On the polynomial depth of various sets of random strings

Philippe Moser [*]

Abstract

We introduce a general framework for defining the depth of an infinite binary sequence with respect to a class of observers. We show that our general framework captures all depth notions introduced in computability/complexity theory so far. We review most such notions, show how they are particular cases of our general depth framework, and review some classical results about the different depth notions. We use our framework to define new notions of polynomial depth (called monotone poly depth), based on a polynomial version of monotone Kolmogorov complexity. We show that monotone poly depth satisfies all desirable properties of depth notions. We give two natural examples of deep sets, by showing that both the set of Levin random strings and the set of Kolmogorov random strings are monotone poly deep.

1 Introduction

From the observation that nature contains both very simple and highly complex structures, Bennett introduced the profound concept of logical depth [1], as a formal definition of *useful* information, as opposed to (random) information in the traditional algorithmic information theory sense. Bennett's original idea is to categorize structures in three groups: trivial, random and deep; where trivial structures being completely predictable contain no useful information; random ones, being completely unpredictable, do not contain any useful information either; both (trivial and random) being therefore shallow objects. On the other hand, structures that are neither random nor trivial i.e., that contain intricate patterns that are neither fully predictable nor completely unpredictable, contain useful information; they are called deep structures. Although random sequences contain a lot of information (in the sense of algorithmic information theory), this information is not of much value, and such sequences are shallow.

Bennett observed that deep objects, because they contain complex well-hidden patterns, cannot be created by easy processes. This observation was formalized in the so-called *slow growth law*, which states that if a simple process (a truth table reduction) transforms some (source) sequence into an (image) sequence that is deep, then the source sequence it started from must be deep i.e., no easy process can transform a shallow sequence into a deep one.

Bennett's logical depth is based on Kolmogorov complexity. Intuitively, a binary sequence is deep, if the more time an algorithm is given, the better it can compress the sequence. Although Bennett's formulation is theoretically very elegant, it is uncomputable, due to the uncomputability of Kolmogorov complexity.

[*]Department of Computer Science, National University of Ireland Maynooth, Maynooth, Co. Kildare, Ireland.

To overcome the uncomputability of logical depth, several notions of feasible depth have been proposed so far. In [2] Juedes, Lathrop and Lutz proposed a computable version of depth, called recursive computational depth. Although recursive depth gets rid of the uncomputability in Bennett's formulation, it is still far from feasible and cannot be used in complexity theory, as most languages of interest in complexity theory –e.g. any NP-complete language– being computable, are neither recursive-deep nor Bennett deep. This strongly motivates the study of polynomial depth notions in complexity theory, and several such notions have been studied so far. Antunes, Fortnow, van Melkebeek and Vinodchandran introduced a depth notion based on polynomial distinguishing complexity in [3]. Doty and Moser studied a depth notion based on polynomial time predictors [4]. In an attempt to reduce the complexity even further, Doty and Moser [4] introduced a finite state versions of depth (based on finite-state compressors).

As seen by the different notions of depth that were studied, depth is not an absolute but a relative notion i.e., a deep sequence is always deep *with respect* to some class of observers. As an example a book of Chinese poetry will look like a sequence of almost randomly ordered symbols to an observer that does not read Chinese i.e., not deep from his point of view. On the other hand, for Chinese readers, the more they read and study the book, the more information they will be able to extract from it, therefore considering it deep from their point of view.

In this paper we introduce a general framework for defining the depth of an infinite binary sequence with respect to classes of competing observers G and G'. Informally we say sequence S is deep relative to G, G' (or (G, G')-deep) if for any observer A from G there is an observer A' from G' such that A' can extract more information from S than A can. More formally suppose we have two classes G, G' of algorithms that compute on binary strings, together with a function perf that measures (by a real between 0 and 1) how much better an algorithm A in G performs on input string x than algorithm $A' \in G'$. For example, if G, G' are classes of compression algorithms, then given compression algorithms A, A' and string x, a possible performance measure of A vs A' on x is the difference of compression ratios $\frac{|A(x)|-|A'(x)|}{|x|}$. As another example, suppose G, G' are classes of predictors trying to predict the bits of the characteristic sequence of an NP-complete language like SAT. A possible performance measure is the difference of the number of correct predictions divided by the total number of bits. The classes G, G' together with the performance function yields a natural notion of depth relative to G, G' i.e., a definition of useful information in the eye of observers drawn from classes G, G'.

We show that our general framework actually captures all depth notions introduced in computability/complexity theory so far [1, 2, 3, 4], which can all be seen as particular instances of our general depth framework. We review most of these previous notions and show how they fit into our general depth framework. We also mention some results that were obtained for these depth notions, and some of their limitations.

Next we use our general depth framework to define new notions of polynomial depth (called monotone-polynomial depth), with the goal of overcoming limitations of previous feasible depth notions, namely the fact that distinguishing depth [3] could collapse (i.e. all sets are shallow), although this is unlikely as it would imply factoring to be in P (see [3] for more details), and that the polynomial depth of [4] is based on oblivious polynomial time predictors that cannot read their input (predictors must predict the nth bit of a sequence without access to the history, i.e. bits $1, 2, \ldots, n-1$).

Our goal is to obtains notions that satisfy the slow growth law, and for which deep objects can be proved to exist unconditionally. The classes of observers (the classes G and G') we consider are based on the notion of monotone polynomial time compression [5], which is a polynomial version of monotone Kolmogorov complexity.

We show that our notions of monotone polynomial depth have all the desired properties of a depth notion, i.e. both trivial and random sequences are shallow, they satisfy a slow-growth law, and deep objects can be shown to exist unconditionally.

Although logical depth is a very profound concept, there have not been many examples of natural deep sequences in the literature so far. Bennett [1] showed that the halting language is deep. Juedes, Lathrop and Lutz [6] generalized Bennett's result by showing that every weakly useful sequence (i.e. a sequence such that the set of languages that can be reduced to it has measure non-zero) is deep, a result that was shown to hold in the context of polynomial depth [4].

In this paper, we give two natural examples of deep languages, in the context of monotone poly depth, namely the set of Levin random strings and the set of Kolmogorov random strings. Levin randomness is a standard randomness notion due to Levin [7]; it is a computable approximation of Kolmogorov complexity, that enjoys many useful properties, among others it provides a search strategy for finding solutions of NP problems, that is optimal up to a multiplicative constant (see [8]).

We show that in the context of polynomial monotone depth, having a test that detects randomness (i.e the set of Levin random strings), is deeper than having access to randomness (a random sequence), by proving the the set of Levin random string is monotone poly deep.

Several authors [9, 10, 11, 12] showed the computational power of the set of Kolmogorov random strings by reducing (using several types of reductions) a broad range of complexity classes to it. Our finding that the set of Kolmogorov random strings is monotone-poly deep is consistent with the results by these authors [9, 10, 11, 12] that show this set contains a lot of useful information.

Earlier drafts of this work appeared in [13, 14].

2 Preliminaries

We write $\mathbb{N}$ for the set of all nonnegative integers. Let us fix some notations for strings and languages. A *string* is an element of $\{0,1\}^n$ for some integer n. We denote by $s_0, s_1, \dots, s_n$ the standard enumeration of strings in lexicographic order. For a string x, its length is denoted by $|x|$. The empty string is denoted by λ. We say string y is a prefix of string x, denoted $y \sqsubseteq x$ (resp. $y \sqsubset x$), if there exists a string a (resp. a non null string) such that $x = ya$. We write $x \sim y$ if x is a prefix of y or vice-versa. For a string x, $\mathrm{dbl}(x)$ is x with every bit doubled. For a set of strings S, let $\mathrm{first}(S)$ denote the first (with respect to the lexicographical order) string of S. A set of strings is prefix free if no string in the set is the prefix of another string in the set.

A sequence is an infinite binary string, i.e. an element of $\{0,1\}^\infty$. For $S \in \{0,1\}^\infty$ and $i, j \in \mathbb{N}$, we write $S[i..j]$ for the string consisting of the i^{th} through j^{th} bits of S, with the convention that $S[i..j] = \lambda$ if $i > j$, and $S[1]$ is the leftmost bit of S. We write $S[i]$ for $S[i..i]$ (the i^{th} bit of S). For a sequence S divided into blocks $S = S_1S_2S_3\dots$, where S_i are strings, $S \upharpoonright S_i$ (resp $S \urcorner S_i$) denotes $S_1 \dots S_i$ (resp. $S_1 \dots S_{i-1}$). For $w \in \{0,1\}^*$ and $S \in \{0,1\}^\infty$, we write $w \sqsubseteq S$ if w is a prefix of S, i.e., if $w = S[1..|w|]$. Unless otherwise specified, logarithms are taken in base 2.

A *language* is a set of strings. The characteristic sequence of a language L is the sequence $\chi_L \in \{0,1\}^\infty$, whose nth bit is one iff $s_n \in L$. We will often use the notation L for χ_L.

TM stands for Turing machine. A monotone TM is a TM such that for any strings x, y, $M(xy) \sqsupseteq M(x)$. A TM is called prefix free if the set of admissible programs is a prefix free set.

Fix a prefix-free universal Turing machine U. The Kolmogorov complexity of x is the length of the shortest program that outputs x i.e.,

$$K(x) = \min_p \{|p| : \ U(p) = x\}.$$

The definition of K does not depend on the choice of the universal TM U, up to an additive constant (see [8]).

E denotes the standard linear exponential time complexity class $\mathsf{E} = \cup_{c\in\mathbb{N}}\mathsf{DTIME}(2^{cn})$. A time bound is a monotone time constructible function $t : \mathbb{N} \to \mathbb{N}$, i.e. there is a TM that on input any string of length n halts in exactly $t(n)$ steps. We will consider the following standard time bound families: Poly $= \cup_{k\in\mathbb{N}}\{t(n) = kn^k\}$, Lin $= \cup_{k\in\mathbb{N}}\{t(n) = kn\}$, Polylog $= \cup_{k\in\mathbb{N}}\{t(n) = k\log^k n\}$ and Comp $= \{t| \ t \text{ is a time bound}\}$.

3 A general framework for depth

Let us give the formal definition of our depth framework, based on the idea of competing observers classes G, G' trying to maximize the amount of information extracted from a string x. More formally let G, G' be two classes of algorithms computing on binary strings. Let perf : $G \times G' \times \{0,1\}^* \to [0,1]$ be a function that measures how much better algorithm A in G' performs on input string x compared to $A \in G$, where 1 (resp. 0) means optimal outperformance (resp. same performance). Usually G' is at least as powerful (or equal to) as G, i.e. $G \subseteq G'$.

Let M be a family of computable functions, where for every $m \in M$ and every integer n, $1 \leq m(n) \leq n$; for example $M = \{c\log n | c \in \mathbb{N}\}$. M will measure by how much an algorithm A' performs better than algorithm A. We thus have all the tools to define depth with respect to (G, G', M).

Definition 3.1 *A sequence $S \in \{0,1\}^\infty$ is a.e. (resp. i.o.) (G, G', M)-deep if for every bound $m \in M$ and every $A \in G$, there exists $A' \in G'$ such that for almost every (resp. infinitely many) $n \in \mathbb{N}$*

$$\mathrm{perf}(A, A', S[1..n]) \geq \frac{m(n)}{n}.$$

The difference between a.e. and i.o. depth is similar to the difference between (resource-bounded)-packing dimension and (resource-bounded)-dimension (see e.g. [15]), where a compressor is required to compress infinitely many prefixes, or almost all prefixes. Bennett's depth [1] is an a.e. notion. Sometimes when the observers are very weak e.g. finite-state, i.o. is the best achievable (e.g. see [4]).

Other variations that have been considered are obtained by replacing "for all bounds $m \in M$" by "there exists bound $m \in M$". Both variations have been used by researchers to define depth notions.

Although defining a depth notion (often) requires arbitrary choices, e.g. a.e. vs i.o., "for all bounds" vs "there exists a bound", choice of M, etc.; they are all variations of a same common theme, which is captured by our general depth framework, and that in essence says that a sequence is deep with respect to (G, G', M) if given any algorithm in G there is an algorithm in G' that performs better (by a factor measured by M) on the sequence.

For readability purposes, we let our depth framework be as general as possible, not putting any restrictions on the perf function for example. A malicious opponent can very easily define a flawed depth notion (e.g. where all sets are deep) that still satisfies our framework. The goal of our framework is not to defeat malicious opponents, but to help non-specialists to quickly get a broad intuition of the notion of depth, without having to look at all existing variations in details. The framework can help researchers define their own notion of depth, but it doesn't spare them the (hard) work of proving that their depth notion is meaningful and sound. We will see such an example in Section 5, where we use our framework to construct a new polynomial depth notion.

This general definition actually captures all depth notions introduced in computability/complexity theory so far [1, 2, 3, 4], as we shall see in the next section, where we will review some of these notions together with some of the results that were obtained for each of them.

4 A review of some computational depth notion

4.1 Bennett's logical depth

The first notion of depth, Bennett's logical depth [1], is based on Kolmogorov complexity. We rephrase it in our general depth framework. We need the following broad definition of compressor.

Definition 4.1 *A compressor is a one-to-one function $C : \{0,1\}^* \to \{0,1\}^*$.*

A compressor is computable if there is a TM T such that for any string x, $T(x) = C(x)$. For the rest of this paper fix a prefix-free universal TM U. $U(p, a)$ denotes machine U run on program p and advice string a, $U(p)$ denotes $U(p, \lambda)$. The universal compressor is given by

$$C_U(x) = \text{first}\{p \mid U(p) = x\}$$

i.e. it is the lexicographically first (hence shortest) program that makes U output x. It is well known that the choice of U affects the size of the output of C_U only up to an additive constant. We therefore omit U in the notation and write C_K (such that $|C_K(x)| = K(x)$). Bennett's logical depth is obtained by letting (note: this in not Bennett's original formulation, but an equivalent one, as shown in [6])

$$\begin{aligned} G &= \{C \mid C \text{ is a computable compressor}\} \\ G' &= \{C_K\} \\ \text{perf}(C, C_K, x) &= \frac{|C(x)| - |C_K(x)|}{|x|} \\ M &= \mathbb{N} \end{aligned}$$

i.e. S is Bennett-deep if for every $C \in G$, every $m \in \mathbb{N}$ and almost every $n \in \mathbb{N}$

$$|C(S[0..n])| - |C_K(S[0..n])| \geq m.$$

On inputs where perf() is greater than 1 (resp less than 0), we set its value to 1 (resp 0), e.g. taking $\min(1, \text{perf}())$ (resp a max with 0). For simplicity of notations we don't write the min (resp max) and will do this for the rest of this paper.

Among others, Bennett showed in [1] that both Martin-Löf random and computable sequences are shallow (i.e. not deep), logical depth satisfies a slow growth law for truth-table reductions, and that the Halting language is deep. This was later generalized by Juedes, Lathrop and Lutz [6] to the class of weakly useful languages –A language is weakly useful if the set of languages reducible to it is not small (in a computable Lebesgue measure sense; see [6] for more details)– where it was shown that every weakly useful language is Bennett-deep.

The main limitation of Bennett's notion is that the universal compressor C_K is not computable. A way to overcome this was proposed by Lathrop and Lutz in [2], by replacing G' with G, in Bennett's formulation, which we describe next.

4.2 Recursive computational depth

Recursive computational depth was proposed in [2] as a way to overcome the uncomputability in Bennett's definition. It is obtained by letting

$$\begin{aligned} G = G' &= \{C \mid C \text{ is a computable compressor}\} \\ \text{perf}(C, C', x) &= \frac{|C(x)| - |C'(x)|}{|x|} \\ M &= \mathbb{N} \end{aligned}$$

i.e. S is recursive deep if for every $C \in G$, and every $m \in \mathbb{N}$ there exists $C' \in G$ such that for almost every $n \in \mathbb{N}$

$$|C(S[0..n])| - |C'(S[0..n])| \geq m.$$

Among others it was shown in [2] that Bennett's depth and recursive depth are two separate notions, that recursive depth also satisfies a slow growth law for truth-table reductions and that both Martin-Löf random and computable sequences are not recursive deep.

Although recursive depth gets rid of the uncomputable universal compressor C_K, it is still far from feasible and cannot be used in complexity theory, as most languages of interest in complexity theory –e.g. any NP-complete language– being computable, are neither recursive-deep nor Bennett deep. This motivates a notion of polynomial depth in the context of complexity theory. We review the first such notion in the following section.

4.3 Distinguishers based polynomial time depth

Antunes, Fortnow, van Melkebeek and Vinodchandran introduced three depth notions in [3]. The first two – called basic computational depth – are only "half polynomial" in the sense that G' contains the universal compressor C_K, and can be expressed in our framework. The third notion was obtained via the notion of distinguishers. Here is a definition.

Definition 4.2 *A distinguisher D for a string x is a function $D : \{0,1\}^* \to \{0,1\}$, such that for any string z,*

$$D(z) = \begin{cases} 1 & \textit{if } z = x \\ 0 & \textit{otherwise.} \end{cases}$$

A n^i-time ($i \in \mathbb{N}$) distinguisher for x is a distinguisher D for x that is computable in time $O(|x|^i)$.

The third depth notion in [3] is obtained by letting

$$\begin{aligned} G &= \big\{C_i \mid i \in \mathbb{N}, C_i(x) = \text{first}\{p \mid U(p) = x \text{ in at most } i|x|^i \text{ steps}\}\big\} \\ G' &= \big\{D_i \mid i \in \mathbb{N}, D_i(x) = \text{first}\{p \mid U(p, \_) \text{ is a } n^i\text{-distinguisher for } x\}\big\} \\ \text{perf}(A, A', x) &= \frac{|A(x)| - |A'(x)|}{|x|} \\ M &= O(\log n). \end{aligned}$$

The relationship between G and G' i.e. between distinguishing a string and producing a string in polynomial time is not known, (beyond the fact that producing implies distinguishing). It is possible both G and G' are very close, which would make the notion of distinguishing depth trivial, but this is unlikely as it would imply factoring to be in P (see [3] for more details).

Connections were demonstrated between depth and average-case complexity, nonuniform circuit complexity, and efficient search for satisfying assignments to Boolean formulas in [3], and the third depth notion was shown to satisfy a slow growth law for restricted polynomial time reductions.

4.4 Predictors based polynomial depth

The depth notion from Doty and Moser [4] is based on polynomial time oblivious predictors, that try to predict the next bit of the characteristic sequence of a language without having access to the history of previously seen bits; here is a definition

Definition 4.3 *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 (i.e. a function $d : \{0,1\}^* \to \mathbb{R}_+$ such that $d(\lambda) = 1$ and $d(w0)+d(w1) = 2d(w)$ for any string w) $d_p : \{0,1\}^* \to [0,\infty)$ given by

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

This definition can be motivated by the following betting game in which gambler d_p puts bets on the successive membership bits of the hidden language L. The game proceeds in infinitely many rounds where at the end of round n, it is revealed to the gambler whether $s_n \in L$ or not. The game starts with capital 1. Then, in round n, d_p bets a certain fraction $\epsilon_w p(w)$ of his current capital $d_p(w)$, that the nth word $s_n \in L$, and bets the remaining capital $(1 - \epsilon_w)d_p(w)$ on the complementary event $s_n \notin L$. The game is fair, i.e. the amount put on the correct event is doubled, the one put on the wrong guess is lost. The value of $d_p(w)$, where $w = \chi_L[0..n]$ equals the capital of d_p after round n on language L.

An oblivious polynomial predictor p is an oblivious predictor such that $p(s_n, b)$ is computable in time polynomial in n. Polynomial oblivious predictors were used in [4] to define a polynomial depth notion by letting

$$\begin{aligned} G = G' &= \{p \mid p \text{ is an oblivious polynomial time predictor}\} \\ \text{perf}(p, p', x) &= \frac{\log d_{p'}(x) - \log d_p(x)}{|x|} \\ M &= \{\log \log n + O(1)\} \end{aligned}$$

It was shown in [4] that this depth notion satisfies a slow growth law for restricted polynomial time reductions, that similarly to Bennett's notion, polynomial time weakly useful languages (a polynomial version of weakly useful languages) are polynomial deep, and that the corresponding polynomial time version of random and computable sets are not polynomial deep.

Polynomial oblivious predictors are somehow restricted because they cannot access the history of previously seen bits. We will overcome this limitation in Section 5 by introducing a notion of depth based on monotone polynomial time compressors.

Secondly, although the predictors are polynomial time computable, there is no control over the polynomial exponent. In an attempt to reduce the computational power of the algorithms even further, a finite state version of depth was introduced by Doty and Moser in [4], which we review next.

4.5 Finite-state Depth

Finite-state depth [4] is based on the standard model of finite-state transducer. 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.*

As usual the canonical extension of the transition function $\widehat{\delta} : \{0,1\}^* \to Q$ is defined for all $x \in \{0,1\}^*$ and $a \in \{0,1\}$ by the recursion

$$\widehat{\delta}(\lambda) = q_0, \text{ and } \widehat{\delta}(xa) = \delta(\widehat{\delta}(x), a).$$

For $x \in \{0,1\}^*$, the *output* of T on x is the string $T(x)$ defined by the recursion

$$T(\lambda) = \lambda, \text{ and } T(xa) = T(x)\nu(\widehat{\delta}(x), a)$$

for all $x \in \{0,1\}^*$ and $a \in \{0,1\}$.

An 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. An 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. A finite state compressor, is an *information lossless finite-state transducer* (ILFST). ILFST denotes the set of all information lossless finite-state transducers.

The finite state depth notion from [4] is obtained by considering finite state compressors, i.e. by letting

$$\begin{aligned} G = G' &= \text{ILFST} \\ \text{perf}(C, C', x) &= \frac{|C(x)| - |C'(x)|}{|x|} \\ M &= \{m(n) = \alpha n |\ \alpha \in [0,1]\} \end{aligned}$$

It is shown in [4] that FS-deep sequences exist, that FS-depth satisfies a slow growth law for information lossless finite state transducers, and that the corresponding finite-state versions of random and computable sets are not finite-state deep.

5 Monotone polynomial depth

To try to overcome some limitations of previous polynomial depth notions, we use our general depth framework to define a new polynomial depth notion based on polynomial monotone compression from [5].

Definition 5.1 *[5] Let Δ be a family of (at least linear) time bounds (e.g. Poly, Lin, etc) and $S \in \{0,1\}^\infty$. A Δ-compression of S is a 3-tuple (C, D, p) where C, D are monotone TMs and $p \in \{0,1\}^\infty$ such that there exists a time bound $t \in \Delta$ such that*

1. *Decompression: For all $j \in \mathbb{N}$, $D(p[1..j])$ outputs $S[1..i_{D,j}]$ in time $t(i_{D,j} + j)$, where $i_{D,j}$ is a monotone sequence of integers.*
2. *Compression: For all $i \in \mathbb{N}$, $C(S[1..i])$ outputs several strings in time $t(i)$, one of which is a prefix p' of p, such that $D(p') \sqsupseteq S[1..i]$.*

The integer $i_{D,j}$ is the number of bits the decompressor D can output given j bits of input i.e., the larger the difference $i_{D,j} - j$ the greater the compression. It is also implicit in [5] that the decompression should not take too much time before it outputs bits, i.e. for every $k \leq i_{D,j}$, the k-th bit of $S[1..i_{D,j}] = D(p[1..j])$ is output after at most $t(k)$ steps, where t is the time bound of D.

When $\Delta = \text{Comp}$, we drop the compression requirement, i.e. a Comp-compression is a 2-tuple (D, p). This is because the compressor C may be uncomputable. When $\Delta = \text{Comp}$ we are in the realms of Kolmogorov complexity, where similarly there is no (computable) compressor but only a computable decompressor (the universal TM U).

To avoid extreme decompression of the form "On input n, output $2^{2^{\cdot^{\cdot^{\cdot^{2^n}}}}}$ zeroes", we fix the maximal decompression factor we allow i.e., let MD (maximal decompression) be a function such that $\text{MD}(j)$ is computable in $O(\text{MD(j)})$ time for any integer j (e.g. $\text{MD}(j) = 2^{2^{2^j}}$). We require that for any Δ-compression (C, D, p), and for every integer j,

$$i_{D,j} \leq \text{MD}(j)$$

i.e. MD is the same for all compressors; and is assumed fixed for the rest of this paper. This is not really a restriction as the equivalence between p-measure and polynomial monotone compression of [5] still holds for MD-bounded polynomial monotone compression. The main reason for a bounded version is to avoid trivial sequences such as $S = 000\ldots$ to be deep by having more and more decompression.

Let us introduce our new monotone-poly-depth notions, based on monotone poly compressors.

$$\begin{aligned}
G = G(S) &= \{(C, D, p) |\ (C, D, p) \text{ is a } \Delta\text{-compression of } S\} \\
G' = G'(S) &= \big\{(C', D', p') |\ (C', D', p') \text{ is a } \Delta'\text{-compression of } S\big\} \\
\text{perf}(D, D', S[1..n]) &= \frac{i_{D', j_n^{D'}} - i_{D, j_n^{D'}}}{n \log i_{D', j_n^{D'}}} \\
M &= O(1)
\end{aligned}$$

where $j_n^{D'} = \max_j [i_{D',j} \leq n]$, $(n \in \mathbb{N})$ is the maximum number of bits of p' that decompresses to a prefix of $S[1..n]$ (this guarantees $\text{perf}() \leq 1$). Thus the performance is measured by comparing how many bits of S D' can extract given $j_n^{D'}$ bits of program compared to D given the same number of bits, normalized by dividing by n multiplied by a bound function $\log i_{D',j_n^{D'}}$.

Because the important parameter for monotone compression [5] is the decompression rate $i_{D,j}$ (as opposed to the compression rate), our monotone depth notion is based on this rate also. The $\log i_{D',j_n^{D'}}$ term could be moved in the definition of the bounds family M, but to keep definitions concise, we chose to leave M independent of G, G'. We will often use the following reformulation in terms of decompression rates.

Definition 5.2 *$S \in \{0,1\}^\infty$ is a.e. (resp i.o.) (Δ, Δ')-deep if for every Δ-compression (C, D, p) of S and any $a > 0$, there exists a Δ'-compression (C',D',p') of S such that for almost every (resp. infinitely many) $j \in \mathbb{N}$*

$$i_{D',j} - i_{D,j} \geq a \log i_{D',j}. \tag{1}$$

All our results on monotone-poly depth use the stronger a.e. formulation (which implies an i.o. result), except Theorem 6.4.
A sequence is (Δ, Δ')-shallow if it is not (Δ, Δ')-deep.

The choice of the log function in Equation 1 is arbitrary. As mentioned in Section 4, previous depth notions e.g. [1, 2], only required the difference be unbounded and the rate was not specified, (Note that Bennett's and recursive depth also work with a log rate function). Most feasible depth notions published after Bennett's paper [3, 4] used a logarithmic rate function. We choose to do the same.

Recall from section 4 recursive depth [2] is defined in terms of computable observers competing against computable observers, i.e. G and G' have the same power. The natural polynomial version is to let $\Delta = \Delta' = \text{Poly}$. We call this notion monotone-Poly-depth.

As mentioned in section 4, Bennett's depth [1] on the other hand is based on observers of different strength i.e., computable compressors competing against noncomputable Kolmogorov complexity. For Bennett's notion, there is no unique translation into the polynomial world. We propose to study $(\Delta = \text{Lin}, \Delta' = \text{Poly})$ as a polynomial version of Bennett's depth (called monotone-Lin-depth), which encompasses the idea of observers of different strength (Lin vs Poly), but keeping both in the polynomial setting. The choice $(\Delta = \text{Lin}, \Delta' = \text{Poly})$ is actually flexible, and Poly (resp. Lin) could be replaced by anything strictly stronger (resp. weaker) than Lin, e.g. $O(n^2)$ (resp. Polylog), without modifying our results on monotone-Lin-depth from Section 7 (the choice Δ =Polylog, would require a modification of the notion of Δ-compression [5] to allow for sublinear running time, in the same way as martingales where modified to allow sublinear time bounds in [16]). The choice Lin vs Poly reflects the difference in power of complexity classes E and EXP, which are the complexity classes on which Δ-compression was first introduced in [5] to define a measure notion, and which traditionally are the two complexity classes considered for measure notions based on polynomial time martingales [17].

In [3] Antunes et al. proposed another resource-bounded version of Bennett's depth [1] called basic computational depth, by looking at bounded (sublinear or polynomial) Kolmogorov complexity vs unbounded Kolmogorov complexity. We introduce a translation of basic computational depth [3] in the setting of polynomial monotone compressors, by setting $(\Delta = \text{Poly}, \Delta' = \text{Comp})$. We call this notion basic-monotone-Poly-depth (bm-Poly-depth); bm-Poly-depth captures the idea behind basic computational depth [3] but with Kolmogorov complexity replaced by monotone compressors.

6 Basic properties of monotone-Poly-depth

In the next section we study the basic properties of monotone-Poly-depth. All results remain true for both monotone-Lin-depth and bm-Poly-depth.

It is a key feature of logical depth [1] that both trivial (computable) and random sequences are shallow. In this section we show that a similar result holds in the context of monotone-Poly-depth. Let us define what is meant by trivial sequences in the context of polynomial depth. Informally a sequence is trivial if its prefixes can be maximally compressed.

Definition 6.1 *Let $S \in \{0,1\}^\infty$. S is Poly-optimally-compressible if there exists a Poly-compression (C, D, p) of S, such that $i_{D,j} = MD(j)$ for almost every $j \in \mathbb{N}$.*

As an example, it is easy to check that the characteristic sequences of languages in E are Poly-optimally-compressible. The following result shows that optimally-compressible sequences are shallow.

Theorem 6.1 *Every Poly-optimally-compressible sequence is a.e. Poly-shallow.*

Proof. Let S be a Poly-optimally-compressible sequence, and let (C_0, D_0, p_0) be a Poly-compressor witnessing this fact. By definition, S is Poly-shallow if there exists a Poly-compression (C, D, p) of S and $a > 0$, such that for any Poly-compression (C', D', p') of S, and for infinitely many $j \in \mathbb{N}$

$$i_{D',j} - i_{D,j} < a \log i_{D',j}.$$

Letting $a = 1$ and $(C, D, p) = (C_0, D_0, p_0)$ implies that for almost every $j \in \mathbb{N}$, $i_{D,j} = \mathrm{MD}(j)$. Let (C', D', p') be any Poly-compression of S, then for almost every $j \in \mathbb{N}$

$$i_{D',j} - i_{D,j} \le \mathrm{MD}(j) - i_{D,j} = \mathrm{MD}(j) - \mathrm{MD}(j) = 0 < \log i_{D',j}.$$

□

On the other extremity of the scale of randomness, we have random sequences. Informally a sequence is Poly-random if it is not compressible by more than a constant.

Definition 6.2 *Let $S \in \{0,1\}^\infty$. S is Poly-random if for every Poly-compression (C, D, p) of S, there exists $c \in \mathbb{N}$ such that for almost every $j \in \mathbb{N}$*

$$i_{D,j} \le j + c.$$

The following result shows that random sequences are shallow.

Theorem 6.2 *Every Poly-random sequence is a.e. Poly-shallow.*

Proof. Let S be a Poly-random sequence, and let (C, D, p) be the identity compressor i.e. $p = S$ and $C(x) = D(x) = x$ for any string x (in particular $i_{D,j} = j$). Let $a = 1$. Let us show that for any Poly-compression (C', D', p') of S, and for infinitely many $j \in \mathbb{N}$

$$i_{D',j} - i_{D,j} < a \log i_{D',j}.$$

Let (C', D', p') be any Poly-compression of S; then there exists $c \in \mathbb{N}$ such that for almost every $j \in \mathbb{N}$, $i_{D',j} - j \le c$. Thus

$$i_{D',j} - i_{D,j} = i_{D',j} - j \le c < \log i_{D',j}$$

for almost every $j \in \mathbb{N}$. □

6.1 Slow growth law

A key property of logical depth [1], is that depth cannot be easily created. The formalization of this idea is known as the slow-growth law. It states that if a simple process transforms some (source) sequence into an (image) sequence that is deep, then the source sequence it started from must be deep i.e., no easy process can transform a shallow sequence into a deep one. Bennett proved a slow growth law for truth-table reductions (i.e. in the context of logical depth, simple process corresponds to truth-table reductions).

In the following section, we prove a slow growth law in the context of monotone-Poly-depth. As the power of polynomial monotone compressors is much smaller than the unbounded time case considered for Bennett's logical depth, we need to reduce the power of "simple processes" accordingly. We consider honest m-to-one polynomial time reductions; similarly, honest m-to-one reductions were considered in [3, 4] Here is a definition.

Definition 6.3 *Let $S, T \in \{0,1\}^\infty$. S is Poly-monotone reducible to T, if there exist Poly-compression (N, M, p) of S with $p = T$ such that*

1. *M is d-to-one for some integer d.*
2. *Honesty: There exists $a > 0$ such that for every $n \in \mathbb{N}$*

$$n - a \log n \leq |M(T[1..n])| \leq n + a \log n$$

and the same honesty property holds for N.

The following result is a slow-growth law for monotone-Poly-depth. A similar result holds for both monotone-Lin-depth and bm-Poly-depth (provided the reduction is linear-time bounded for monotone-Lin-depth).

Theorem 6.3 *Let $S, T \in \{0,1\}^\infty$, such that S is a.e. monotone-Poly-deep and Poly-monotone reducible to T. Then T is a.e. monotone-Poly-deep.*

Proof. Let S, T be as above, and let M, N denote the Poly-monotone reduction. Let c be the honesty constant (for both M, N). Let us show that T is Poly-deep; let $a > 0$ and (C, D, p) be a Poly-compression of T. This induces a Poly-compression (C_1, D_1, p) for S, with $D_1 = M \circ D$ and $C_1 = C \circ N$. C_1, D_1 run in Poly-time. For any $j, n \in \mathbb{N}$ we have

$$D_1(p[1..j]) = M(T[1..i_{D,j}]) \sqsupseteq S[1..i_{D,j} - c \log i_{D,j}]$$

by honesty of M, which implies

$$i_{D_1,j} \geq i_{D,j} - c \log i_{D,j}. \tag{2}$$

Also

$$C_1(S[1..n]) = C \circ N(S[1..n])$$

where $N(S[1..n])$ outputs several strings one of which is a prefix x of T (i.e. $x = T[1..m]$), such that

$$M(x) = S[1..n'] \quad \text{with } n' \geq n$$

and $C(T[1..m])$ outputs several strings one of which is a prefix y of p, such that

$$D(y) = T[1..m']$$

with $m' \geq m$. Thus

$$D_1(y) = M(D(y)) = M(T[1..m']) \sqsupseteq M(T[1..m]) = S[1..n'] \sqsupseteq S[1..n].$$

This shows that (C_1, D_1, p) is a Poly-compression for S.

Let $b = 2a + 3c > 0$. Since S is Poly-deep, there exists a Poly-compression (C_2, D_2, p_2) for S such that for every $j \in \mathbb{N}$

$$i_{D_2,j} - i_{D_1,j} > b \log i_{D_2,j}. \tag{3}$$

We construct a Poly-compression (C', D', p_2) of T with $D' = N \circ D_2$ and $C' = C_2 \circ M$. C', D' run in Poly-time. For any $j, n \in \mathbb{N}$ we have

$$D'(p_2[1..j]) = N(S[1..i_{D_2,j}]) \sqsupseteq T[1..i_{D_2,j} - c \log i_{D_2,j}]$$

by honesty of N, which implies

$$i_{D',j} \geq i_{D_2,j} - c \log i_{D_2,j} \tag{4}$$

and

$$C'(T[1..n]) = C_2(M(T[1..n])) = C_2(S[1..n'])$$

where n' is an integer such that $M(T[1..n]) = S[1..n']$, and $C_2(S[1..n'])$ outputs several strings one of which is a prefix x' of p_2, such that

$$D_2(x') = S[1..n'']$$

with $n'' \geq n'$. Thus

$$D'(x') = N(S[1..n''])$$

where $N(S[1..n''])$ outputs several strings one of which is a prefix y of T, such that

$$M(y) \sqsupseteq S[1..n''] \sqsupseteq S[1..n'].$$

If $y \sqsubset T[1..n]$, then because M is both monotone and d-to-one (for some constant d), we have $n \leq |y| + d$, thus y can be output together with all its possible 2^{d+1} extensions of total length at most $|y| + d$ (i.e. all ya with $a \in \{0,1\}^{\leq d}$), one of which will be $T[1..n]$. This shows that (C', D', p_2) is a Poly-compression for T.

By Equation 2 we have

$$\log i_{D_1,j} \geq \log(i_{D,j} - c \log i_{D,j}) \geq \frac{1}{2} \log i_{D,j}. \tag{5}$$

Similarly Equation 3 yields

$$\log i_{D_1,j} \leq \log i_{D_2,j} \tag{6}$$

and since $D' = N \circ D_2$

$$\log i_{D',j} \leq 2 \log i_{D_2,j}. \tag{7}$$

Thus for almost every j

$$\begin{aligned}
i_{D',j} - i_{D,j} &\geq i_{D_2,j} - i_{D,j} - c\log i_{D_2,j} && \text{by Equation 4}\\
&\geq i_{D_2,j} - i_{D_1,j} - c\log i_{D_2,j} - c\log i_{D,j} && \text{by Equation 2}\\
&\geq i_{D_2,j} - i_{D_1,j} - c\log i_{D_2,j} - 2c\log i_{D_1,j} && \text{by Equation 5}\\
&\geq i_{D_2,j} - i_{D_1,j} - c\log i_{D_2,j} - 2c\log i_{D_2,j} && \text{by Equation 6}\\
&\geq (b-3c)\log i_{D_2,j} && \text{by Equation 3}\\
&\geq \frac{b-3c}{2}\log i_{D',j} && \text{by Equation 7}\\
&= a\log i_{D',j} && \text{by definition of } b
\end{aligned}$$

i.e. T is deep. □

A similar proof shows that the result holds for both monotone-Lin-depth and bm-Poly-depth (provided the reduction is linear-time bounded for monotone-Lin-depth).

6.2 A Poly deep sequence

The following result shows that our notion admits the existence of deep sequences. Similarly to other feasible depth notions with restricted power [4], our result is an i.o. result.

The proof uses the equivalence between compressors and martingales from [5]. A direct proof can be given without martingales, but using martingales makes the proof easier. It is also interesting to see the correspondence martingales-compressor in the context of depth.

Theorem 6.4 *There exists an i.o. monotone-Poly-deep sequence.*

Proof. As shown in [5] polynomial compression yields an alternative characterization of resource-bounded measure zero sets. Resource-bounded measure is a measure theory within the complexity class E developed by Lutz [17], which is obtained by imposing polynomial resource-bounds on a game theoretical characterization of classical Lebesgue measure, via martingales. A martingale is a function $d : \{0,1\}^* \to \mathbb{R}_+$ such that, for every $w \in \{0,1\}^*$, $2d(w) = d(w0) + d(w1)$, and $d(\lambda) = 1$. This definition can be motivated by the following betting game in which a gambler puts bets on the successive membership bits of a hidden language A. The game proceeds in infinitely many rounds where at the end of round n, it is revealed to the gambler whether $s_n \in A$ or not. A polynomial (computable) martingale is a martingale computable in time polynomial in the input size.

In [5] the following equivalence between polynomial martingales and monotone compression was shown.

Lemma 6.1 *Given a polynomial computable martingale d, and a sequence w, there exists a Poly-compression (C, D) for w such that for any $j \in \mathbb{N}$*

$$i_{D,j} - j \geq \log d(w[1..i_{D,j}]) - 4.$$

Alternatively given a Poly-compression (C, D) for w, there exists a polynomial martingale d such that for any $j \in \mathbb{N}$

$$\log d(w[1..i_{D,j}]) \geq i_{D,j} - j - 2.$$

It was shown in [17] that for every $k \in \mathbb{N}$ there exists a n^k-universal martingale d_k computable in polynomial time, such that for any martingale d computable in time n^k, there exists $c \in \mathbb{N}$, such that for any $w \in \{0,1\}^*$

$$d_k(w) \geq \frac{1}{c} \cdot d(w).$$

The sequence $S = S_1^1 S_1^2 S_2^2 \dots S_1^l \dots S_l^l$ is constructed by induction, where $|S_1^1| = 1$, $|S_k^l| = \mathrm{MD}(|S \upharpoonleft S_k^l|) - |S \upharpoonleft S_k^l|$, and S_k^l diagonalizes against d_k (i.e. d_k does not increase on S_k^l); more precisely, define S_k^l by induction, where for every $t \in \{0, \dots, |S_k^l| - 1\}$, the next bit $b \in \{0,1\}$ of S_k^l is chosen such that

$$d_k((S \upharpoonleft S_k^l) S_k^l[0..t]b) \leq d_k((S \upharpoonleft S_k^l) S_k^l[0..t]).$$

We need the following lemma.

Lemma 6.2 *Let (C, D) be a Poly-compression of S. There exists $k \in \mathbb{N}$ such that for every $l \geq k$, and $j = |S \upharpoonleft S_k^l|$ it holds*

$$i_{D,j} \leq 2j + k.$$

Proof. Let us prove the lemma. Let (C, D) be a Poly-compression of S. By Lemma 6.1 there exists a polynomial martingale d such that for every $j \in \mathbb{N}$

$$\log d(w[1..i_{D,j}]) \geq i_{D,j} - j - 2.$$

Suppose d runs in time n^k (for some $k \in \mathbb{N}$), thus there exists $c \in \mathbb{N}$ such that $d_k(w) \geq \frac{1}{c} d(w)$ for any string w, i.e.

$$\log d_k(w[1..i_{D,j}]) \geq i_{D,j} - j - 2 - \log c.$$

Let $l \geq k$ and $j = |S \upharpoonleft S_k^l|$. We have

$$i_{D,j} \leq \mathrm{MD}(j) = \mathrm{MD}(|S \upharpoonleft S_k^l|) = |S \upharpoonright S_k^l|.$$

By construction of S_k^l, d_k does not increase on it; as d_k can at most double its capital on every bit of $S \upharpoonleft S_k^l$, we have

$$\log d_k(w[1..i_{D,j}]) \leq |S \upharpoonleft S_k^l|$$

which implies

$$i_{D,j} \leq 2j + 2 + \log c.$$

Replacing k and c by $\max(k, c) + 2$ proves the lemma.

Let us show that S is monotone-Poly-deep. Let (C, D) be a Poly-compression of S, and let $a > 0$. Let k be given by Lemma 6.2, $l \geq k$, and $j = |S \upharpoonleft S_k^l|$.

Consider the following Poly-compression (C', D', p) for S. Informally D' reconstructs S_k^l using d_k. Program p is equal to S, except blocks S_k^l (for every $l \in \mathbb{N}$) are omitted, i.e more formally

$$p = S_1^1 S_1^2 S_2^2 \dots S_1^l \dots S_{k-1}^l S_{k+1}^l \dots S_l^l.$$

Since it is easy to compute the sizes of the blocks S_k^l, it is easy to determine where each block starts and stops in p. D' on input a prefix p' of p, can reconstruct the parts in S not in an S_k^l ($l \in \mathbb{N}$) block, by just reading p'. The parts in S in an S_k^l ($l \in \mathbb{N}$) block, can be reconstructed using d_k. On input prefix p' of p of length j, D' can output $S \upharpoonright S_k^l$, i.e. $i_{D',j} = |S \upharpoonright S_k^l| = \mathrm{MD}(j)$. Thus Lemma 6.2 implies

$$\begin{aligned} i_{D',j} - i_{D,j} &\geq \mathrm{MD}(j) - 2j - k \\ &> \frac{1}{2}\mathrm{MD}(j) \\ &= \frac{1}{2} i_{D',j} \\ &> a \log i_{D',j}. \end{aligned}$$

Since there are infinitely many $j = |S \upharpoonright S_k^l|$ (one for every $l \geq k$), S is i.o. monotone-Poly-deep. □

7 The set of Levin random strings is deep

We show that the characteristic sequence of the set of random strings is deep. Our result holds for the standard randomness notion due to Levin [7]; Levin's notion is a computable approximation of Kolmogorov complexity, that enjoys many useful properties, among others it provides a search strategy for finding solutions of NP problems, that is optimal up to a multiplicative constant (see [8]). Here is a definition.

Definition 7.1 *Fix a prefix-free universal Turing machine U. The Levin complexity of a string x is*

$$\mathrm{Kt}(x) = \min_p \{|p| + \log t : \ U(p) = x \textit{ in at most } t \textit{ steps}\}.$$

The definition of Kt does not depend on the choice of the universal TM U, up to an additive constant (see [8]).

The set of Levin random strings is

$$R_{\mathrm{Kt}} = \{x \in \{0,1\}^* : \ \mathrm{Kt}(x) \geq |x| + \log |x|\}. \tag{8}$$

By a standard program counting argument, it is easy to see that $R_{\mathrm{Kt}} \neq \emptyset$. Although the strings in R_{Kt} are shallow, the characteristic sequence of R_{Kt} contains useful information, i.e. is monotone-Lin-deep, as the following result shows.

Theorem 7.1 *R_{Kt} is a.e. monotone-Lin-deep.*

Proof. We need the following lemma.

Lemma 7.1 *For every Lin-compression (C, D, p) of R_{Kt} and for almost every $j \in \mathbb{N}$*

$$i_{D,j} \leq 2^{2^{3j}}.$$

Proof. Let us prove the lemma by contradiction. Suppose there is a linear compression (C, D, p) of R_{Kt} and an infinite set J of integers j such that $i_{D,j} > 2^{2^{3j}}$ i.e.,

$$R_{\mathrm{Kt}} \sqsupseteq D(p[1..j]) \sqsupseteq R_{\mathrm{Kt}}[1..2^{2^{3j}+1}].$$

Let $j \in J$ be large (to be determined later). Letting $d = p[1..j]$ yields a string with high Kt complexity: from $\pi = \langle D, d\rangle$ recover j (from the length of d) and $R_{\mathrm{Kt}}[1..2^{2^{3j}+1}]$. Output the first y with $|y| = 2^{3j}$ and $R_{\mathrm{Kt}}(y) = 1$, i.e.

$$\mathrm{Kt}(y) \geq 2^{3j} + 3j.$$

By encoding π the standard way, i.e. $\pi = \mathrm{dbl}(\langle D\rangle)01d$

$$|\pi| \leq |d| + O(1) = j + O(1).$$

The time to construct y is the time to recover $R_{\mathrm{Kt}}[1..2^{2^{3j}+1}]$ (less than $O(2^{2^{3j}})$ steps) and the time to find y in $R_{\mathrm{Kt}}[1..2^{2^{3j}+1}]$ (less than $O(2^{2^{3j}})$ steps), i.e. a total of at most $O(2^{2^{3j}})$ steps. Therefore

$$\mathrm{Kt}(y) \leq |\pi| + \log O(2^{2^{3j}}) \leq j + O(1) + 2^{3j} < 2^{3j} + 3j$$

for j large enough, which contradicts $R_{\mathrm{Kt}}(y) = 1$; thus ending the proof of the lemma. □

Lemma 7.2 *There exists a Poly-compression (C, D, p) of R_{Kt} such that for almost every $j \in \mathbb{N}$*

$$i_{D,j} = \mathrm{MD}(j).$$

Proof. Let $p = 0^\infty$. D on input $p[1..j]$ computes $i_{D,j} := \mathrm{MD}(j)$. D constructs $R_{\mathrm{Kt}}[1..i_{D,j}]$ by simulating the universal TM on all programs π_l of size at most $\log i_{D,j} + \log\log i_{D,j}$ during t_l steps (the simulation stops as soon as $t_l > 2^{\log i_{D,j} + \log\log i_{D,j}}$), the resulting string of such a simulation is denoted x_l. All strings x_l with $|x_l| \leq \log i_{D,j}$, for which $|\pi_l| + \log t_l \leq |x_l| + \log |x_l|$ have membership bit 0 in the characteristic sequence $R_{\mathrm{Kt}}[1..i_{D,j}]$. All remaining bits in $R_{\mathrm{Kt}}[1..i_{D,j}]$ are 1s. The running time of D is less than

$$O(2^{\log i_{D,j} + \log\log i_{D,j}}) \cdot 2^{\log i_{D,j} + \log\log i_{D,j}} \leq (i_{D,j})^c$$

for some $c \in \mathbb{N}$.

The compressor C on input $R_{\mathrm{Kt}}[1..i]$ finds the smallest j such that $\mathrm{MD}(j) \geq i$, and outputs 0^j. C runs in time polynomial in i. This ends the proof of the lemma. □

Let us show that R_{Kt} is monotone-Lin-deep. Let $a > 0$ and (C, D, p) be a Lin-compression of R_{Kt}, and let (C', D', p') be the Poly-compression from Lemma 7.2. We have

$$\begin{aligned}
i_{D',j} - i_{D,j} &= \mathrm{MD}(j) - i_{D,j} && \text{by Lemma 7.2}\\
&\geq \mathrm{MD}(j) - 2^{2^{3j}} && \text{by Lemma 7.1}\\
&> \frac{1}{2}\mathrm{MD}(j) && \text{by definition of MD}\\
&= \frac{1}{2} i_{D',j} && \text{by definition of } i_{D',j}\\
&> a \log i_{D',j}
\end{aligned}$$

for almost every j i.e. R_{Kt} is monotone-Lin-deep.

8 The set of Kolmogorov-random strings is deep

The next result shows that the set of Kolmogorov random strings is bm-Poly-deep.

Definition 8.1 *Fix a prefix-free universal Turing machine U. For a time bound t, the t-bounded Kolmogorov complexity of x is*

$$K^t(x) = \min\{|p| : \ U(p) = x, \ \textit{and } U \textit{ halts in at most } t(|x|) \textit{ steps}\}.$$

Let $0 < \epsilon < 1$. The set of Kolmogorov random string is

$$R_{K,\epsilon} = \{x \in \{0,1\}^* : \ K(x) \geq \epsilon|x|\}. \tag{9}$$

Theorem 8.1 *Let $0 < \epsilon < 1$. $R_{K,\epsilon}$ is a.e. bm-Poly-deep.*

Proof. Let $0 < \epsilon < 1$ We need the following lemma.

Lemma 8.1 *For every Poly-compression (C, D, p) of $R_{K,\epsilon}$ and for almost every $j \in \mathbb{N}$*

$$i_{D,j} < 2^{j+1}.$$

Proof. Let us prove the lemma by contradiction. Suppose there is a Poly-compression (C, D, p) of $R_{K,\epsilon}$ and an infinite set N of integers j such that $i_{D,j} \geq 2^{j+1}$. Let $c = 4/(1-\epsilon)$ and $j \in N$.

Let $y_1, \ldots, y_c \in \{0,1\}^j$ such that

$$K^{2^{n^2}}(\langle y_1, \ldots, y_c\rangle) \geq cj - O(\log j) \quad \text{but} \quad K(\langle y_1, \ldots, y_c\rangle) \leq O(\log j).$$

Such a c-tuple can be found by simulating U on all programs of appropriate size running in at most 2^{n^2} steps. We have $R_{K,\epsilon}(y_t) = 0$ for every $t = 1, \ldots, c$.

Consider $L = \{(l_1, \cdots, l_c)| \ 1 \leq l_t \leq 2^{\epsilon(j+1)}, \ t = 1, \ldots, c\}$. Let $Q = \{q_l | \ l \in L\}$ with $q_l = \langle \text{code}, p[1..j], l\rangle$ be the set of programs such that U on input q_l simulates $D(p[1..j])$ to reconstruct $R_{K,\epsilon}[1..i_{D,j}] \sqsupseteq R_{K,\epsilon}[1..2^{j+1}]$, which takes time less than $2^{O(j)}$ (U stops once D already output the 2^{j+1} first bits of $R_{K,\epsilon}$). U constructs

$$R_0 = \{r_1 < r_2 < \ldots | \ r_t \in \{0,1\}^{\leq j}, R_{K,\epsilon}(r_t) = 0\}$$

the lexicographically ordered set of all strings of length at most j whose characteristic bit in $R_{K,\epsilon}$ is 0, which takes time $O(2^j)$. If $r_{l_1}, \cdots, r_{l_c} \in R_0$ then output $\langle r_{l_1}, \cdots, r_{l_c}\rangle$ else halt, which takes time at most $O(2^j)$.

On any program $q_l \in Q$, U runs in less than $O(2^j) + 2^{O(j)} \leq 2^{j^2}$ steps. Moreover all l_t ($t = 1, \ldots, c$) can be encoded in at most $\epsilon(j+1)$ bits (because $|R_0| \leq 2^{\epsilon(j+1)}$) i.e., all programs $q_l \in Q$ have size bounded by

$$|q_l| \leq c\epsilon(j+1) + j + O(\log j) \leq (c\epsilon + 1)j + O(\log j).$$

Because $R_{K,\epsilon}(y_t) = 0$ for every $t = 1, \ldots, c$, let $v = (v_1, \ldots, v_c) \in L$ be the vector of the positions of $y_1, \ldots, y_c$ in $R_{K,\epsilon}$ i.e., $r_{v_t} = y_t$ for every $t = 1, \ldots, c$. Thus U on input q_v outputs $\langle y_1, \ldots, y_c\rangle$ i.e., q_v is a program for $\langle y_1, \ldots, y_c\rangle$ that runs in less than $2^{j^2} < 2^{|\langle y_1, \ldots, y_c\rangle|^2}$ steps. Thus we have

$$K^{2^{n^2}}(\langle y_1, \ldots, y_c\rangle) \leq (c\epsilon + 1)j + O(\log j) \quad \text{which implies}$$

$$cj - O(\log j) \leq (c\epsilon + 1)j + O(\log j) \quad \text{i.e.,}$$

$$cj \leq (c\epsilon + 1)j + O(\log j) \leq (c\epsilon + 2)j$$

thus $c(1-\epsilon) \leq 2$ which is a contradiction. □

Lemma 8.2 *There exists a Comp-compression (D, p) of $R_{K,\epsilon}$ such that for almost every $j \in \mathbb{N}$*

$$i_{D,j} = 2^{j/\epsilon}.$$

Proof. Let $p = \Omega[1..n]$ where Ω is the halting probability $\Omega = \sum_{p:U(p)\downarrow} 2^{-|p|}$. D on input $p[1..\epsilon j]$ (wlog ϵj is an integer, otherwise it is replaced by $\lfloor \epsilon j \rfloor$) can compute using standard Dove-tailing (see [8]) whether $U(p)\downarrow$ for all programs p with $|p| \leq \epsilon j$, i.e. it can reconstruct $R_{K,\epsilon}[1..2^{j+1} - 1]$. We have $i_{D,\epsilon j} = 2^j$ i.e., $i_{D,j} = 2^{j/\epsilon}$. □

Let us show that $R_{K,\epsilon}$ is bm-Poly-deep. Let $a > 0$ and (C, D, p) be a Poly-compression of $R_{K,\epsilon}$, and let (D', p') be the Comp-compression from Lemma 8.2. We have

$$\begin{aligned}
i_{D',j} - i_{D,j} &= 2^{j/\epsilon} - i_{D,j} && \text{by Lemma 8.2} \\
&\geq 2^{j/\epsilon} - 2^{j+1} && \text{by Lemma 8.1} \\
&= 2^j(2^{(1/\epsilon-1)j} - 2) \\
&> 2^j && \text{for } j \text{ large enough} \\
&> a \log i_{D',j}
\end{aligned}$$

for almost every j, i.e. $R_{K,\epsilon}$ is a.e. bm-Poly-deep. □

Final remark

We hope our general depth framework will help other researchers investigate new depth notions in the future. As seen from Section 4, most existing depth notions are based on some class of compression algorithms, which as seen by our general depth framework, is only one –among many others– particular way to define the depth of a sequence, therefore leaving the door open to the study of new depth notions, not necessarily based on the compression paradigm.

References

[1] C. H. Bennett, Logical depth and physical complexity, The Universal Turing Machine, A Half-Century Survey (1988) 227–257.

[2] J. I. Lathrop, J. H. Lutz, Recursive computational depth, Inf. Comput. 153 (1) (1999) 139–172.

[3] L. Antunes, L. Fortnow, D. van Melkebeek, N. Vinodchandran, Computational depth: Concept and applications, Theoretical Computer Science 354 (2006) 391–404.

[4] D. Doty, P. Moser, Feasible depth, in: S. B. Cooper, B. Löwe, A. Sorbi (Eds.), CiE, Vol. 4497 of Lecture Notes in Computer Science, Springer, 2007, pp. 228–237.

[5] H. Buhrman, L. Longpré, Compressibility and resource bounded measure, SIAM J. Comput. 31 (3) (2001) 876–886.

[6] D. W. Juedes, J. I. Lathrop, J. H. Lutz, Computational depth and reducibility, Theor. Comput. Sci. 132 (2) (1994) 37–70.

[7] L. A. Levin, Randomness conservation inequalities; information and independence in mathematical theories, Information and Control 61 (1) (1984) 15–37.

[8] M. Li, P. Vitanyi, Introduction to Kolmogorov complexity and its applications, Springer, 1993.

[9] E. Allender, H. Buhrman, M. Koucký, What can be efficiently reduced to the Kolmogorov-random strings?, Ann. Pure Appl. Logic 138 (1-3) (2006) 2–19.

[10] E. Allender, H. Buhrman, M. Koucký, D. van Melkebeek, D. Ronneburger, Power from random strings, SIAM J. Comput. 35 (6) (2006) 1467–1493.

[11] E. Allender, H. Buhrman, M. Koucký, What can be efficiently reduced to the K-random strings?, in: V. Diekert, M. Habib (Eds.), STACS, Vol. 2996 of Lecture Notes in Computer Science, Springer, 2004, pp. 584–595.

[12] E. Allender, H. Buhrman, M. Koucký, D. van Melkebeek, D. Ronneburger, Power from random strings, in: FOCS, IEEE Computer Society, 2002, pp. 669–678.

[13] P. Moser, A general notion of useful information, in: T. Neary, D. Woods, A. K. Seda, N. Murphy (Eds.), CSP, Vol. 1 of EPTCS, 2008, pp. 164–171.

[14] P. Moser, On the polynomial depth of various sets of random strings, in: M. Ogihara, J. Tarui (Eds.), TAMC, Vol. 6648 of Lecture Notes in Computer Science, Springer, 2011, pp. 517–527.

[15] K. B. Athreya, J. M. Hitchcock, J. H. Lutz, E. Mayordomo, Effective strong dimension in algorithmic information and computational complexity, SIAM J. Comput. 37 (3) (2007) 671–705.

[16] P. Moser, Martingale families and dimension in P, Theor. Comput. Sci. 400 (1-3) (2008) 46–61.

[17] J. Lutz, Almost everywhere high nonuniform complexity, Journal of Computer and System Science 44 (1992) 220–258.