

Feasible Depth

David Doty*

Philippe Moser†

Abstract

This paper introduces two complexity-theoretic formulations of Bennett’s logical depth: *finite-state depth* and *polynomial-time depth*. It is shown that for both formulations, trivial and random infinite sequences are shallow, and a *slow growth law* holds, implying that deep sequences cannot be created easily from shallow sequences. Furthermore, the E analogue of the halting language is shown to be polynomial-time deep, by proving a more general result: every *linear-time weakly useful for E* language (a language to which a nonnegligible subset of E can be reduced in uniform linear time) is polynomial-time deep.

1 Introduction

Whereas many structures found in nature are highly complex (a DNA sequence, a cell), some seem much simpler, either because of their complete regularity (ice), or their complete randomness (gas). Bennett introduced logical depth [3] to formalize computationally the difference between complex and non-complex (trivial or random) structures. Briefly, a logically deep object is one with a shorter description than itself, but which requires a long time to compute from this short description.

Depth is not a measure of information contained in an object, which correlates with *randomness*, but rather its value, or its *useful* information content. According to classical [17] or algorithmic information theory [13], the information content of a sequence is not representative of its value. Consider an infinite binary sequence produced by random coin tosses. Although the sequence contains a large amount of information in the sense that, with probability 1, it cannot be significantly compressed, its information is not of much value, except as a source of input to randomized algorithms. Contrast this with the characteristic sequence of the halting language, access to which enables any computably enumerable language to be decided in linear time. From this perspective, the halting sequence is much more useful than a randomly generated sequence.

Bennett’s logical depth separates the sequences that are deep (i.e., that show high internal organization) from those that are shallow (i.e., not deep). Informally, deep sequences are those which contain redundancy, but in such a way that an algorithm requires extensive resources to exploit the redundancy (for instance, to compress or to predict the sequence). In other words, deep sequences are organized, but in a nontrivial way. Highly redundant sequences like 00000... are shallow, because they are trivially organized. Random sequences are shallow,

*Department of Computer Science, Iowa State University, Ames, IA 50011 USA. ddoty (at) iastate (dot) edu.

†Dept de Informàtica e Ingenieria de Sistemas, Centro Politécnico Superior, Zaragoza, Spain. mosersan (at) gmail (dot) com. This author was partially supported by subvenciones para grupos de investigación Gobierno de Aragón UZ-T27 and subvenciones de fomento de movilidad Gobierno de Aragón MI31/2005.

because they are completely unorganized. One of the key features of Bennett’s logical depth is that it obeys a *slow growth law* [3, 11]: no fast process can transform a shallow sequence into a deep one. Therefore a deep object can be created only through a complex, time-consuming process.

Bennett [3] showed that the halting language is deep, arguing that its depth was evidence of its usefulness. Juedes, Lathrop, and Lutz [11] generalized this result and solidified the connection between usefulness and depth by proving that every *weakly useful* language [8] is deep, where a weakly useful language is one to which a nonnegligible subset of the decidable languages (in the sense of resource-bounded measure theory [14]) reduce in a fixed computable time bound.

Unfortunately, because it is based on Kolmogorov complexity, Bennett’s logical depth is not computable. Antunes, Fortnow, van Melkebeek, and Vinodchandran [1] investigated several polynomial-time formulations of depth as instances of the more general concept of computational depth obtained by considering the difference between variants of Kolmogorov complexity. Deep and intriguing connections were demonstrated between depth and average-case complexity, nonuniform circuit complexity, and efficient search for satisfying assignments to Boolean formulas. Nevertheless, some of the depth notions in [1] require complexity assumptions to prove the existence of deep sequences, and not all the depth notions obey slow growth laws. Furthermore, [1] lacks a polynomial-time analogue of the Juedes-Lathrop-Lutz theorem demonstrating that useful objects are necessarily deep.

The aim of this paper is to propose a feasible depth notion that satisfies a slow growth law and in which deep sequences can be proven to exist. We propose two such notions: finite-state depth, and polynomial-time depth. Furthermore, we connect polynomial-time depth to usefulness in deciding languages in the complexity class E . In both cases, the definition of depth intuitively reflects that of Bennett’s logical depth: a sequence is deep if it is redundant, but an algorithm requires extensive resources in order to exploit the redundancy.

Our formulation of finite-state depth is based on the classical model of finite-state compressors and decompressors introduced by Shannon [17] and investigated by Huffman [10] and Ziv and Lempel [19]. Informally, a sequence is finite-state deep if given more states, a finite-state machine can decompress the sequence from an input significantly shorter than is possible with fewer states. We show that both finite-state trivial sequences (i.e sequences with finite-state strong dimension [2] equal to zero) and finite-state random sequences (those with finite-state dimension [6] equal to 1, or equivalently normal sequences [4]) are shallow. Our main result in this section shows that finite-state depth obeys a slow growth law: no information lossless finite-state transducer can transform a finite-state shallow sequence into a finite-state deep sequence. We conclude the section by proving the existence of finite-state deep sequences.

Our formulation of polynomial-time depth – contrary to finite-state depth – is not based on compression algorithms but on polynomial-time computable predictors. Given a language L , a polynomial-time predictor is a polynomial-time computable function that, given an input string x , predicts the probability that $x \in L$. Informally, L is polynomial-time deep if, given more time, a predictor is better able to predict membership of strings in L . We show that both E -trivial languages (languages in the complexity class E) and E -random languages are polynomial-time shallow. Our main results in this section are a slow growth law – with respect to linear time many-one reductions whose inverse is linear time computable – similar to that for finite-state depth and logical depth, and a theorem stating that any language which is “useful” for quickly deciding languages in E must be polynomial-time deep. More

precisely, every language that is linear-time weakly useful for E – every language to which a nonnegligible (in the sense of resource-bounded measure [14]) subset of E can be reduced in a fixed linear time bound – is polynomial-time deep. It follows that H_E , the E version of the halting language, is polynomial-time deep.

2 Preliminaries

\mathbb{N} is the set of all nonnegative integers. A (*finite*) *string* is an element of $\{0, 1\}^*$. An (*infinite*) *sequence* is an element of the Cantor space $\{0, 1\}^\infty$. For a string or sequence S and $n \in \mathbb{N}$, $S \upharpoonright n$ denotes the prefix consisting of the first n bits of S . For a string w and a string or sequence S , we write $w \sqsubseteq S$ to denote that $w = S \upharpoonright n$ for some $n \in \mathbb{N}$. For a string x , its length is denoted by $|x|$. $s_0, s_1, s_2 \dots$ denotes the standard enumeration of the strings in $\{0, 1\}^*$ in lexicographical order, where $s_0 = \lambda$ denotes the empty string. If x, y are strings, we write $x < y$ if $|x| < |y|$ or $|x| = |y|$ and x precedes y in alphabetical order.

A *language* is a subset of $\{0, 1\}^*$. A *class* is a set of languages. The *characteristic sequence* of a language L is the sequence $\chi_L \in \{0, 1\}^\infty$, whose n^{th} bit is 1 if and only if $s_n \in L$. Because $L \mapsto \chi_L$ is a bijection, we will often speak of languages and sequences interchangeably, with it understood that the “sequence” L refers to χ_L , and the “language” χ_L refers to L . For $n \in \mathbb{N}$, we write $L \upharpoonright n$ to denote $\chi_L \upharpoonright n$. We denote by $E = \bigcup_{c \in \mathbb{N}} \text{DTIME}(2^{cn})$ and $\text{EXP} = \bigcup_{c \in \mathbb{N}} \text{DTIME}(2^{n^c})$.

Let $i \leq j \in \mathbb{N}$. The i^{th} projection function $\text{proj}_i : (\{0, 1\}^*)^j \rightarrow \{0, 1\}^*$, is given by $\text{proj}_i(x_1, \dots, x_j) = x_i$.

3 Finite-State Depth

3.1 Finite-State Compression

We use a model of finite-state compressors and decompressors based on finite-state transducers, which was introduced in a similar form by Shannon [17] and investigated by Huffman [10] and Ziv and Lempel [19]. Kohavi [12] gives an extensive treatment of the subject.

A *finite-state transducer (FST)* is a 4-tuple $T = (Q, \delta, \nu, q_0)$, where

- Q is a nonempty, finite set of *states*,
- $\delta : Q \times \{0, 1\} \rightarrow Q$ is the *transition function*,
- $\nu : Q \times \{0, 1\} \rightarrow \{0, 1\}^*$ is the *output function*,
- $q_0 \in Q$ is the *initial state*.

Furthermore, we assume that every state in Q is reachable from q_0 .

For all $x \in \{0, 1\}^*$ and $a \in \{0, 1\}$, define the *extended transition function* $\widehat{\delta} : \{0, 1\}^* \rightarrow Q$ by the recursion $\widehat{\delta}(\lambda) = q_0$, and $\widehat{\delta}(xa) = \delta(\widehat{\delta}(x), a)$. For $x \in \{0, 1\}^*$, we define the *output* of T on x to be the string $T(x)$ defined by the recursion $T(\lambda) = \lambda$, and $T(xa) = T(x)\nu(\widehat{\delta}(x), a)$ for all $x \in \{0, 1\}^*$ and $a \in \{0, 1\}$.

A FST can trivially act as an “optimal compressor” by outputting λ on every transition arrow, but this is, of course, a useless compressor, because the input cannot be recovered. A FST $T = (Q, \delta, \nu, q_0)$ is *information lossless (IL)* if the function $x \mapsto (T(x), \widehat{\delta}(x))$ is one-to-one; i.e., if the output and final state of T on input $x \in \{0, 1\}^*$ uniquely identify x . An

information lossless finite-state transducer (ILFST) is a FST that is IL. We write FST to denote the set of all finite-state transducers, and we write ILFST to denote the set of all information lossless finite-state transducers. We say $f : \{0, 1\}^\infty \rightarrow \{0, 1\}^\infty$ is *FS computable* (resp. *ILFS computable*) if there is a FST (resp. ILFST) T such that, for all $S \in \{0, 1\}^\infty$, $\lim_{n \rightarrow \infty} |T(S \upharpoonright n)| = \infty$ and, for all $n \in \mathbb{N}$, $T(S \upharpoonright n) \subseteq f(S)$. In this case, define $T(S) = f(S)$.

The following well-known theorem [10, 12] states that the function from $\{0, 1\}^*$ to $\{0, 1\}^*$ computed by an ILFST can be inverted – in an approximate sense – by another ILFST.

Theorem 3.1. *For any ILFST T , there exists an ILFST T^{-1} and a constant $c \in \mathbb{N}$ such that, for all $x \in \{0, 1\}^*$, $x \upharpoonright (|x| - c) \subseteq T^{-1}(T(x)) \subseteq x$.*

Corollary 3.2. *For any ILFST T , there exists an ILFST T^{-1} such that, for all sequences S , $T^{-1}(T(S)) = S$.*

Fix some standard binary representation $\sigma_T \in \{0, 1\}^*$ of each FST T , and define $|T| = |\sigma_T|$. For all $k \in \mathbb{N}$, define

$$\begin{aligned} \text{FST}^{\leq k} &= \{T \in \text{FST} : |T| \leq k\}, \\ \text{ILFST}^{\leq k} &= \{T \in \text{ILFST} : |T| \leq k\} \end{aligned}$$

Let $k \in \mathbb{N}$ and $x \in \{0, 1\}^*$. The k -FS *decompression complexity* (or when k is clear from context, *FS complexity*) of x is

$$D_{\text{FS}}^k(x) = \min_{p \in \{0, 1\}^*} \left\{ |p| \mid (\exists T \in \text{FST}^{\leq k}) T(p) = x \right\},$$

i.e., the size of the smallest program $p \in \{0, 1\}^*$ such that some k -bit FST outputs x on input p .

For a fixed k , D_{FS}^k is a finite state analogue of Kolmogorov complexity. For any sequence S , define the *finite-state dimension* of S by

$$\dim_{\text{FS}}(S) = \lim_{k \rightarrow \infty} \liminf_{n \rightarrow \infty} \frac{D_{\text{FS}}^k(S \upharpoonright n)}{n}, \quad (3.1)$$

and the *finite-state strong dimension* of S by

$$\text{Dim}_{\text{FS}}(S) = \lim_{k \rightarrow \infty} \limsup_{n \rightarrow \infty} \frac{D_{\text{FS}}^k(S \upharpoonright n)}{n}. \quad (3.2)$$

Finite-state dimension and strong dimension measure the degree of finite-state randomness of a sequence. The above definitions are equivalent [18, 7] to several other definitions of finite-state dimension and strong dimension in terms of finite-state gamblers [6, 2], entropy rates [19, 5], information lossless finite-state compressors [19, 6, 2], and finite-state log-loss predictors [9].

Schnorr and Stimm [16] (and more explicitly, Bourke, Hitchcock, and Vinodchandran [5]) showed that a sequence has finite-state dimension 1 if and only if it is *normal* in the sense of Borel [4], meaning that for all $k \in \mathbb{N}$, every substring of length k occurs in S with limiting frequency 2^{-k} .

The next two lemmas show that ILFST's cannot alter the FS complexity of a string by very much.

Lemma 3.3. *Let M be an ILFST. Then*

$$(\exists c_1 \in \mathbb{N})(\forall k \in \mathbb{N})(\forall x \in \{0, 1\}^*) D_{\text{FS}}^{k+c_1}(M(x)) \leq D_{\text{FS}}^k(x).$$

Proof. The proof idea of the lemma is the following. Let k, x be as above, let p be a k -minimal program for x , i.e. $A(p) = x$ where $A \in \text{FST}^{\leq k}$, and $D_{\text{FS}}^k(x) = |p|$. We construct A' and p' for $M(x)$. Let $p' = p$ and let A' be the automata which on input p' simulates $A(p)$, and plugs the output into M . The size of A' is roughly the size of A plus the size of M , i.e. $D_{\text{FS}}^{k+c_1}(M(x)) \leq D_{\text{FS}}^k(x)$, for some constant c_1 . More formally, let

$$\delta_A : Q_A \times \{0, 1\} \rightarrow Q_A$$

be the transition function of A , with

$$Q_C = \{(q_i, s_i) \mid 1 \leq i \leq t_C\} \subset (\mathcal{P}(\{0, 1\}^*))^2 \quad C \in \{A, M\}$$

where $q_i \in \{0, 1\}^*$ are the states and $s_i \in \{0, 1\}^*$ are the corresponding outputs strings, and let $\delta_M : Q_M \times \{0, 1\} \rightarrow Q_M$ be the transition function for M . We construct $\delta' : Q' \times \{0, 1\} \rightarrow Q'$ for A' . Let

$$Q' = Q_A \times Q_M \times \{A, M\} \times \{0, 1\}^{\leq t} \times \{0, 1\}^{\leq t}$$

where t is a constant depending on A and M . Let $(q_A, s_A) \in Q_A$, $(q_M, s_M) \in Q_M$, $s, m \in \{0, 1\}^{\leq t}$ and $b \in \{0, 1\}$. Define

$$\begin{cases} \delta'((q_A, s_A), (q_M, s_M), A, s, m, b) \\ \quad = (\delta_A((q_A, s_A), b), (q_M, s_M), M, \lambda, \text{proj}_2(\delta_A((q_A, s_A), b))) \\ \delta'((q_A, s_A), (q_M, s_M), M, s, m, b) \\ \quad = ((q_A, s_A), \delta_M((q_M, s_M), m), A, \text{proj}_2(\delta_M((q_M, s_M), m)), \lambda). \end{cases}$$

□

Lemma 3.4. *Let M be an ILFST. Then*

$$(\exists c_2 \in \mathbb{N})(\forall k \in \mathbb{N})(\forall x \in \{0, 1\}^*) D_{\text{FS}}^{k+c_2}(x) \leq D_{\text{FS}}^k(M(x)).$$

Proof. The proof is similar to Lemma 3.3. Let k, x be as above. Because of Theorem 3.1, there exists an ILFST M^{-1} and a constant b such that for any string x , $x \upharpoonright |x| - b \sqsubseteq M^{-1}(M(x)) \sqsubseteq x$.

Let p be a k -minimal program for $M(x)$, i.e. $A(p) = M(x)$ where $A \in \text{FST}^{\leq k}$, and $D_{\text{FS}}^k(M(x)) = |p|$. We construct A' and p' for x . Let $y = M^{-1}(M(x))$, i.e. $yz = x$ and $|z| \leq b$. Let $p' = p$ and let A' be the automata which on input p' simulates $A(p)$, plugs the output into M^{-1} and adds z at the end of M^{-1} 's output. The size of A' is roughly the size of A plus the size of M plus the size of z (which is of size at most b), i.e. $D_{\text{FS}}^{k+c_2}(M(x)) \leq D_{\text{FS}}^k(x)$, for some constant c_2 . □

3.2 Finite-State Depth

A sequence is finite-state deep if given more states, a finite state machine can decompress the sequence from a significantly shorter input.

Definition 3.5. A sequence S is *finite-state deep* if

$$(\exists \alpha > 0)(\forall k \in \mathbb{N})(\exists k' \in \mathbb{N})(\exists^\infty n \in \mathbb{N}) \ D_{\text{FS}}^k(S \upharpoonright n) - D_{\text{FS}}^{k'}(S \upharpoonright n) \geq \alpha n.$$

A sequence S is *finite-state shallow* if it is not finite-state deep.

Remark. All theorems in this section remain true if the quantification in the definition of finite-state depth is changed to

$$(\forall k \in \mathbb{N})(\exists \alpha > 0)(\exists k' \in \mathbb{N})(\exists^\infty n \in \mathbb{N}) \ D_{\text{FS}}^k(S \upharpoonright n) - D_{\text{FS}}^{k'}(S \upharpoonright n) \geq \alpha n.$$

Note that any sequence deep by the former definition must be deep by the latter definition.

The following result shows that finite-state trivial sequences are shallow.

Theorem 3.6. Let S be a sequence with $\text{Dim}_{\text{FS}}(S) = 0$. Then S is finite-state shallow.

Proof. Let S be as above and $\alpha > 0$. By (3.2) let k be such that

$$\limsup_{n \rightarrow \infty} \frac{D_{\text{FS}}^k(S \upharpoonright n)}{n} < \alpha,$$

i.e. $(\forall^\infty n \in \mathbb{N}) \ D_{\text{FS}}^k(S \upharpoonright n) < \alpha n$. Therefore for any $k' \in \mathbb{N}$

$$(\forall^\infty n \in \mathbb{N}) \ D_{\text{FS}}^k(S \upharpoonright n) - D_{\text{FS}}^{k'}(S \upharpoonright n) \leq D_{\text{FS}}^k(S \upharpoonright n) < \alpha n.$$

Since α is arbitrary, S is finite-state shallow. □

The following result shows that finite-state random sequences (i.e., sequences S with $\text{dim}_{\text{FS}}(S) = 1$, or normal sequences) are shallow.

Theorem 3.7. Every normal sequence is finite-state shallow.

Proof. Let S be normal, $k \in \mathbb{N}$ and $\alpha > 0$. Because normal sequences have finite-state dimension 1,

$$(\forall k' \in \mathbb{N})(\forall^\infty n \in \mathbb{N}) \ D_{\text{FS}}^{k'}(S \upharpoonright n) > \left(1 - \frac{\alpha}{2}\right) n.$$

Thus

$$(\forall k' \in \mathbb{N})(\forall^\infty n \in \mathbb{N}) \ D_{\text{FS}}^k(S \upharpoonright n) - D_{\text{FS}}^{k'}(S \upharpoonright n) < n - \left(1 - \frac{\alpha}{2}\right) n = \frac{\alpha}{2} n < \alpha n.$$

Because α is arbitrary, S is finite-state shallow. □

Finite-state deep sequences cannot be created easily, as the following theorem shows. More precisely, no ILFST can transform a finite-state shallow sequence into a finite-state deep sequence.

Theorem 3.8 (Slow growth law). Let S be any sequence, let $f : \{0, 1\}^\infty \rightarrow \{0, 1\}^\infty$ be ILFS computable, and let $S' = f(S)$. If S' is finite-state deep, then S is finite-state deep.

Proof. Let S, S', f be as above and M be an ILFST computing f .

Because S' is finite-state deep,

$$(\exists \alpha > 0)(\forall k \in \mathbb{N})(\exists k' \in \mathbb{N})(\exists^\infty n \in \mathbb{N}) D_{\text{FS}}^k(S' \upharpoonright n) - D_{\text{FS}}^{k'}(S' \upharpoonright n) \geq \alpha n. \quad (3.3)$$

Let $l \in \mathbb{N}$ and let $c = \max\{c_1, c_2\}$ where c_1, c_2 are the two constants in Lemmas 3.3 and 3.4. Let $l' = k' + c$ where k' is obtained from (3.3) with $k = l + c$. For all $n \in \mathbb{N}$, denote by m_n the smallest integer such that $M(S \upharpoonright m_n) = S' \upharpoonright n$. Because M is IL, it cannot visit a state twice without outputting at least one bit, so there exists a constant $\beta > 0$ such that, for all $n \in \mathbb{N}$, $n \geq \beta m_n$. For infinitely many $n \in \mathbb{N}$,

$$\begin{aligned} & D_{\text{FS}}^l(S \upharpoonright m_n) - D_{\text{FS}}^{l'}(S \upharpoonright m_n) \\ &= D_{\text{FS}}^l(S \upharpoonright m_n) - D_{\text{FS}}^{k'+c}(S \upharpoonright m_n) && l' = k' + c \\ &\geq D_{\text{FS}}^l(S \upharpoonright m_n) - D_{\text{FS}}^{k'}(M(S \upharpoonright m_n)) && \text{Lemma 3.4} \\ &= D_{\text{FS}}^{k-c}(S \upharpoonright m_n) - D_{\text{FS}}^{k'}(M(S \upharpoonright m_n)) && k = l + c \\ &\geq D_{\text{FS}}^k(M(S \upharpoonright m_n)) - D_{\text{FS}}^{k'}(M(S \upharpoonright m_n)) && \text{Lemma 3.3} \\ &= D_{\text{FS}}^k(S' \upharpoonright n) - D_{\text{FS}}^{k'}(S' \upharpoonright n) && \text{definition of } m_n \\ &\geq \alpha n && \text{by (3.3)} \\ &\geq \alpha \beta m_n, && \text{because } M \text{ is IL} \end{aligned}$$

whence S is finite-state deep. \square

We next prove the existence of finite-state deep sequences. We require two technical lemmas first, which place bounds on the FS complexity of two concatenated strings.

Lemma 3.9. $(\forall l \in \mathbb{N})(\forall x, y \in \{0, 1\}^*) D_{\text{FS}}^l(xy) \geq D_{\text{FS}}^l(x) + D_{\text{FS}}^l(y) - 2^l$.

Proof. Let l, x, y be as above and suppose $D_{\text{FS}}^l(xy) = |pp'|$ where $T \in \text{FST}^{\leq l}$, $p, p' \in \{0, 1\}^*$, with $T(pp') = xy$, and $T(p) = x$. Thus $D_{\text{FS}}^l(x) \leq |p|$ and there exists $s \in \{0, 1\}^{\leq 2^l}$ (because T has less than 2^l states) such that $T(sp') = y$, i.e. $D_{\text{FS}}^l(y) \leq |p'| + 2^l$. Therefore

$$D_{\text{FS}}^l(xy) = |p| + |p'| \geq D_{\text{FS}}^l(x) + D_{\text{FS}}^l(y) - 2^l,$$

which proves the lemma. \square

Lemma 3.10. $(\exists c \in \mathbb{N})(\forall l \in \mathbb{N})(\forall x, y \in \{0, 1\}^*) D_{\text{FS}}^{l+c}(xy) \leq 2|x| + D_{\text{FS}}^l(y) + 2$.

Proof. Let l, x, y be as above and let p be a minimal program for y , i.e. $D_{\text{FS}}^l(y) = |p|$ where $A(p) = y$ with $p \in \{0, 1\}^*$ and $A \in \text{FST}^{\leq l}$. Let $p' = x'01p$ where x' is x with every bit doubled and let $A' \in \text{FST}^{\leq l+c}$ where c is a constant independent of l be the following FST for xy : $A'(p')$ uses $d(x)$ to output x , then upon reading 01 , it outputs $A(p)$. \square

Theorem 3.11. *There exists a finite-state deep sequence.*

Proof. For all $r \in \{0, 1\}^*$, define the FST $T_r = (\{q_0\}, \delta, \nu, q_0)$, where, for $b \in \{0, 1\}$, $\delta(q_0, b) = q_0$ and $\nu(q_0, b) = r$. Define the constant $c' = |T_r| - |r|$ (i.e., the number of extra bits beyond r required to describe T_r ; note that this is a constant independent of r).

We construct the finite-state deep sequence $S = S_1 S_2 \dots$ in stages, with $S_i \in \{0, 1\}^*$ for all $i \in \mathbb{N}$. Let $\phi : \mathbb{N} \rightarrow \mathbb{N} \times \mathbb{N}$ be a function such that $(\forall k \in \mathbb{N})(\exists^\infty j \in \mathbb{N}) \phi(j) = (k, 2^{2^{k+1}})$, and

for all $i \in \mathbb{N}$, $\text{proj}_2(\phi(i)) = 2^{2^{\text{proj}_1(\phi(i))+1}}$. Let $j \in \mathbb{N}$ and suppose the prefix $S_1 S_2 \dots S_{j-1}$ has already been constructed. Let $t_{j-1} = |S_1 S_2 \dots S_{j-1}|$. Let $(k, k') = \phi(j)$, so that $k' = 2^{2^{k+1}}$.

Intuitively, at stage j , we will diagonalize against k -bit FST's to make $D_{\text{FS}}^k(S_1 \dots S_j)$ large, while helping a particular $(k' + c)$ -bit FST (c the constant from Lemma 3.10) so that $D_{\text{FS}}^{k'+c}(S_1 \dots S_j)$ is small.

Let $r_j \in \{0, 1\}^{k'-c'}$ be k -FS-random in the sense that

$$D_{\text{FS}}^k(r_j) \geq |r_j| - 2^{k/2}. \quad (3.4)$$

Note that such a string always exists because there are at most $|\text{FST}^{\leq k}| \cdot 2^{|r_j| - 2^{k/2}} < 2^{|r_j|}$ strings contradicting (3.4). Let $u_j = 12t_{j-1}$. Let $S_j = r_j^{u_j/|r_j|}$ be $u_j/|r_j|$ consecutive copies of r_j . Let $T = T_{r_j}$ as described above. Then $|T| = k'$. It is clear that T outputs $S_j = r_j^{u_j/|r_j|}$ on any input program of length $u_j/|r_j|$. Therefore $D_{\text{FS}}^{k'}(S_j) \leq u_j/|r_j|$. Lemma 3.10 implies that

$$D_{\text{FS}}^{k'+c}(S_1 \dots S_j) \leq 2|S_1 \dots S_{j-1}| + D_{\text{FS}}^{k'}(S_j) + 2,$$

whence

$$D_{\text{FS}}^{k'+c}(S_1 \dots S_j) \leq 2t_{j-1} + \frac{u_j}{|r_j|} + 2. \quad (3.5)$$

Note that

$$\begin{aligned} D_{\text{FS}}^k(S_1 \dots S_j) &\geq D_{\text{FS}}^k(S_1 \dots S_{j-1}) + D_{\text{FS}}^k(S_j) - 2^k && \text{Lemma 3.9} \\ &\geq D_{\text{FS}}^k(S_j) - 2^k \\ &\geq \frac{u_j}{|r_j|} D_{\text{FS}}^k(r_j) - \left(\frac{u_j}{|r_j|} + 1 \right) 2^k && \text{Lemma 3.9} \\ &\geq u_j - \frac{u_j}{|r_j|} 2^{k/2} - \left(\frac{u_j}{|r_j|} + 1 \right) 2^k && \text{choice of } r_j \\ &\geq u_j - \frac{u_j}{|r_j|} 2^{k+1} \\ &= u_j \left(1 - \frac{2^{k+1}}{k'} \right). \end{aligned} \quad (3.6)$$

By (3.5) and (3.6),

$$\begin{aligned} &D_{\text{FS}}^k(S_1 \dots S_j) - D_{\text{FS}}^{k'+c}(S_1 \dots S_j) \\ &\geq u_j \left(1 - \frac{2^{k+1}}{k'} - \frac{1}{|r_j|} \right) - 2t_{j-1} - 2 \\ &= u_j \left(1 - \frac{2^{k+1} - 1}{k'} \right) - 2t_{j-1} - 2 \\ &\geq \frac{u_j}{2} - 2t_{j-1} && \text{def of } k' \\ &= \frac{1}{4}u_j + t_{j-1} \\ &\geq \frac{1}{4}(|S_j| + |S_1 \dots S_{j-1}|) && \text{def of } u_j \text{ and } t_{j-1} \\ &= \frac{1}{4}|S_1 \dots S_j|. \end{aligned}$$

Because $\phi(j)$ revisits every pair (k, k') , with $k' = 2^{2^{k+1}}$, for every k , there exists $\widehat{k} = k' + c$ such that, on infinitely many j , the above inequality holds. Hence S is finite-state deep. \square

4 Polynomial-Time Depth

4.1 Measure in the Complexity Class E

We use Lutz's measure theory for the complexity class E, which we now briefly describe. See [15] for more details.

Measure on E is obtained by imposing appropriate resource bounds on a game theoretical characterization of the classical Lebesgue measure. A *martingale* is a function $d : \{0, 1\}^* \rightarrow [0, \infty)$ such that, for every $w \in \{0, 1\}^*$,

$$d(w) = \frac{d(w0) + d(w1)}{2}.$$

We say that a martingale d *succeeds* on a language L if $\limsup_{n \rightarrow \infty} d(L \upharpoonright n) = \infty$. Intuitively, d is a gambler that bets money on each successive bit of χ_L , doubling the money bet on the bit that occurs, and losing the rest. It succeeds by making unbounded money.

A class of languages \mathcal{C} has *p-measure zero*, and we write $\mu_p(\mathcal{C}) = 0$, if there is a polynomial-time computable martingale that succeeds on every language in \mathcal{C} . \mathcal{C} has *measure zero in E*, denoted $\mu(\mathcal{C} | E) = 0$, if $\mathcal{C} \cap E$ has p-measure zero. A class \mathcal{C} has *p-measure one* if $\overline{\mathcal{C}}$ has p-measure zero, where $\overline{\mathcal{C}}$ denotes the complement of \mathcal{C} , and \mathcal{C} has *measure one in E*, denoted $\mu(\mathcal{C} | E) = 1$, if $E - \mathcal{C}$ has p-measure 0. p-measure yields a size notion on the class E similar to Lebesgue measure on the Cantor space. Subsets of E that have p-measure zero are then “small subsets of E”; for example, the singleton set $\{L\}$ for any $L \in E$. E, being the largest subset of itself, has p-measure one.

We say that a language L is *E-random* if $\{L\}$ does not have p-measure 0.

4.2 Polynomial-Time Depth

This section proposes a variation of depth based on polynomial-time predictors, which, given a language L , try to predict the n^{th} bit of L 's characteristic sequence (i.e., the membership of s_n in L), without having access to $L \upharpoonright (n - 1)$ (unlike a martingale). Intuitively, L is polynomial-time deep if giving a polynomial-time predictor more time allows it to predict bits of L with significantly greater accuracy.

A *predictor* is a function $P : \{0, 1\}^* \times \{0, 1\} \rightarrow [0, 1]$ such that $P(x, 0) + P(x, 1) = 1$ for any string x . Intuitively, when trying to predict a language L , $P(x, 1)$ is the probability with which the predictor predicts that $x \in L$. To measure how well a predictor P predicts L , we consider its associated martingale $p : \{0, 1\}^* \rightarrow [0, \infty)$ given by

$$p(L \upharpoonright n) = 2^n \prod_{y < s_n} P(y, L(y)).$$

We shall consider predictors P such that $P(s_n, b)$ is computable in time polynomial in n , and call such a P a polynomial-time predictor (pp), and p its polynomial-time betting strategy (pbs).

Definition 4.1. A language L is *polynomial-time deep* if there exists $a > 0$ such that, for all pbs p , there exists a pbs p' such that, for infinitely many $n \in \mathbb{N}$,

$$\frac{p'(L \upharpoonright n)}{p(L \upharpoonright n)} \geq a \log n,$$

with the convention that $\frac{1}{0} = \infty$. L is *polynomial-time shallow* if it is not polynomial-time deep.

Remark. The choice of $O(\log n)$ in the definition of *polynomial-time deep* is somewhat arbitrary, and can be replaced by faster growing functions like polynomials or 2^{an} for a fixed $a > 0$.

Sequences that are either trivial or random for \mathbf{E} are polynomial-time shallow.

Proposition 4.2.

1. If $L \in \mathbf{E}$, then L is polynomial-time shallow.
2. If L is \mathbf{E} -random, then L is polynomial-time shallow.

Proof. For the first item, let $a > 0$ and $L \in \mathbf{E}$. Then there exists a pbs p that predicts L correctly on every string, i.e. $p(L \upharpoonright n) = 2^n$ for every $n \in \mathbb{N}$. Hence for any pbs p' we have

$$\frac{p'(L \upharpoonright n)}{p(L \upharpoonright n)} \leq \frac{2^n}{2^n} = 1 < a \log n$$

for almost every n . Because a is arbitrary, S is polynomial-time shallow.

For the second item let $a > 0$ and suppose L is \mathbf{E} -random, i.e. for any pbs d there exists $c \in \mathbb{N}$ such that for every $n \in \mathbb{N}$ $p(L \upharpoonright n) < c$. Hence for any pbs p' we have

$$\frac{p'(L \upharpoonright n)}{p(L \upharpoonright n)} \leq \frac{c}{c'} < a \log n$$

for almost every n , i.e. S is polynomial-time shallow. □

4.3 Slow Growth Law

Given two languages L_1, L_2 and $c \in \mathbb{N}$, we say that L_1 is *cn-time many-one reducible* to L_2 , and we write $L_1 \leq_m^{cn} L_2$, if there is a Turing machine M that on input s_n , halts in less than cn steps and outputs $M(s_n) > s_n$ (M computes a strictly monotone function) such that $s_n \in L_1$ if and only if $M(s_n) \in L_2$. L_1 is *cn-time reversible many-one reducible* to L_2 , and we write $L_1 \leq_{rm}^{cn} L_2$, if $L_1 \leq_m^{cn} L_2$ via M and there exists a cn -time-bounded Turing machine M^{-1} such that, for all $n \in \mathbb{N}$, $M^{-1}(M(s_n)) = s_n$. We say L_1 is *linear-time reversible many-one reducible* to L_2 , and we write $L_1 \leq_{rm}^{\text{lin}} L_2$, if there exists c such that $L_1 \leq_{rm}^{cn} L_2$.

Theorem 4.3 (Slow Growth Law). *Let L_1, L_2 be two languages such that $L_1 \leq_{rm}^{\text{lin}} L_2$. If L_1 is polynomial-time deep, then L_2 is polynomial-time deep.*

Proof. Let f be the reduction from L_1 to L_2 , and let $c \in \mathbb{N}$ such that $f(x)$ and $f^{-1}(x)$ are computable in time $c|x|$. Let $g_1(n) = a \log n$ ($a > 0$) be given by the depth of L_1 . Let $g_2(n) = \frac{a}{c} \log n$. Let us show that L_2 is polynomial-time deep. Let p_2 be any pbs. Consider the following pbs p_1 , where $P_1(x, b) = P_2(f(x), b)$ for any string x , and bit b . p_1 is poly-time computable because p_2 and f are. Since L_1 is polynomial-time deep, there exist a pbs p'_1 and an infinite set N such that for every $n \in \mathbb{N}$,

$$\frac{p'_1(L_1 \upharpoonright n)}{p_1(L_1 \upharpoonright n)} \geq g_1(n).$$

Consider the following pbs p'_2 , where

$$P'_2(x, b) = \begin{cases} P'_1(f^{-1}(x), b) & \text{if } f^{-1}(x) \text{ exists,} \\ P_2(x, b) & \text{otherwise} \end{cases}$$

Hence for every $n \in N$,

$$\begin{aligned} & a \log(n) \\ \leq & \frac{p'_1(L_1 \upharpoonright n)}{p_1(L_1 \upharpoonright n)} && \text{because } L_1 \text{ is polynomial-time deep} \\ = & \frac{\prod_{x \leq s_n} P'_1(x, L_1(x))}{\prod_{x \leq s_n} P_1(x, L_1(x))} \\ = & \frac{\prod_{x \leq s_n} P'_2(f(x), L_2(f(x)))}{\prod_{x \leq s_n} P_1(x, L_1(x))} \\ = & \frac{\prod_{x \leq s_n} P'_2(f(x), L_2(f(x)))}{\prod_{x \leq s_n} P_2(f(x), L_2(f(x)))} \\ = & \frac{\prod_{y \in f(\{s_0, \dots, s_n\})} P'_2(y, L_2(y))}{\prod_{y \in f(\{s_0, \dots, s_n\})} P_2(y, L_2(y))} && \text{putting } y = f(x) \\ = & \frac{\prod_{y \leq f(s_n)} P'_2(y, L_2(y))}{\prod_{y \leq f(s_n)} P_2(y, L_2(y))} && \begin{array}{l} f \text{ strictly monotone and} \\ P'_2(x, b) = P_2(x, b) \text{ if } f^{-1}(x) \text{ undefined} \end{array} \\ = & \frac{p'_2(L_2 \upharpoonright f(s_n))}{p_2(L_2 \upharpoonright f(s_n))} \end{aligned}$$

Because on the time bound on f, f^{-1} , for $s_{m_n} := f(s_n)$ we have $m_n \leq n^c$, thus for any $n \in N$

$$\frac{a}{c} \log m_n \leq \frac{a}{c} \log(n^c) = a \log(n) \leq \frac{p'_2(L_2 \upharpoonright m_n)}{p_2(L_2 \upharpoonright m_n)}$$

i.e. L_2 is polynomial-time deep. □

4.4 Linear-Time Weakly Useful for E Languages

In [3] Bennett showed that the halting language is deep, and Juedes, Lathrop, and Lutz [11] generalized this result. We prove a polynomial-time version of the result of Juedes, Lathrop, and Lutz, namely, that every linear-time weakly useful for E language – a language to which a nonnegligible subset of E can be reduced in fixed linear time – is polynomial-time deep.

Following the definition of weakly useful languages from [11] and [8], we define a language to be linear-time weakly useful for \mathbf{E} if the set of languages in \mathbf{E} reducible to it within a fixed linear time bound is not small, i.e. does not have p -measure zero in \mathbf{E} . Intuitively, a linear-time weakly-useful language for \mathbf{E} is somewhere in between an \mathbf{E} -hard language and a trivial language, in the sense that it does not necessarily enable one to decide *all* languages in \mathbf{E} , but a nonnegligible subset of them (although with the stronger requirement of a fixed linear time bound).

Definition 4.4. A language L is *linear-time weakly useful for \mathbf{E}* if there is a $c \in \mathbb{N}$ such that the set of languages reversible many-one reducible to L in time cn does not have measure zero in \mathbf{E} , i.e. if

$$\mu(L^{\geq_{\text{rm}}^{cn}} | \mathbf{E}) \neq 0$$

where

$$L^{\geq_{\text{rm}}^{cn}} = \{A \mid A \leq_{\text{rm}}^{cn} L\}.$$

An example of a linear-time weakly useful for \mathbf{E} language is the halting language for \mathbf{E} , defined as follows. Fix a standard linear-time computable invertible encoding $(x, y) \mapsto \langle x, y \rangle$. Let M_1, M_2, \dots be an enumeration of machines deciding languages in \mathbf{E} , where machine M_i runs in time 2^{in} . The \mathbf{E} -halting language is given by $H_{\mathbf{E}} = \{ \langle x, i \rangle \mid M_i \text{ accepts } x \}$. It is easy to verify that access to the \mathbf{E} -halting language allows one to decide every language in \mathbf{E} using a reversible many-one $2n$ -time reduction; i.e., $\mathbf{E} \subseteq H_{\mathbf{E}}^{\geq_{\text{rm}}^{2n}}$, whence $H_{\mathbf{E}}$ is linear-time weakly useful for \mathbf{E} .

For all $g : \mathbb{N} \rightarrow \mathbb{N}$ and pbs p define

$$D_p^g = \left\{ L \in \{0, 1\}^\infty \mid (\exists \text{ pbs } p') (\exists^\infty n \in \mathbb{N}) \frac{p'(L \upharpoonright n)}{p(L \upharpoonright n)} \geq g(n) \right\}.$$

Note that L is polynomial-time deep if and only if there exists $a > 0$ such that, for all pbs p , $L \in D_p^{a \log n}$.

Lemma 4.5. For any $g : \mathbb{N} \rightarrow \mathbb{N}$ such that $\lim_{n \rightarrow \infty} 2^n/g(n) = \infty$ and any pbs p , $\mu(D_p^g | \mathbf{E}) = 1$.

Proof. Let g, p be as above. Let $L \in \mathbf{E} - D_p^g$. It suffices to show that p succeeds on L . $L \in \mathbf{E}$ implies the existence of a pbs p' such that for any $n \in \mathbb{N}$, $p'(L \upharpoonright n) = 2^n$. $L \notin D_p^g$ implies that $(\forall \text{ pbs } p') (\exists^\infty n \in \mathbb{N}) p(L \upharpoonright n) > p'(L \upharpoonright n)/g(n)$. Thus $p(L \upharpoonright n) > 2^n/g(n)$, which by hypothesis grows unboundedly as $n \rightarrow \infty$; i.e., p succeeds on L . \square

Theorem 4.6. Every linear-time weakly useful for \mathbf{E} language is polynomial-time deep.

Proof. Let B be a linear-time weakly useful for \mathbf{E} language, i.e. $\mu(B^{\geq_{\text{rm}}^{cn}} | \mathbf{E}) \neq 0$ for some $c \in \mathbb{N}$. Let p_2 be any pbs, and $a = \frac{1}{c}$. It suffices to show that $B \in D_{p_2}^{a \log n}$. Let p_1 be constructed from p_2 as in the proof of Theorem 4.3. $D_{p_1}^{\log n} \cap \mathbf{E}$ has p -measure one by Lemma 4.5, and $B^{\geq_{\text{rm}}^{cn}}$ does not have measure zero in \mathbf{E} . Thus $D_{p_1}^{\log n} \cap B^{\geq_{\text{rm}}^{cn}} \neq \emptyset$, whence there exists a language $A \in D_{p_1}^{\log n} \cap B^{\geq_{\text{rm}}^{cn}}$. Thus $A \leq_{\text{rm}}^{cn} B$ and $A \in D_{p_1}^{\log n}$, so by Theorem 4.3, $B \in D_{p_2}^{a \log n}$. \square

Corollary 4.7. $H_{\mathbf{E}}$ is polynomial-time deep.

Corollary 4.8. *No language in E is linear-time weakly useful for E .*

Corollary 4.9. *No E -random language is linear-time weakly useful for E .*

No decidable language is deep in the sense of Bennett [3] (see also [11, Corollary 5.7]). However, the halting language H is deep and, while not decidable, is computably enumerable. Compare this with the fact that Corollary 4.7 and Proposition 4.2 (or a simple diagonalization) imply that $H_E \notin E$. It is easy to verify, however, $H_E \in \text{DTIME}(2^{n^2}) \subseteq \text{EXP}$. Thus, polynomial-time depth mirrors Bennett’s depth in that E -decidable languages are not polynomial-time deep, but polynomial-time deep languages can be found “close” to E . Similarly, Lemma 4.5 tells us, in an analogous fashion to Corollary 5.10 of [11], that “partially deep” sequences can be found in abundance in E .

References

- [1] L. Antunes, L. Fortnow, D. van Melkebeek, and N. Vinodchandran. Computational depth: Concept and applications. *Theoretical Computer Science*, 354(3):391–404, 2006. Special issue for selected papers from the 14th International Symposium on Fundamentals of Computation Theory.
- [2] K. B. Athreya, J. M. Hitchcock, J. H. Lutz, and E. Mayordomo. Effective strong dimension, algorithmic information, and computational complexity. *SIAM Journal on Computing*. To appear. Preliminary version appeared in *Proceedings of the 21st International Symposium on Theoretical Aspects of Computer Science*, pages 632–643.
- [3] C. H. Bennett. Logical depth and physical complexity. In R. Herken, editor, *The Universal Turing Machine: A Half-Century Survey*, pages 227–257. Oxford University Press, London, 1988.
- [4] E. Borel. Sur les probabilités dénombrables et leurs applications arithmétiques. *Rendiconti del Circolo Matematico di Palermo*, 27:247–271, 1909.
- [5] C. Bourke, J. M. Hitchcock, and N. V. Vinodchandran. Entropy rates and finite-state dimension. *Theoretical Computer Science*, 349:392–406, 2005. To appear.
- [6] J. J. Dai, J. I. Lathrop, J. H. Lutz, and E. Mayordomo. Finite-state dimension. *Theoretical Computer Science*, 310:1–33, 2004.
- [7] D. Doty and P. Moser. Finite-state dimension and lossy decompressors. Technical Report cs.CC/0609096, Computing Research Repository, 2006.
- [8] S. A. Fenner, J. H. Lutz, E. Mayordomo, and P. Reardon. Weakly useful sequences. *Information and Computation*, 197:41–54, 2005.
- [9] J. M. Hitchcock. Fractal dimension and logarithmic loss unpredictability. *Theoretical Computer Science*, 304(1–3):431–441, 2003.
- [10] D. A. Huffman. Canonical forms for information-lossless finite-state logical machines. *IRE Trans. Circuit Theory CT-6 (Special Supplement)*, pages 41–59, 1959. Also available in E.F. Moore (ed.), *Sequential Machine: Selected Papers*, Addison-Wesley, 1964, pages 866–871.

- [11] D. W. Juedes, J. I. Lathrop, and J. H. Lutz. Computational depth and reducibility. *Theoretical Computer Science*, 132(1–2):37–70, 1994.
- [12] Z. Kohavi. *Switching and Finite Automata Theory (Second Edition)*. McGraw-Hill, 1978.
- [13] M. Li and P. M. B. Vitányi. *An Introduction to Kolmogorov Complexity and its Applications*. Springer-Verlag, Berlin, 1997. Second Edition.
- [14] J. H. Lutz. Almost everywhere high nonuniform complexity. *J. Comput. Syst. Sci.*, 44(2):220–258, 1992.
- [15] J. H. Lutz. The quantitative structure of exponential time. In L. A. Hemaspaandra and A. L. Selman, editors, *Complexity Theory Retrospective II*, pages 225–254. Springer-Verlag, 1997.
- [16] C. P. Schnorr and H. Stimm. Endliche Automaten und Zufallsfolgen. *Acta Informatica*, 1:345–359, 1972.
- [17] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–656, 1948.
- [18] D. Sheinwald, A. Lempel, and J. Ziv. On encoding and decoding with two-way head machines. *Information and Computation*, 116, 1995.
- [19] J. Ziv and A. Lempel. Compression of individual sequences via variable-rate coding. *IEEE Transaction on Information Theory*, 24:530–536, 1978.