Feasible Depth

David Doty[1*] and Philippe Moser[2†]

[1] Department of Computer Science, Iowa State University, Ames, IA 50011, USA.
`ddoty (at) iastate (dot) edu`
[2] Dept de Informática e Ingeniería de Sistemas, Centro Politécnico Superior,
Zaragoza, Spain. `mosersan (at) gmail (dot) com`

Abstract. This paper introduces two complexity-theoretic formulations of Bennett's computational 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 language to which a non-negligible subset of E can be reduced in uniform exponential time is polynomial-time deep.

Keywords: dimension, depth, randomness, polynomial-time, finite-state

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 computational depth [3] to formalize computationally the difference between complex and non-complex (trivial or random) structures. Briefly, a computationally 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 [18] or algorithmic information theory [14], 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 computational 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, because they are completely unorganized. One of the key features of Bennett's computational 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 [15]) reduce in a fixed computable time bound.

Unfortunately, because it is based on Kolmogorov complexity, Bennett's computational depth is not computable. Lathrop and Lutz [13] investigated *recursive computational depth*, which is computable, but not within any feasible time scale. 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 an 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 computational 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 [18] and investigated by Huffman [10] and Ziv and Lempel [20]. Informally, a sequence is finite-state deep if given more states, a finite-state machine can decompress the se-

quence from an input significantly shorter than is possible with fewer states. We show that both finite-state trivial sequences (sequences with finite-state strong dimension [2] equal to 0) 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 oblivious predictors. Given a language L, a polynomial-time oblivious 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 similar to that for finite-state depth and computational depth, and a theorem stating that any language which is "useful" for quickly deciding languages in E must be polynomial-time deep. It follows that H_E, the E version of the halting language, is polynomial-time deep.

2   Preliminaries

Proofs of some results are found in the Appendix.

$\mathbb{N}$ 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 $\mathbf{C} = \{0,1\}^\infty$. For a string or sequence S and $i,j \in \mathbb{N}$, $S[i \mathbin{..} j]$ denotes the substring consisting of the ith through the jth bits of S, inclusive, and $S \restriction n$ denotes $S[0 \mathbin{..} n-1]$. For a string x and a string or sequence S, we write $x \sqsubseteq S$ to denote that $x = S \restriction n$ for some $n \in \mathbb{N}$. For a string x, its length is denoted by $|x|$. $s_0, s_1, s_2 \ldots$ 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, and $x \le y$ if $x < y$ or $x = y$.

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 $L \restriction n$ to denote $\chi_L \restriction n$. Given $s_n \in \{0,1\}^*$, let $L(s_n) = \chi_L[n]$ (the value 1 if $s_n \in L$, and 0 if $s_n \notin L$). Let $\mathsf{E} = \bigcup_{c \in \mathbb{N}} \mathsf{DTIME}(2^{cn})$ and $\mathsf{EXP} = \bigcup_{c \in \mathbb{N}} \mathsf{DTIME}(2^{n^c})$.

Let $1 \le i \le j \in \mathbb{N}$. The ith projection function $\mathrm{proj}_i : (\{0,1\}^*)^j \to \{0,1\}^*$, is given by $\mathrm{proj}_i(x_1, \ldots, x_j) = x_i$.

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 [18] and investigated by Huffman [10] and Ziv and Lempel [20]. 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*,
- $\delta : Q \times \{0, 1\} \to Q$ is the *transition function*,
- $\nu : Q \times \{0, 1\} \to \{0, 1\}^*$ 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* $\widehat{\delta} : \{0, 1\}^* \to Q$ 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 $f : \{0, 1\}^\infty \to \{0, 1\}^\infty$ is *FS computable* (resp. *ILFS computable*) if there is a FST (resp. ILFST) T such that, for all $S \in \{0, 1\}^\infty$, $\lim_{n \to \infty} |T(S \restriction n)| = \infty$ and, for all $n \in \mathbb{N}$, $T(S \restriction n) \sqsubseteq f(S)$. 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 $x \in \{0, 1\}^*$, $x \restriction (|x| - c) \sqsubseteq T^{-1}(T(x)) \sqsubseteq x$.*

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

$$\mathrm{D}_{\mathrm{FS}}^k(x) = \min_{p \in \{0,1\}^*} \left\{ |p| \;\middle|\; (\exists T \in \mathrm{FST}^{\leq k}) \; T(p) = x \right\},$$

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, $\mathrm{D}_{\mathrm{FS}}^k$ is a finite state analogue of Kolmogorov complexity. For any sequence S, define the *finite-state dimension* of S by

$$\dim_{\mathrm{FS}}(S) = \lim_{k \to \infty} \liminf_{n \to \infty} \frac{\mathrm{D}_{\mathrm{FS}}^k(S \upharpoonright n)}{n}, \tag{3.1}$$

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

$$\mathrm{Dim}_{\mathrm{FS}}(S) = \lim_{k \to \infty} \limsup_{n \to \infty} \frac{\mathrm{D}_{\mathrm{FS}}^k(S \upharpoonright n)}{n}. \tag{3.2}$$

Finite-state dimension and strong dimension measure the degree of finite-state randomness of a sequence. The above definitions are equivalent [7, 19] to several other definitions of finite-state dimension and strong dimension in terms of finite-state gamblers [2,6], entropy rates [5,20], information lossless finite-state compressors [2, 6, 20], and finite-state log-loss predictors [9].

Schnorr and Stimm [17] (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}.

3.2 Finite-State Depth

Intuitively, a sequence is finite-state deep if a finite state transducer, given additional states (or more accurately, additional bits with which to represent the transducer), can decompress the sequence from a significantly shorter input.

Definition 3.3. *A sequence S is* finite-state deep *if*

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

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

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

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

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

Finite-state trivial and finite-state random sequences are finite-state shallow.

Proposition 3.4. *Let $S \in \mathbf{C}$.*

1. *If $\mathrm{Dim}_{\mathrm{FS}}(S) = 0$, then S is finite-state shallow.*
2. *If S is normal (i.e., if $\dim_{\mathrm{FS}}(S) = 1$), then 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 3.5 (Finite-state slow growth law). *Let S be any sequence, let $f : \{0,1\}^\infty \to \{0,1\}^\infty$ be ILFS computable, and let $S' = f(S)$. If S' is finite-state deep, then S is finite-state deep.*

Finally, we show that finite-state deep sequences exist.

Theorem 3.6. *There exists a finite-state deep sequence.*

4 Polynomial-Time Depth

Because the time bound defining polynomial-time depth is in terms of the characteristic sequence of a language, we focus on the class E of languages decidable in time $2^{c|s_n|}$ for a fixed $c \in \mathbb{N}$, or equivalently, n^c, where n is the length of the characteristic sequence of a language up to the string s_n.

4.1 Measure in E

We use Lutz's measure theory for the complexity class E, which we now briefly describe. See [16] for more details.

Measure on E is obtained by imposing appropriate resource bounds on a game theoretical characterization of the classical Lebesgue measure of subsets of $\mathbf{C}$. A *martingale* is a function $d : \{0,1\}^* \to [0,\infty)$ 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 $\limsup_{n \to \infty} d(L \restriction n) = \infty$. 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 $\mu_{\mathrm{p}}(\mathcal{C}) = 0$, if there is a polynomial-time computable martingale that succeeds on every language in $\mathcal{C}$. $\mathcal{C}$ has *measure zero in* E, denoted $\mu(\mathcal{C} \,|\, \mathsf{E}) = 0$, if $\mathcal{C} \cap \mathsf{E}$ has p-measure zero. A class $\mathcal{C}$ has p-*measure one*, denoted $\mu_{\mathrm{p}}(\mathcal{C}) = 1$, 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 $\mu(\mathcal{C} \,|\, \mathsf{E}) = 1$, if $\mathsf{E} - \mathcal{C}$ has p-measure zero. We say that a language L is E-*random* if the singleton $\{L\}$ does not have p-measure zero.

Measure in E 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 $L \in \mathsf{E}$. E, being the largest subset of itself, has p-measure one.

4.2 Polynomial-Time Depth

This section proposes a variation of depth based on polynomial-time oblivious predictors, which, given a language L, try to predict $L[n]$ (i.e., the membership of s_n in L), *without* having access to $L[0 .. n-1]$. This is in contrast to a martingale, where the bet on $L[n]$ is by definition a function of $L[0 .. n - 1]$. 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.

An *oblivious predictor* is a function $P : \{0,1\}^* \times \{0,1\} \to [0,1]$ such that, for all $x \in \{0,1\}^*$, $P(x,0) + P(x,1) = 1$. 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 $p : \{0,1\}^* \to [0,\infty)$ given by

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

We shall consider predictors P such that $P(s_n, b)$ is computable in time polynomial in n (hence computable in time $2^{c|s_n|}$ for some constant c), and call such a P a *polynomial-time oblivious predictor*, and we call the martingale p its *polynomial-time oblivious martingale (pom)*, with the convention that predictors are given in uppercase and pom in lowercase.

Definition 4.1. *A language L is* polynomial-time deep *if there exists $a > 0$ such that, for all pom p, there exists a pom 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.*

Languages that are trivial or random for E are polynomial-time shallow.

Proposition 4.2. *Let L be a language.*

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

4.3 Slow Growth Law

Let $f : \{0,1\}^* \to \{0,1\}^*$. We say f is *monotone* if, for all $x, y \in \{0,1\}^*$, $x < y \implies f(x) < f(y)$. Given $l : \mathbb{N} \to \mathbb{N}$, we say f is *l-bounded* if, for all $x \in \{0,1\}^*$, $|f(x)| \leq l(|x|)$. Given two languages L_1, L_2 and a time bound $t : \mathbb{N} \to \mathbb{N}$ and length bound $l : \mathbb{N} \to \mathbb{N}$, we say that L_1 is *t-time l-bounded monotone many-one reducible* to L_2 (abbreviated *t-l-M reducible*), and we write $L_1 \leq_{\mathrm{M}}^{t,l} L_2$, if there is a Turing machine M computing a monotone, l-bounded reduction $f : \{0,1\}^* \to \{0,1\}^*$ such that, on input s_n, M halts in at most $t(|s_n|) = t(\log n)$ steps and

outputs $f(s_n) \in \{0,1\}^*$ such that $s_n \in L_1$ if and only if $f(s_n) \in L_2$. We say L_1 is E-*time linearly bounded monotone many-one reducible* to L_2 (abbreviated E-*Lb-M reducible*), and we write $L_1 \leq_{\mathrm{M}}^{\mathsf{E,Lb}} L_2$, if there exists $c \in \mathbb{N}$ such that $L_1 \leq_{\mathrm{M}}^{2^{c|s_n|},c|s_n|} L_2$. We follow the convention of letting n refer to the length of a characteristic sequence, rather than the length of the input string s_n. Therefore, equivalently, $L_1 \leq_{\mathrm{M}}^{n^c,n^c} L_2$; i.e., $f(s_n)$ is computable in time n^c, and, if $m \in \mathbb{N}$ is such that $s_m = f(s_n)$, then $m \leq n^c$.

The following result shows that shallow sequences cannot be transformed into deep ones by simple processes.

Theorem 4.3 (Polynomial-time slow growth law). *Let L_1, L_2 be languages such that $L_1 \leq_{\mathrm{M}}^{\mathsf{E,Lb}} L_2$. If L_1 is polynomial-time deep, then L_2 is polynomial-time deep.*

4.4 Languages that are Useful for E

In [3] Bennett showed that the halting language is deep, and Juedes, Lathrop, and Lutz [11] generalized this result by showing every weakly useful [8, 11] language is deep. We prove a polynomial-time version of the result of Juedes, Lathrop, and Lutz, namely, that every E-Lb-M weakly useful language is polynomial-time deep.

Following the definition of weakly useful languages from [11] and [8], we define a language L to be E-Lb-M weakly useful if the set of languages in E that are reducible to L – within a fixed time and length bound – is not small (does not have measure zero in E). Intuitively, an E-useful language is somewhere in between an E-hard language and a trivial language, in the sense that the language does not necessarily enable one to decide *all* languages in E, but rather a nonnegligible subset of them. Note, however, that an E-hard (for instance, under polynomial-time many-one reductions) language may not necessarily be E-Lb-M weakly useful because of the requirements that an E-Lb-M reduction be monotone and linearly bounded.

Definition 4.4. *A language L is* E-Lb-M weakly useful *if there is a $c \in \mathbb{N}$ such that the set of languages $2^{c|s_n|}$-$c|s_n|$-M reducible to L does not have measure zero in* E*, i.e., if*

$$\mu \left(L^{\geq_{\mathrm{M}}^{2^{c|s_n|},c|s_n|}} \,\middle|\, \mathsf{E} \right) \neq 0$$

where

$$L^{\geq_{\mathrm{M}}^{2^{c|s_n|},c|s_n|}} = \left\{ A \,\middle|\, A \leq_{\mathrm{M}}^{2^{c|s_n|},c|s_n|} L \right\}.$$

In other words, a language L is weakly useful if a nonneglible subset of E monotonically many-one reduces to L within a fixed exponential time bound and fixed linear length bound. An example of an E-Lb-M weakly useful language is the halting language for E, defined as follows. Fix a standard linear-time computable invertible encoding of pairs of strings $(x,y) \mapsto \langle x,y \rangle$. Let $M_1, M_2, \ldots$ be an enumeration of machines deciding languages in E, where machine M_i runs in time

$2^{i|s_n|}$. The E-halting language is given by $H_\mathsf{E} = \left\{ \langle 0^i, x \rangle \mid M_i \text{ accepts } x \right\}$. It is easy to verify that access to the E-halting language allows one to decide every language $L_i \in \mathsf{E}$, decided by machine M_i, using the $1.01|s_n|$-time-bounded, $1.01|s_n|$-length-bounded, monotone reduction $f(x) = \langle 0^i, x \rangle$; i.e., $\mathsf{E} \subseteq H_\mathsf{E}^{\geq 1.01|s_n|, 1.01|s_n|}_{\mathrm{M}}$, whence H_E is E-Lb-M weakly useful.

For every $g : \mathbb{N} \to \mathbb{N}$ and pom p define

$$D_p^g = \left\{ L \in \mathbf{C} \;\middle|\; (\exists \text{ pom } p')(\exists^\infty n \in \mathbb{N}) \; \frac{p'(L \upharpoonright n)}{p(L \upharpoonright n)} \geq g(n) \right\}.$$

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

Lemma 4.5. *For any $g : \mathbb{N} \to \mathbb{N}$ such that $g(n) = o(2^n)$ and any pom p, $\mu\left(D_p^g \,\middle|\, \mathsf{E}\right) = 1$.*

Theorem 4.6. *Every E-Lb-M weakly useful language is polynomial-time deep.*

Corollary 4.7. *H_E is polynomial-time deep.*

Corollary 4.8. *No language in E is E-Lb-M weakly useful.*

Corollary 4.9. *No E-random language is E-Lb-M weakly useful.*

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 4.8 (or a simple diagonalization) implies that $H_\mathsf{E} \notin \mathsf{E}$. It is easy to verify, however, $H_\mathsf{E} \in \mathsf{DTIME}(2^{|s_n|^2}) \subseteq \mathsf{EXP}$. 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 *V. Diekert and M. Habib (eds.), Proceedings of the 21st International Symposium on Theoretical Aspects of Computer Science*, Springer Lecture Notes in Computer Science, Montpellier, France, March 25-27, 2004, pp. 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. J. I. Lathrop and J. H. Lutz. Recursive computational depth. *Information and Computation*, 153(2):139–172, 1999.

14. M. Li and P. M. B. Vitányi. *An Introduction to Kolmogorov Complexity and its Applications*. Springer-Verlag, Berlin, 1997. Second Edition.

15. J. H. Lutz. Almost everywhere high nonuniform complexity. *J. Comput. Syst. Sci.*, 44(2):220–258, 1992.

16. 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.

17. C. P. Schnorr and H. Stimm. Endliche Automaten und Zufallsfolgen. *Acta Informatica*, 1:345–359, 1972.

18. C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–656, 1948.

19. D. Sheinwald, A. Lempel, and J. Ziv. On encoding and decoding with two-way head machines. *Information and Computation*, 116(1):128–133, Jan. 1995.

20. J. Ziv and A. Lempel. Compression of individual sequences via variable-rate coding. *IEEE Transaction on Information Theory*, 24:530–536, 1978.

5 Appendix

This section contains proofs of some results from the main text, as well as auxiliary results needs for these proofs.

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

Lemma 5.1. *Let M be an ILFST. Then*

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

Proof. The proof idea of the lemma is the following. Let k, x be in the statement of the lemma, let p be a k-minimal program for x, i.e. $A(p) = x$ where $A \in \mathrm{FST}^{\leq k}$, and $\mathrm{D}_{\mathrm{FS}}^k(x) = |p|$. 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. $\mathrm{D}_{\mathrm{FS}}^{k+c_1}(M(x)) \leq \mathrm{D}_{\mathrm{FS}}^k(x)$, for some constant c_1. More formally, let

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

be the transition function of A, with

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

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

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

$\square$

Lemma 5.2. *Let M be an ILFST. Then*

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

Proof. The proof is similar to Lemma 5.1. Let k, x be as in the statement of the lemma. By 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 $A \in \mathrm{FST}^{\leq k}$, and $\mathrm{D}_{\mathrm{FS}}^k(M(x)) = |p|$. 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. $\mathrm{D}_{\mathrm{FS}}^{k+c_2}(M(x)) \leq \mathrm{D}_{\mathrm{FS}}^k(x)$, for some constant c_2. $\qquad\square$

Proof (of Proposition 3.4). Let $S \in \mathbf{C}$ satisfy $\mathrm{Dim}_{\mathrm{FS}}(S) = 0$ and let $\alpha > 0$. By (3.2) let $k \in \mathbb{N}$ be such that

$$\limsup_{n \to \infty} \frac{\mathrm{D}_{\mathrm{FS}}^k(S \restriction n)}{n} < \alpha,$$

i.e. $(\forall^\infty n \in \mathbb{N})\ \mathrm{D}_{\mathrm{FS}}^k(S \restriction n) < \alpha n$. Therefore

$$(\forall k' \in \mathbb{N})(\forall^\infty n \in \mathbb{N})\ \mathrm{D}_{\mathrm{FS}}^k(S \restriction n) - \mathrm{D}_{\mathrm{FS}}^{k'}(S \restriction n) \leq \mathrm{D}_{\mathrm{FS}}^k(S \restriction n) < \alpha n.$$

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

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

$$(\forall k' \in \mathbb{N})(\forall^\infty n \in \mathbb{N})\ \mathrm{D}_{\mathrm{FS}}^{k'}(S \restriction n) > (1 - \alpha)\, n.$$

Thus

$$(\forall k' \in \mathbb{N})(\forall^\infty n \in \mathbb{N})\ \mathrm{D}_{\mathrm{FS}}^k(S \restriction n) - \mathrm{D}_{\mathrm{FS}}^{k'}(S \restriction n) < n - (1 - \alpha)\, n = \alpha n.$$

Because α is arbitrary, S is finite-state shallow. $\qquad\square$

Proof (of Theorem 3.5). Let S, S', f be as in the statement of the lemma and M be an ILFST computing f. Because S' is finite-state deep,

$$(\exists \alpha > 0)(\forall k \in \mathbb{N})(\exists k' \in \mathbb{N})(\exists^\infty n \in \mathbb{N})\ \mathrm{D}_{\mathrm{FS}}^k(S' \restriction n) - \mathrm{D}_{\mathrm{FS}}^{k'}(S' \restriction n) \geq \alpha n. \quad (5.1)$$

Let $l \in \mathbb{N}$ and let $c = \max\{c_1, c_2\}$ where c_1, c_2 are the two constants in Lemmas 5.1 and 5.2. Let $l' = k' + c$ where k' is obtained from (5.1) with $k = l + c$. For all $n \in \mathbb{N}$, denote by m_n the smallest integer such that $M(S \restriction m_n) = S' \restriction n$. 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 $n \in \mathbb{N}$,

$$
\begin{aligned}
&\mathrm{D}_{\mathrm{FS}}^l(S \restriction m_n) - \mathrm{D}_{\mathrm{FS}}^{l'}(S \restriction m_n) \\
&= \mathrm{D}_{\mathrm{FS}}^l(S \restriction m_n) - \mathrm{D}_{\mathrm{FS}}^{k'+c}(S \restriction m_n) && l' = k' + c \\
&\geq \mathrm{D}_{\mathrm{FS}}^l(S \restriction m_n) - \mathrm{D}_{\mathrm{FS}}^{k'}(M(S \restriction m_n)) && \text{Lemma 5.2} \\
&= \mathrm{D}_{\mathrm{FS}}^{k-c}(S \restriction m_n) - \mathrm{D}_{\mathrm{FS}}^{k'}(M(S \restriction m_n)) && k = l + c \\
&\geq \mathrm{D}_{\mathrm{FS}}^{k}(M(S \restriction m_n)) - \mathrm{D}_{\mathrm{FS}}^{k'}(M(S \restriction m_n)) && \text{Lemma 5.1} \\
&= \mathrm{D}_{\mathrm{FS}}^{k}(S' \restriction n)) - \mathrm{D}_{\mathrm{FS}}^{k'}(S' \restriction n)) && \text{definition of } m_n \\
&\geq \alpha n && \text{by (5.1)} \\
&\geq \alpha \beta m_n, && \text{because } M \text{ is IL}
\end{aligned}
$$

whence S is finite-state deep. $\qquad\square$

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 5.3. $(\forall l \in \mathbb{N})(\forall x, y \in \{0,1\}^*)$ $\mathrm{D}_{\mathrm{FS}}^l(xy) \geq \mathrm{D}_{\mathrm{FS}}^l(x) + \mathrm{D}_{\mathrm{FS}}^l(y) - 2^l$.

Proof. Let l, x, y be as in the statement of the lemma and suppose $\mathrm{D}_{\mathrm{FS}}^l(xy) = |pp'|$ where $T \in \mathrm{FST}^{\leq l}$, $p, p' \in \{0,1\}^*$, with $T(pp') = xy$, and $T(p) = x$. Thus $\mathrm{D}_{\mathrm{FS}}^l(x) \leq |p|$ 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. $\mathrm{D}_{\mathrm{FS}}^l(y) \leq |p'| + 2^l$. Therefore

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

which proves the lemma. □

Lemma 5.4. $(\exists c \in \mathbb{N})(\forall l \in \mathbb{N})(\forall x, y \in \{0,1\}^*)$ $\mathrm{D}_{\mathrm{FS}}^{l+c}(xy) \leq 2|x| + \mathrm{D}_{\mathrm{FS}}^l(y) + 2$.

Proof. Let l, x, y be as in the statement of the lemma and let p be a minimal program for y, i.e. $\mathrm{D}_{\mathrm{FS}}^l(y) = |p|$ where $A(p) = y$ with $p \in \{0,1\}^*$ and $A \in \mathrm{FST}^{\leq l}$. Let $p' = x'01p$ where x' is x with every bit doubled and let $A' \in \mathrm{FST}^{\leq l+c}$ 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 $A(p)$. □

Proof (of Theorem 3.6). 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 $S = S_1 S_2 \ldots$ in stages, with $S_i \in \{0,1\}^*$ for all $i \in \mathbb{N}$. Let $\phi : \mathbb{N} \to \mathbb{N} \times \mathbb{N}$ 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 $i \in \mathbb{N}$, $\mathrm{proj}_2(\phi(i)) = 2^{2^{\mathrm{proj}_1(\phi(i))+1}}$. Let $j \in \mathbb{N}$ and suppose the prefix $S_1 S_2 \ldots S_{j-1}$ has already been constructed. Let $t_{j-1} = |S_1 S_2 \ldots S_{j-1}|$. 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 $\mathrm{D}_{\mathrm{FS}}^k(S_1 \ldots S_j)$ large, while helping a particular $(k' + c)$-bit FST (c the constant from Lemma 5.4) so that $\mathrm{D}_{\mathrm{FS}}^{k'+c}(S_1 \ldots S_j)$ is small.

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

$$\mathrm{D}_{\mathrm{FS}}^k(r_j) \geq |r_j| - 2^{k/2}. \tag{5.2}$$

Note that such a string always exists because there are at most $|\mathrm{FST}^{\leq k}| \cdot 2^{|r_j|-2^{k/2}} < 2^{|r_j|}$ strings contradicting (5.2). 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 $\mathrm{D}_{\mathrm{FS}}^{k'}(S_j) \leq u_j/|r_j|$. Lemma 5.4 implies that

$$\mathrm{D}_{\mathrm{FS}}^{k'+c}(S_1 \ldots S_j) \leq 2|S_1 \ldots S_{j-1}| + \mathrm{D}_{\mathrm{FS}}^{k'}(S_j) + 2,$$

whence

$$\mathrm{D}_{\mathrm{FS}}^{k'+c}(S_1\ldots S_j) \le 2t_{j-1} + \frac{u_j}{|r_j|} + 2. \tag{5.3}$$

Note that

$$
\begin{aligned}
\mathrm{D}_{\mathrm{FS}}^{k}(S_1\ldots S_j) &\ge \mathrm{D}_{\mathrm{FS}}^{k}(S_1\ldots S_{j-1}) + \mathrm{D}_{\mathrm{FS}}^{k}(S_j) - 2^k && \text{Lemma 5.3}\\
&\ge \mathrm{D}_{\mathrm{FS}}^{k}(S_j) - 2^k \\
&\ge \frac{u_j}{|r_j|}\mathrm{D}_{\mathrm{FS}}^{k}(r_j) - \left(\frac{u_j}{|r_j|} + 1\right)2^k && \text{Lemma 5.3}\\
&\ge u_j - \frac{u_j}{|r_j|}2^{k/2} - \left(\frac{u_j}{|r_j|} + 1\right)2^k && \text{choice of } r_j\\
&\ge u_j - \frac{u_j}{|r_j|}2^{k+1} \\
&> u_j\left(1 - \frac{2^{k+1}}{k'}\right). \tag{5.4}
\end{aligned}
$$

By (5.3) and (5.4),

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

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. □

Proof (of Proposition 4.2). For the first item, let $a > 0$ and $L \in \mathsf{E}$. Then there exists a pom p that predicts L correctly on every string, i.e., $p(L \restriction n) = 2^n$ for every $n \in \mathbb{N}$. Hence for any pom p' we have

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

for all but finitely many n. Because a is arbitrary, L is polynomial-time shallow.

For the second item let $a > 0$ and L be E-random. Then for any pom p there exists $c_p \in \mathbb{N}$ such that for every $n \in \mathbb{N}$, $p(L \restriction n) < c_p$. Fix a pom p such that $p(L \restriction n) \geq 1$ for all $n \in \mathbb{N}$. Then for any pom p' we have

$$\frac{p'(L \restriction n)}{p(L \restriction n)} \leq \frac{c_{p'}}{1} < a \log n$$

for all but finitely many n. Thus L is polynomial-time shallow. $\qquad\square$

Proof (of Theorem 4.3). Let $f : \{0,1\}^* \to \{0,1\}^*$ be the E-Lb-M reduction from L_1 to L_2, and let $c \in \mathbb{N}$ such that f is computable in time n^c $(= 2^{c|s_n|})$, so that, for all $x \in \{0,1\}^*$, $x \in L_1 \iff f(x) \in L_2$, and $|f(x)| \leq c|x|$.

Let p_2 be any pom, such that P_2 is computable in time n^k for some k. Consider the pom p_1, where, for all $x \in \{0,1\}^*$ and $b \in \{0,1\}$, $P_1(x,b) = P_2(f(x),b)$. Then, if $x = s_n$, P_1 is computable in time $n^c + |f(x)|^k \leq n^c + n^{ck}$, so P_1 is computable in time polynomial in n. Since L_1 is polynomial-time deep, there exist a pom p'_1, a constant $a > 0$, and an infinite set $N \subseteq \mathbb{N}$ such that, for every $n \in N$,

$$\frac{p'_1(L_1 \restriction n)}{p_1(L_1 \restriction n)} \geq a \log n. \tag{5.5}$$

Consider the following pom p'_2, where, for all $y \in \{0,1\}^*$ and $b \in \{0,1\}$,

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

For all $n \in \mathbb{N}$, define $m_n \in \mathbb{N}$ such that $s_{m_n} \triangleq f(s_n)$. Because f is monotone, it is 1-1. Thus, if $f^{-1}(y)$ exists, then it is unique. Because f is monotone, $n \leq m_n$. Therefore, letting $y = s_{m_n}$ and $x = f^{-1}(y) = s_n$, x can be computed from y in time polynomial in m_n by searching all strings $w \leq s_{m_n}$ and checking whether $f(w) = y$, which takes at most $m_n n^c \leq m_n^{c+1}$ steps. Hence P'_2 is polynomial-time computable.

For every $n \in N$,

$$
\begin{aligned}
a \log n &\leq \frac{p_1'(L_1 \restriction n)}{p_1(L_1 \restriction n)} && \text{by (5.5)}\\
&= \frac{\prod_{x \leq s_n} P_1'(x, L_1(x))}{\prod_{x \leq s_n} P_1(x, L_1(x))}\\
&= \frac{\prod_{x \leq s_n} P_2'(f(x), L_2(f(x)))}{\prod_{x \leq s_n} P_2(f(x), L_2(f(x)))}\\
&= \frac{\prod_{y \in f(\{s_0, \ldots, s_n\})} P_2'(y, L_2(y))}{\prod_{y \in f(\{s_0, \ldots, s_n\})} P_2(y, L_2(y))} && \text{putting } y = f(x)\\
&= \frac{\prod_{y \leq f(s_n)} P_2'(y, L_2(y))}{\prod_{y \leq f(s_n)} P_2(y, L_2(y))} && P_2'(y,b) = P_2(y,b) \text{ if } f^{-1}(y)\\
& && \text{undefined, and } f \text{ monotone}\\
&= \frac{p_2'(L_2 \restriction m_n)}{p_2(L_2 \restriction m_n)}.
\end{aligned}
$$

Because f is linearly bounded, for all $n \in \mathbb{N}$, $m_n \leq n^c$. Thus for any $n \in N$,

$$
\frac{a}{c} \log m_n \leq \frac{a}{c} \log n^c = a \log n \leq \frac{p_2'(L_2 \restriction m_n)}{p_2(L_2 \restriction m_n)}.
$$

Thus the constant a/c testifies that L_2 is polynomial-time deep. $\qquad\square$

Proof (of Theorem 4.6). Let B be any E-Lb-M weakly useful language, i.e. $\mu\left(B^{\geq_M^{2^{c|s_n|},c|s_n|}} \,\middle|\, \mathsf{E}\right) \neq 0$ for some $c \in \mathbb{N}$. Let $a = \frac{1}{c}$ and let p_2 be any pom. 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. By Lemma 4.5, $\mu\left(D_{p_1}^{\log n} \,\middle|\, \mathsf{E}\right) = 1$. Thus $D_{p_1}^{\log n} \cap B^{\geq_M^{2^{c|s_n|},c|s_n|}} \neq \varnothing$, whence there exists a language $A \in D_{p_1}^{\log n} \cap B^{\geq_M^{2^{c|s_n|},c|s_n|}}$. Thus $A \leq_M^{2^{c|s_n|},c|s_n|} B$ and $A \in D_{p_1}^{\log n}$, so by the proof of Theorem 4.3, $B \in D_{p_2}^{a \log n}$. $\qquad\square$

Proof (of Lemma 4.5). Let g, p be as in the statement of the lemma. Let $L \in \mathsf{E} - D_p^g$. It suffices to show that p succeeds on L. $L \in \mathsf{E}$ implies the existence of a pom p' such that for any $n \in \mathbb{N}$, $p'(L \restriction n) = 2^n$. $L \notin D_p^g$ implies that for all pom p' there are infinitely many $n \in \mathbb{N}$ such that $p(L \restriction n) > p'(L \restriction n)/g(n)$. Thus $p(L \restriction n) > 2^n/g(n)$, which grows unboundedly as $n \to \infty$ because $g = o(2^n)$; i.e., p succeeds on L. $\qquad\square$