoonright m(n_1)$.

At stage $n_1$ of the construction, we have that $\alpha \upharpoonright m(n_1) = \sigma_i$ for some $i$. Now because $d(\alpha \upharpoonright m(n_1)) = 2^c$, we have that $|A_i| = 1$. Hence $\tau_1$ is the unique element in $2^{<\omega}$ such that $P(\tau_1) = \alpha \upharpoonright m(n_1)$. Thus $\tau_1 = \tau \preceq \tau_2$. □

We can use Proposition 2.6 to build a quick process machine that turns high martingale values into short descriptions. While this proposition only works for martingales with selection functions, we will show that this is not a significant limitation for our purposes. First we consider the case of Schnorr randomness and weak randomness.

**Proposition 2.8.** *Let $d$ be a computable martingale and let $h$ be a computable order. There exists a computable martingale $\widehat{d}$ and $m$, a selection function for $\widehat{d}$, such that for all $\alpha \in 2^\omega$:*

(1) $\forall n\, d(\alpha \upharpoonright n) \geq h(n) \Rightarrow \forall n \widehat{d}(\alpha \upharpoonright m(n)) = 2^n$; *and*
(2) $\exists^\infty n\, d(\alpha \upharpoonright n) \geq h(n) \Rightarrow \exists^\infty n \widehat{d}(\alpha \upharpoonright m(n)) = 2^n$.

**Proof.** We build $\widehat{d}$ by adopting the betting strategy of $d$ unless this strategy would force $\widehat{d}(\sigma) > 2^n$ for some string $\sigma$ of length $m(n)$. In this case, we restrain the betting so that $\widehat{d}(\sigma) = 2^n$.

We construct $\widehat{d}$ and $m$ as follows. At stage 0, define $m(0) = 0$ and $\widehat{d}(\lambda) = 1$. At stage $s+1$, define

$$m(s+1) = \max\{r(m(s)) + s + 2, \min\{x : h(x) \geq 2^{m(s)+s+1}\}\}.$$

For all $\sigma \in 2^{<\omega}$ such that $m(s) < |\sigma| \leq m(s+1)$, we inductively define $\widehat{d}(\sigma)$ by

$$\widehat{d}(\sigma i) = \begin{cases} 2^{s+1} & \text{if } d(\sigma i) \cdot \frac{\widehat{d}(\sigma)}{d(\sigma)} \geq 2^{s+1} \\ 2\widehat{d}(\sigma) - 2^{s+1} & \text{if } d(\sigma(1-i)) \cdot \frac{\widehat{d}(\sigma)}{d(\sigma)} \geq 2^{s+1} \\ d(\sigma i) \cdot \frac{\widehat{d}(\sigma)}{d(\sigma)} & \text{otherwise.} \end{cases}$$

To verify the construction first note that the construction ensures that $m$ is a selection function for $\widehat{d}$. Now consider if for all $n$, $d(\alpha \upharpoonright n) \geq h(n)$. First we have that $\widehat{d}(\alpha \upharpoonright m(0)) = \widehat{d}(\lambda) = 2^0$. Second if $\widehat{d}(\alpha \upharpoonright m(n)) = 2^n$, then by definition of $m$,

$$d(\alpha \upharpoonright m(n+1)) \cdot \frac{\widehat{d}(\alpha \upharpoonright m(n))}{d(\alpha \upharpoonright m(n))} \geq h(m(n+1)) \cdot \frac{2^n}{2^{m(n)}} > 2^{n+1}.$$

This means that $\widehat{d}(\alpha \upharpoonright m(n+1)) = 2^{n+1}$.

Now assume that there are infinitely many $n$ such that $d(\alpha \upharpoonright n) \geq h(n)$. To show that there are infinitely many $n$ such that $\widehat{d}(\alpha \upharpoonright m(n)) = 2^n$, it is sufficient to show that for any $c$, there exists an $n$ and an $s$ such that $d(\alpha \upharpoonright n) \geq 2^{s+c}$ and $m(s) \leq n < m(s+1)$. From this it follows that the construction must restrain the martingale along $\alpha$ infinitely often, and hence for infinitely many $n$, $\widehat{d}(\alpha \upharpoonright m(n)) = 2^n$. Fix any $c$ and choose $s$ and $n$ such that $c \leq m(s-1)$, $m(s) \leq n < m(s+1)$ and $d(\alpha \upharpoonright n) \geq h(n)$. It follows that

$$d(\alpha \upharpoonright n) \geq h(m(s)) \geq 2^{m(s-1)+s} = 2^{c+s}. \quad \square$$

**Corollary 2.9.** (1) *If $\alpha$ is not Schnorr random, then there exists a quick process machine $P$ and a strictly increasing computable function $m$ such that $\exists^\infty n\, C^P(\alpha \upharpoonright m(n)) = m(n) - n$.*
(2) *If $\alpha$ is not weakly random, then there exists a quick process machine $P$ and a strictly increasing computable function $m$ such that $\forall n\, C^P(\alpha \upharpoonright m(n)) \leq m(n) - n$.*

**Proof.** These results follow from combining Propositions 2.6 and 2.8. $\square$

To establish a similar result for computable randomness, we need to vary the construction slightly. In the previous proposition, there was an order function that told us what value the martingale "should" have. In that situation we were able to fix a single value at each stage of the construction and prevent the martingale exceeding this value. For the case of computable randomness, we cannot do this because the martingale may win very slowly. However, we can construct a suitable martingale $\widehat{d}$ by restraining the betting the first time a martingale exceeds $2^n$, for any $n$, along any path.

**Proposition 2.10.** *Let $d$ be a computable martingale. There exists a computable martingale $\widehat{d}$ and $m$, a selection function for $\widehat{d}$, such that for all $\alpha \in 2^\omega$, if $d$ succeeds on $\alpha$ then $\forall c\, \exists n\, \widehat{d}(\alpha \upharpoonright m(n)) = 2^c$.*

**Proof.** We construct $\widehat{d}$ and $m$ as follows. At stage 0, define $m(0) = 0$ and $\widehat{d}(\lambda) = 1$. At stage $s+1$, define

$$m(s+1) = r(m(s)) + s + 2.$$

For all $\sigma \in 2^{<\omega}$ such that $|\sigma| = m(s)$, let $n_\sigma = \min\{n : \text{for all } \tau \preceq \sigma, 2^n > \widehat{d}(\tau)\}$. We inductively define $\widehat{d}(\sigma' i)$ for all $i \in \{0,1\}$ and all $\sigma' \succeq \sigma$ with $|\sigma'| < m(s+1)$ as follows.

$$\widehat{d}(\sigma' i) = \begin{cases} 2^{n_\sigma} & \text{if } \widehat{d}(\sigma') \cdot \frac{d(\sigma' i)}{d(\sigma')} \geq 2^{n_\sigma} \\ 2\widehat{d}(\sigma') - 2^{n_\sigma} & \text{if } \widehat{d}(\sigma') \cdot \frac{d(\sigma'(1-i))}{d(\sigma')} \geq 2^{n_\sigma} \\ \widehat{d}(\sigma') \cdot \frac{d(\sigma' i)}{d(\sigma')} & \text{otherwise.} \end{cases}$$

Observe that $\widehat{d}$ is a martingale. This can be proved by induction over the stages of the construction and over the length of $\sigma'$ at each stage. The martingale $\widehat{d}$ is attempting to follow the betting strategy of $d$ above $\sigma$.

To verify the construction we observe that if $d$ succeeds on $\alpha$, then the construction of $\widehat{d}$ will enforce that for all $c$ there exists an $n$, $\widehat{d}(\alpha \upharpoonright m(n)) = 2^c$. Otherwise, we would have that both $\limsup \widehat{d}(\alpha \upharpoonright n)$ is bounded above and that $\widehat{d}(\alpha \upharpoonright n) \geq a \cdot d(\alpha \upharpoonright n)$ for some positive rational $a$. $\square$

**Corollary 2.11.** *If $\alpha$ is not computably random, there exists a quick process machine $P$ such that for all $c$, there exists some $n$ with $C^P(\alpha \upharpoonright n) = n - c$.*

**Proof.** Combine Propositions 2.6 and 2.10. □

We are now able to provide characterisations of computable randomness, Schnorr randomness and weak randomness in terms of quick process machines. First we establish the following lemma.

**Lemma 2.12.** *If $m$ is a strictly increasing computable function, $d$ a computable martingale and $\alpha \in 2^\omega$ such that for all $n$, $d(\alpha \upharpoonright m(n)) \geq 2^n$, then $\alpha$ is not weakly random.*

**Proof.** Construct a computable martingale $\widehat{d}$ from $d$ by following the betting strategy of $d$ except that whenever $\widehat{d}$'s unbanked capital exceeds 2 along some path, $\widehat{d}$ banks half its unbanked capital and only bets with the remaining half. This ensures that the amount of unbanked capital that $\widehat{d}$ has along any path never exceeds 4. This means that at $\alpha \upharpoonright m(n)$, the martingale $\widehat{d}$ must have banked capital at least $n - 2$ times and so $\widehat{d}(\alpha \upharpoonright m(n)) \geq n - 2$. □

**Theorem 2.13.** (1) *A real $\alpha$ is not computably random if and only if there exists a quick process machine $P$, such that*

$$\forall c\, \exists n\, C^P(\alpha \upharpoonright n) \leq n - c.$$

(2) *A real $\alpha$ is not Schnorr random if and only if there exists a quick process machine $P$, and a strictly increasing computable function $m$, such that*

$$\exists^\infty n\, C^P(\alpha \upharpoonright m(n)) \leq m(n) - n.$$

(3) *A real $\alpha$ is not weakly random if and only if there exists a quick process machine $P$, and a strictly increasing computable function $m$ such that*

$$\forall n\, C^P(\alpha \upharpoonright m(n)) \leq m(n) - n.$$

**Proof.** The left to right directions are just Corollaries 2.9 and 2.11 restated. For the right to left direction, given such a quick process machine $P$, we use Proposition 2.1 to build a martingale $d$. Now $d(\alpha \upharpoonright n) \geq 2^{n - C^P(\alpha|n)}$. Hence if $\forall c\, \exists n\, C^P(\alpha \upharpoonright n) \leq n - c$, we have that $\alpha$ is not computably random.

If for some strictly increasing computable $m$, $\exists^\infty n\, C^P(\alpha \upharpoonright m(n)) \leq m(n) - n$, then let $h(x) = \max\{\{0\} \cup \{2^n : m(n) \leq x\}\}$. For infinitely many $n$, $d(\alpha \upharpoonright m(n)) \geq 2^n = h(m(n))$ and so $\alpha$ is not Schnorr random.

Finally assume for some strictly increasing computable $m$, $\forall n C^P(\alpha \upharpoonright m(n)) \leq m(n) - n$. In this case, for all $n$, $d(\alpha \upharpoonright m(n)) \geq 2^n$ and so by Lemma 2.12, $\alpha$ is not weakly random. □

## 3. Quick process machines and truth-table functionals

There is a simple technique to translate between truth-table functionals and quick process machines that allows further characterisations of computable randomness, Schnorr randomness and weak randomness in terms of truth-table reducibility. However, for these characterisations to hold, we need to be careful about how we define the use of a truth-table reduction. Nerode observed that a truth-table reduction can be regarded as a Turing reduction that is total on all oracles [15]. This result is also attributed to Trakhtenbrot [17]. We will take this as our definition of a truth-table reduction. Given a truth-table reduction $\Phi$ we now define $\phi^\alpha(n)$ to be the largest query made of the oracle $\alpha$ during the computation of $\Phi^\alpha(m)$ for any $m \leq n$.

**Theorem 3.1.** (1) *A real $\alpha$ is not computably random if and only if there exists a truth-table functional $\Phi$, and $\beta \in 2^\omega$, such that $\Phi^\beta = \alpha$ and $\forall c\, \exists n\, \phi^\beta(n) \leq n - c$.*

(2) *A real $\alpha$ is not Schnorr random if and only if there exists a truth-table functional $\Phi$, $\beta \in 2^\omega$, and a strictly increasing computable function $m$ such that $\Phi^\beta = \alpha$ and $\exists^\infty n\, \phi^\beta(m(n)) \leq m(n) - n$.*

(3) *A real $\alpha$ is not weakly random if and only if there exists a truth-table functional $\Phi$, $\beta \in 2^\omega$, and a strictly increasing computable function $m$ such that $\Phi^\beta = \alpha$ and $\forall n\, \phi^\beta(m(n)) \leq m(n) - n$.*

**Proof.** The right to left direction of the above statements can be established by constructing a martingale $d$ from a truth-table functional $\Phi$ as follows. Let $d(\sigma) = \mu\{\alpha : \Phi^\alpha \succeq \sigma\} \cdot 2^{|\sigma|}$. The fact that $\Phi$ is total on all oracles makes $d$ computable. Now if $\Phi^\beta = \alpha$, then we have that $d(\alpha \upharpoonright n) \geq 2^{n - \phi^\beta(n)}$. The right to left direction for (1) and (2) follow immediately. For (3) an application of Lemma 2.12 is needed.

To establish the left to right direction, given a quick process machine $P$, we define a truth-table functional $\Phi$ that computes $\Phi^\alpha(n)$ by finding the shortest initial segment of $\tau$ of $\alpha$ such that $|P(\tau)| > n$ and setting $\Phi^\alpha(n)$ to be the $(n+1)$th bit of this output.

Now if $\alpha$ is not computably random, then by Corollary 2.11, there is some quick process machine $P$ such that $\forall c \exists n$ $C^P(\alpha \upharpoonright n) = n - c$. Further by applying Lemma 2.7, for all $c$, we can take some $\tau_c$ with $P(\tau_c) = \alpha \upharpoonright (|\tau_c| + c)$ and that $\{\tau_c : c \in \mathbb{N}\}$ is a chain. Let $\beta = \bigcup_c \tau_c$. Thus $\Phi^\beta = \alpha$ and $\phi^\beta(|P(\tau_c)| - 1) \leq |P(\tau_c)| - c$. Similarly if $\alpha$ is Schnorr random or weakly random. □

The author would like to note that this characterisation of computable randomness in terms of truth-table reducibility has been independently arrived at by Bienvenu and Porter [18].

## Acknowledgments

The author thanks his PhD supervisors, Rod Downey and Noam Greenberg for helpful discussions on this topic. The author also thanks the anonymous referees for their helpful comments.

## References

[1] L. Bienvenu, C. Porter, Strong reductions in effective randomness, Preprint.
[2] G.J. Chaitin, On the length of programs for computing finite binary sequences, J. ACM 13 (1966) 547–569.
[3] A.R. Day, On process complexity, Chic. J. Theoret. Comput. Sci. (2010) 13. Article 4.
[4] O. Demuth, Remarks on the structure of tt-degrees based on constructive measure theory, Comment. Math. Univ. Carolin. 29 (2) (1988) 233–247.
[5] R.G. Downey, D.R. Hirschfeldt, Algorithmic Randomness and Complexity, Springer-Verlag, 2010.
[6] R.G. Downey, E.J. Griffiths, Schnorr randomness, J. Symbolic Logic 69 (2) (2004) 533–554.
[7] R.G. Downey, E.J. Griffiths, S. Reid, On Kurtz randomness, Theoret. Comput. Sci. 321 (2-3) (2004) 249–270.
[8] A.N. Kolmogorov, Three approaches to the quantitiative definition of information, Prob. Inf. Trans. 1 (1965) 1–7.
[9] S. Kurtz, Randomness and genericity in the degrees of unsolvability, Ph.D. Thesis, University of Illinois at Urbana–Champaign, 1981.
[10] L.A. Levin, On the notion of a random sequence, Sov. Math. Dokl. 14 (5) (1973) 1413–1416.
[11] L.A. Levin, A.K. Zvonkin, The complexity of finite objects and the development of the concepts of information and randomness of means of the theory of algorithms, Russian Math. Surveys 25 (6) (1970) 83–124.
[12] M. Li, P. Vitányi, An Introduction to Kolmogorov Complexity and its Applications, second ed., Springer-Verlag New York, Inc, Secaucus, NJ, USA, 1997.
[13] J.H. Lutz, The quantitative structure of exponential time, in: L.A. Hemaspaandra, A.L. Selman (Eds.), In Complexity Theory Retrospective II, Springer, 1997, pp. 225–260.
[14] P. Martin-Löf, The definition of random sequences, Inform. and Control 9 (1966) 602–619.
[15] A. Nerode, General topology and partial recursive functions, in: Summaries of talks presented at the Summer Institute for Symbolic Logic, Cornell University, 1957, pp. 247–251.
[16] A. Nies, Computability and Randomness, Oxford University Press, 2009.
[17] P. Odifreddi, Classical Recursion Theory. The Theory of Functions and Sets of Natural Numbers, in: Studies in Logic and the Foundations of Mathematics, vol. 125, North-Holland Publishing Company, Amsterdam, 1990.
[18] C. Porter, Mathematical and philosophical perspectives on algorithmic randomness, Ph.D. Thesis, University of Notre Dame, 2012.
[19] C.P. Schnorr, A unified approach to the definition of random sequences, Theory of Computing Systems 5 (3) (1971) 246–258.
[20] C.P. Schnorr, Zufälligkeit und wahrscheinlichkeit, in: Lecture Notes in Mathematics, vol. 218, Springer-Verlag, Berlin-New York, 1971.
[21] C.P. Schnorr, Process complexity and effective random test, J. Comput. System Sci. 7 (1973) 376–388.
[22] R.J. Solomonoff, A formal theory of inductive inference, Information and Control 7 (1964) 224–254.
[23] Y. Wang, Randomness and complexity, Ph.D. Thesis, University of Heidelberg, 1996.