On the Difference Between Finite-State and Pushdown Depth

Liam Jordon(✉) and Philippe Moser(✉)

Computer Science Department, National University of Ireland Maynooth, Maynooth, Co Kildare, Ireland
liam.jordon@mu.ie, pmoser@cs.nuim.ie

Abstract. This paper expands upon existing and introduces new formulations of Bennett's logical depth. A new notion based on pushdown compressors is developed. A pushdown deep sequence is constructed. The separation of (previously published) finite-state based and pushdown based depth is shown. The previously published finite state depth notion is extended to an almost everywhere (a.e.) version. An a.e. finite-state deep sequence is shown to exist along with a sequence that is infinitely often (i.o.) but not a.e. finite-state deep. For both finite-state and pushdown, easy and random sequences with respect to each notion are shown to be non-deep, and that a slow growth law holds for pushdown depth.

Keywords: Algorithmic information theory · Kolmogorov complexity · Bennett's logical depth

1 Introduction

In a seminal paper [2], Bennett introduced a new method to measure the *useful* information contained in a piece of data; called logical depth. Logical depth is different from classical information theory in the following sense. Consider a random binary sequence. According to classical information theory, such a random sequence contains a large amount of information because it cannot be significantly compressed while logical depth says that this information is not of much value. Contrast this with a 10-day weather forecast; from the classical information point of view it contains little information (namely no more than the differential equations from which it was originally simulated), but it contains useful information according to logical depth.

Logical depth helps to formalise the difference between complex and non-complex structures. Deep structures can be thought of as structures that contain an underlying patterns which are extremely difficult to find. Given more and more time and resources, an algorithm could spot these patterns and exploit them (such as to compress a sequence).

L. Jordon—Supported by a postgraduate scholarship from the Irish Research Council, Government of Ireland.

A. Chatzigeorgiou et al. (Eds.): SOFSEM 2020, LNCS 12011, pp. 187–198, 2020.
https://doi.org/10.1007/978-3-030-38919-2_16

Bennett's original notion is based on Kolmogorov complexity [2], and interacts nicely with fundamental notions of computability theory as shown in [12].[1] Due to the uncomputability of Kolmogorov complexity, several researchers have attempted to adapt Bennett's notion to lower complexity levels, aka feasible depth. Most of these notions are centered around polynomial time computations [1,10,11] and finite-state machines [6]. Due to the intrinsic limitations of polynomial time (resp. finite state) algorithms, none can match all the nice properties of Bennett's original, i.e. each feasible notion studied represents a trade-off between advantages and limitations.

Similarly to randomness, there is no absolute notion of logical depth, and all variants mentioned above can be seen as variations of a same theme [11], based on the compression framework. However most notions satisfy some basic properties that could be seen as fundamental. These are:

- Random sequences are not deep (for the appropriate randomness notion).
- Computable sequences are not deep (for the appropriate computability notion).
- A slow growth law: deep sequences cannot be *quickly* computed from shallow ones.
- Some deep sequence exists.

In this paper, we continue the study of depth at the finite-state level. In summary, we construct a new depth notion (called ILPDC-depth), based on information lossless pushdown compressors (see [9] for definitions and a comparison with other compressors). We show our notion satisfies the fundamental depth properties mentioned above. We compare ILPDC-depth to finite-state depth [6], and show the two notions are different. This is somehow surprising as pushdown machines are strictly more capable than finite state machines. This shows that although pushdown machines are strictly stronger than finite state machines, stronger does not necessarily mean better.

We also extend the finite-state notion of [6], by introducing an a.e. version (the original [6] is an i.o. version), and show the two notions differ.

Let us explain our results in more details. In the first part of this paper we introduce a notion of pushdown depth. As observed in [11], most depth notions can be expressed in the compression framework, i.e. fix a compressor type T (e.g. finite state, polynomial time, etc.). A sequence S is T-deep, if for every compressor C of type T, there exists a compressor C' of type T (think of C' as being more powerful than C) such that C' compresses almost every prefix of S, better than C. The meaning of "better" will vary with the corresponding depth notion (and actually has consequences on the computational power of

[1] We acknowledge that logical depth was originally defined as depending on both computational complexity and Kolmogorov complexity, which is a descriptional complexity. The new notions in this paper are focused purely on a descriptional complexity lengths, specifically the ratio between the length of the input and the length of the output to restricted classes of transducers. However we continue to call these depth notions to be consistent with previous literature in [6,11].

the sequence, as shown in [10,12]), but for most notions, bounds considered are $O(1), O(\log n)$ and $O(n)$. From the work in [6], it seems linear bounds are appropriate at the finite state level, and thus we also use linear bounds.

We say a sequence is T-deep if for every compressor C of type T, there exists a compressor C' of type T such that on almost every prefix of S (with length denoted n), C' compresses it at least by αn more bits than C, for some constant α. We define ILPDC-depth by setting type T to be information lossless pushdown compressors (ILPDC). Intuitively, an ILPDC is a pushdown transducer such that when run the transducer on input x, the output y and final state q uniquely determines the input x, hence the name information lossless. Contrary to finite state information lossless transducers, it is not known whether this automatically yields a pushdown decompressor (in the finite state model, the finite state decompressor comes for free [7,8]).

We show ILPDC-depth satisfies all fundamental depth properties highlighted above, i.e. both random and easy sequences are not deep, ILPDC-depth satisfies a slow growth law, and there exists a PD-deep sequence.

Next we compare ILPDC-depth to finite state depth [6] (called i.o. FS-depth), and show the two notions are different: we prove there exists a sequence S which is i.o. FS-deep but not ILPDC-deep.

Most notions of depth, measure the compression difference on almost all prefixes of the sequences. Notable exceptions include the original finite state notion [6] (see [12] for further i.o. notions), where the difference is only required be large on infinitely many prefixes of the sequence. Such depth notions are called i.o. depth. In the second part of this paper, we extend the original i.o. FS-depth of [6], to an almost everywhere notion, called a.e. FS-depth. We show there exists an a.e. FS-depth sequence. We also show a.e. FS-depth is a stronger requirement than i.o. FS-depth, by constructing a sequence that is i.o. FS-deep but not a.e. FS-deep.

Due to lack of space, most proofs are omitted. A final journal version of this paper is in preparation.

2 Preliminaries

$\mathbb{N}$ denotes the set of all non-negative integers. A *finite binary* string is an element of $\{0,1\}^*$. A *binary sequence* is an element of $\{0,1\}^\omega$. The length of a string x is denoted by $|x|$. λ denotes the empty string (the string of length 0). For all $n \in \mathbb{N}$, $\{0,1\}^n$ denotes the set of binary strings of length n. For a string (or sequence) S and $i, j \in \mathbb{N}$, $S[i \dots j]$ denotes the i^{th} through j^{th} bits of S with the convention that if $i > j$ then $S[i \dots j] = \lambda$. $S \upharpoonright j$ denotes $S[0 \dots j-1]$, the first j bits of S. For a string x and a string (or sequence) y, xy denotes the string (or sequence) composed of x concatenated with y. For a string x and $n \in \mathbb{N}$, x^n denotes x concatenated with itself n times. For strings $x, y, z \in \{0,1\}^*$, if $w = xyz$, we say y is a substring of w. For a string x, and a string (or sequence) y, we say x is a prefix of y, written as $x \preceq y$, if $x = y[0 \dots |x|-1]$. In particular we occasionally write $x \prec y$ if x is a prefix of y and $|x| < |y|$. The *lexicographic*

ordering of $\{0,1\}^*$ is defined by saying for two strings x, y, x is less than y if either $|x| < |y|$ or else $|x| = |y|$ with $x[n] = 0$ and $y[n] = 1$ for the least n such that $x[n] \neq y[n]$. For a string x, x^{-1} denotes x written in reverse. By intervals of $\mathbb{N}$ we mean closed intervals of $\mathbb{N}$ in the normal sense. All logarithms are taken to be in base 2.

3 Models of Computation

3.1 Finite-State Transducers

We use the standard finite-state transducer model.

Definition 1. *A* finite-state transducer (FST) *is a* 4*-tuple* $T = (Q, q_0, \delta, \nu)$*, where*

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

For all $x \in \{0,1\}^*$ and $b \in \{0,1\}$, the *extended transition function* $\widehat{\delta} : \{0,1\}^* \to Q$, and the transducer output $T : \{0,1\}^* \to \{0,1\}^*$ is defined by the usual recursion.

An FST is *information lossless (IL)* if the function $x \mapsto (T(x), \widehat{\delta}(x))$ is 1–1; i.e. the output and final state of T on input x uniquely identify x. We call an FST that is IL an ILFST. By the identity FST, we mean the ILFST I_{FS} that on every input $x \in \{0,1\}^*$, $I_{\mathrm{FS}}(x) = x$. We write FST to denote the set of all FSTs.

A map $f : \{0,1\}^\omega \to \{0,1\}^\omega$ is said to be *FS computable* (*ILFS computable*) if there is an FST (ILFST) T such that for all $S \in \{0,1\}^\omega$, $\lim_{n\to\infty} |T(S \upharpoonright n)| = \infty$ and for all $n \in \mathbb{N}$, $T(S \upharpoonright n) \preceq f(S)$. In this case we say $T(S) = f(S)$.

It is well known [7,8] that any function computed by an ILFST can be inverted to be approximately computed by another ILFST.

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

Corollary 1. *For any* ILFST T*, there exists an* ILFST T^{-1} *such that for all* $S \in \{0,1\}^\omega$*,* $T^{-1}(T(S)) = S$*.*

3.2 Pushdown Compressors

The model of pushdown compressors we use is the same pushdown compressor model as used in [9]. Note that to keep the model feasible, there is a bound on how long the compressor can empty its stack before it needs to output a symbol.

A *pushdown compressor* (PDC) is an 8-tuple $C = (Q, \Sigma, \Gamma, \delta, \nu, q_0, z_0, c)$ where

1. Q is a non-empty finite set of *states*,
2. Σ is the finite input alphabet,
3. Γ is the finite stack alphabet,
4. $\delta : Q \times (\Sigma \cup \{\lambda\}) \times \Gamma \to Q \times \Gamma^*$ is the *transition function*,
5. $\nu : Q \times (\Sigma \cup \{\lambda\}) \times \Gamma \to \Sigma^*$ is the *output function*,
6. $q_0 \in Q$ is the start state,
7. $z_0 \in \Gamma$ is the special bottom of stack symbol,
8. $c \in \mathbb{N}$ is an upper bound on the number of λ-rules per input bit.

We fix $\Sigma = \{0, 1\}$ and $\Gamma = \{0, 1, z_0\}$. We assume every state in Q is reachable from q_0. We write $\delta = (\delta_Q, \delta_{\Gamma^*})$. The transition function δ accepts λ as an input in addition to $\{0, 1\}$. This means C has the option of altering its stack while not reading an input character. We call this a λ-*rule*. In this case $\delta(q, \lambda, a) = (q', \lambda)$, that is, we pop the top symbol from the top of the stack. To enforce determinism we require that at least one of the following hold for all $q \in Q$ and $a \in \Gamma$:

1. $\delta(q, \lambda, a) = \bot$
2. $\delta(q, b, a) = \bot$ for all $b \in \{0, 1\}$.

δ is restricted so that z_0 cannot be popped off of the stack. That is, for every $q \in Q, b \in \{0, 1\} \cup \{\lambda\}$, either $\delta(q, b, z_0) = \bot$, or $\delta(q, b, z_0) = (q', vz_0)$ where $q' \in Q$ and $v \in \Gamma^*$.

The extended transition function $\delta^* : Q \times \Sigma^* \times \Gamma^+ \to Q \times \Gamma^*$ is defined recursively as usual.

δ^* is abbreviated to δ, and $\delta(q_0, w, z_0)$ to $\delta(w)$. The *output* from state q on input $w \in \{0, 1\}^*$ with $z \in \Gamma^*$ on the top of the stack is defined by the recursion $\nu(q, \lambda, z) = \lambda$,

$$\nu(q, wb, z) = \nu(q, w, z)\nu(\delta_Q(q, w, z), b, \delta_{\Gamma^*}(q, w, z)).$$

The *output* of the compressor C on input $w \in \{0, 1\}^*$ is the string $C(w) = \nu(q_0, q, z_0)$. For a string xy, we write $\bar{\nu}(y)$ as shorthand for $|C(xy)| - |C(y)|$, i.e. the output of C on y after already reading x. It should be clear from context what x is each time this notation is used.

A PDC is said to be *information lossless* (IL) if the function

$$w \mapsto (C(w), \delta_Q(w))$$

is 1–1. A PDC that is IL is called an ILPDC. We write (IL)PDC to be the set of all (IL)PDCs. By the identity PDC I_{PD} we mean the ILPDC that on any input $x \in \{0, 1\}^*$, I_{PD} outputs x without using its stack.

4 Pushdown Depth

Lemma 1 demonstrates the existence of strings that an ILPDC compresses poorly on and is used in proofs throughout this section.

Lemma 1. *Let $d, m \in \mathbb{N}$. Then for all $C \in ILPDC$ with at most d states and for all $x \in \{0,1\}^*$, there exists a string y of length m such that*

$$|C(xy)| - |C(x)| \geq m - \log(d) - 1.$$

The following general depth definition says that S is a.e. T-deep if for every compressor of type T there is a (better) compressor of type T such that the difference of compression, on almost every prefix of S, exceeds some linear bound. More precisely,

Definition 2. *Let S be a sequence. Fix a compressor type* T. *S is a.e.* T*-deep (resp. i.o.* T*-deep) if*

$$(\forall C \in T)(\exists \alpha > 0)(\exists C' \in \mathrm{T})(Qn \in \mathbb{N}) \ [|C(S \upharpoonright n)| - |C'(S \upharpoonright n)| \geq \alpha n],$$

where Q is $\forall^\infty$ (resp. $\exists^\infty$).

Observe if S is a.e. T-deep, it is also i.o. T-deep.

To measure how well a compressor compresses a sequence, we use the following compression ratios.

Definition 3. *Let $S \in \{0,1\}^\omega$. Let T be a family of compressor types.*

1. *The* best-case compression ratio *of type T of S is defined as*

$$\rho_T(S) = \inf\{\liminf_{n\to\infty} \frac{|C(S \upharpoonright n)|}{n} : C \in T\}.$$

2. *The* worst-case compression ratio *of type T of S is defined as*

$$R_T(S) = \inf\{\limsup_{n\to\infty} \frac{|C(S \upharpoonright n)|}{n} : C \in T\}.$$

We define pushdown depth to be a.e. ILPDC-depth.

The following results show that pushdown depth satisfies the basic depth properties, in the sense that both easy and random sequences cannot be deep.

Theorem 2. *Let $S \in \{0,1\}^\omega$.*

1. *If $\rho_{\mathrm{ILPDC}}(S) = 1$, then S is not a.e.* ILPDC*-deep.*
2. *If $R_{\mathrm{ILPDC}}(S) = 0$, then S is not a.e.* ILPDC*-deep.*

The following result shows that pushdown depth satisfies a slow growth law.

Theorem 3 (Slow Growth Law). *Let S be any sequence, let $f : \{0,1\}^\omega \to \{0,1\}^\omega$ be ILFS computable and let $S' = f(S)$. If S' is a.e. ILPDC-deep then S is a.e. ILPDC-deep.*

Remark 1. Theorems 2 and 3 also hold true for i.o. ILPDC-depth.

The following result constructs a pushdown deep sequence S. The sequence is a sequence of blocks, where each block is devoted to some pair of compressors C, C'. On such a block, C compresses poorly, while C' compresses very well. This is achieved by having C' simulate C to find strings it cannot compress. The first found string describes the next bit of the block, and so C' is able to compress the block. In blocks not devoted to him, C' simply simulates C. This ensures that C' never compresses S worse than C, and on blocks devoted to it, C' compresses much better. C' detects whether the current block is devoted to him by signal flags interleaved throughout the sequence. To keep C' IL, as soon as C' makes a wrong prediction, it simply outputs what C does from then on onward. To guarantee an a.e. result, blocks devoted to the same pair repeat every constant number of blocks.

A full construction of C' is omitted for space.

Theorem 4. *There exists an a.e. ILPDC-deep sequence.*

5 Finite-State Depth

The finite-depth in [6] is based on finite-state decompression. However, before we begin examining depth, we first must choose a binary representation of all finite-state transducers.

Definition 4. *A* binary representation of finite-state transducers σ_T *is a partially computable map* $\sigma_T : \{0,1\}^* \to \mathrm{FST}$, *such that for every* FST T, *there exists some* $x \in \{0,1\}^*$ *such that* $\sigma_T(x)$ *fully describes* T. *We say* $|T|_{\sigma_T} = \min\{|x| : \sigma_T(x) = T\}$.

For a binary representation of FSTs σ_T, for all $k \in \mathbb{N}$, define

$$\mathrm{FST}_{\sigma_T}^{\leq k} = \{T \in \mathrm{FST} : |T|_{\sigma_T} \leq k\}.$$

For all $k \in \mathbb{N}$ and $x \in \{0,1\}^*$, the *k-finite-state decompression complexity* of x with respect to binary representation σ_T is defined as

$$D_{\sigma_T}^k(x) = \min_{\pi \in \{0,1\}^*} \left\{ |\pi| \, : \, T \in \mathrm{FST}_{\sigma_T}^{\leq k} \;\&\; T(\pi) = x \right\}.$$

Here π is the shortest program that gives x as an output when inputted into an FST of size k or less with respect to the binary representation σ_T. T can be thought of as the FST that can decompress π to reproduce x.

For the purpose of this paper, we fix the following binary representation of finite-state transducers σ_T. Let $T = (Q, q_0, \delta, \nu)$ be an FST. We define the function the function $\Delta : Q \times \{0,1\} \to Q \times \{0,1\}^*$, where $\Delta(q,b) = (\delta(q,b), \nu(q,b))$. This function Δ completely describes the state transitions and outputs of T. In [3], different encoding schemes are presented to represent each transducer via an encoding of this function Δ.

The binary representation σ_T we fix in this paper is as follows. For a transducer T, if $Q = \{q_1, q_2, \ldots, q_n\}$ and $q_0 = q_i$, for $1 \leq i \leq n$, we encode T by the string

$$d(\text{bin}(i))01\rho$$

where $d(\text{bin}(i))$ is the binary encoding of i which acts as a pointer to the start state of T but with every bit doubled and ρ is an encoding of Δ as seen in [3].

We fix this binary representation σ_T as it is needed to prove Lemma 2 which in turn is needed for Theorem 7, Theorem 8 and Theorem 9. However, we later show that if a sequence S is deep with respect to one depth notion, it is deep with respect to every depth notion. Henceforth, we will drop the σ_T notation and instead write $|T|$ for $|T|_{\sigma_T}$, $\text{FST}^{\leq k}$ for $\text{FST}^{\leq k}_{\sigma_T}$ and $D^k_{\text{FS}}(x)$ instead of $D^k_{\sigma_T}(x)$. All other definitions and results hold and can be proved regardless of the binary representation being used.

To measure the randomness density of a sequence, the following notions are useful. For any sequence S,

1. The *finite-state dimension* of S [5] is defined to be

$$\dim_{\text{FS}}(S) = \lim_{k\to\infty} \liminf_{n\to\infty} \frac{D^k_{\text{FS}}(S \upharpoonright n)}{n},$$

2. The *strong finite-state dimension* of S is defined to be

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

In [6] a notion[2] of depth based on finite-state transducers is introduced called *i.o. finite-state depth.*

Definition 5. *A sequence S is infinitely often (i.o.) finite-state deep if*

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

and $\text{Dim}_{\text{FS}}(S) \neq 0$.

We introduce an a.e. version of the original finite-state notion [6] called *almost everywhere (a.e) finite-state depth.*

Definition 6. *A sequence S is almost everywhere (a.e.) finite-state deep if*

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

and $\text{Dim}_{\text{FS}}(S) \neq 0$.

Remark 2. The condition that $\text{Dim}_{\text{FS}}(S) \neq 0$ is required as otherwise 0^ω would be considered deep.

[2] Actually two notions were introduced, which differ only by the order of quantifiers.

The following result shows that sequences that appear random to finite-state transducers, cannot be finite-state deep. Further study of sequences that appear random to finite-state transducers can be found in [4].

Theorem 5. *Let* $S \in \{0,1\}^\omega$.
If $\dim_{\mathrm{FS}}(S) = 1$*, then* S *is not a.e. finite-state deep.*

Remark 3. We originally hoped to include a version of a slow growth law for a.e. FS-depth. However an adequate notion nor proof has not been found as of yet.

The following theorem demonstrates that if a sequence S is a.e. FS-deep when the size of finite-state transducers are viewed with respect to one binary representation, then it is a.e. FS-deep regardless of what binary representation is used.

Theorem 6. *Let* π_T *be a binary representation of FSTs. Let* S *be an a.e. FS-deep sequence when the size of the finite-state transducers are viewed with respect to a binary representation* π_T*. Then* S *is a.e. FS-deep when the size of the finite-state transducers are viewed with respect to any other binary representation.*

To prove the existence of an a.e. FS-deep sequence we need he following two lemmas.

Lemma 2. *For our fixed binary representation* σ_T*, we have that for* $k \geq 4$, $\forall n \in \mathbb{N}, \forall x, y, z \in \{0,1\}^*$,

$$D^k_{\mathrm{FS}}(xy^n z) \geq D^{3k}_{\mathrm{FS}}(x) + nD^{3k}_{\mathrm{FS}}(y) + D^{3k}_{\mathrm{FS}}(z).$$

Lemma 3. $\forall \epsilon > 0, \forall k \in \mathbb{N}, \exists k' \in \mathbb{N}, \forall x, y, \in \{0,1\}^*$*, whenever* $D^k_{\mathrm{FS}}(x)$ *is sufficiently large*

$$D^{k'}_{\mathrm{FS}}(xy) \leq (1+\epsilon)D^k_{\mathrm{FS}}(x) + D^k_{\mathrm{FS}}(y) + 2.$$

In the following result we construct an a.e. finite-state deep sequence. The sequence is constructed in consecutive blocks, each block is devoted to some pair k, k'. On such a block, transducers of size k do poorly, while some larger transducer of size k' does very well. The key difference with the proof in [6], is that blocks devoted to the same pair k, k' repeat every constant number of blocks. This ensures an a.e. finite-state deep sequence, as opposed to a mere i.o. finite-state deep sequence.

Theorem 7. *There exists an a.e. finite-state deep sequence.*

Remark 4. If $S \in \{0,1\}^\omega$ is a.e. finite-state deep then it is i.o. finite-state deep.

The following result shows that being i.o. FS-deep is a weaker requirement than being a.e. FS-deep.

Theorem 8. *There exists a sequence* S *that is i.o. FS-deep but not a.e. FS-deep.*

5.1 Separation from ILPDC-depth

We next demonstrate a difference between i.o finite state depth [6] to our a.e. pushdown depth notion by constructing a sequence that is i.o. finite-state deep but not a.e. pushdown deep.

Theorem 9. *There exists a sequence S such that S is i.o. finite-state deep, but S is not a.e. ILPDC deep.*

Proof. Fix some $\epsilon > 0$ small. Split $\mathbb{N}$ into intervals $I_1, I_2, I_3, \ldots$ such that $|I_1| = 2^a$ for the smallest constant a such that $2^a > \frac{1}{\epsilon}$, and $|I_j| = 2^{|I_1|+\cdots+|I_{j-1}|}$ for $j \geq 2$. Define $m_j = \min(I_j)$ and $M_j = \max(I_j)$. We construct the sequence $S = S_1 S_2 \cdots$ in stages, with $S_j \in \{0,1\}^{|I_j|}$ for all $j \in \mathbb{N}$. So $S_j = S[m_j \ldots M_j]$.

For S_j, if j is even, S_j is devoted to some FST description bound length $k \in \mathbb{N}$ (what occurs for j odd is discussed later in the proof.) Specifically for each k, k is devoted to every substring S_j where for all $n \geq 0$, $j = 2^k + n2^{k+1}$. $k = 1$ is first devoted to S_2 and every 4^{th} substring after that. $k = 2$ is devoted to S_4 and every 8^{th} interval after that, and so on.

Consider description length k. Let r_k be a string of length $|I_{2^k}|$ such that r_k is $3k$-FS random in the sense that $D^{3k}_{\text{FS}}(r_k) \geq |r_k| - 4k$. Such a string exists as there are at most $|\text{FST}^{\leq 3k}| \cdot 2^{|r_k|-4k} < 2^{|r_k|}$ strings contradicting this. If S_j is devoted to k, we set $S_j = r_k^{\frac{|I_j|}{|r_k|}}$.

First we show S is i.o. FS-deep by examining prefixes of the form $S_1 S_2 \cdots S_j$, for j even. Let $k \geq 4$ and suppose k is devoted to S_j. Then by Lemma 2

$$D^k_{\text{FS}}(S_1 S_2 \cdots S_j) \geq D^{3k}_{\text{FS}}(S_1 S_2 \cdots S_{j-1}) + \frac{|S_j|}{|r_k|} D^{3k}_{\text{FS}}(r_k) \geq \frac{|S_j|}{|r_k|}(|r_k| - 4k).$$

For all $r \in \{0,1\}^*$, define the single state FST $T_r = (\{q_0\}, q_0, \delta, \nu)$, where for $b \in \{0,1\}$, $\nu(q_0, b) = r$. Let k' be large enough so that $I_{\text{FS}}, T_{r_k} \in \text{FST}^{\leq k'}$. Hence $D^{k'}_{\text{FS}}(S_j) \leq \frac{|S_j|}{|r_k|}$ and $D^{k'}_{\text{FS}}(S_1 \cdots S_{j-1}) \leq |S_1 \cdots S_{j-1}|$. Let $\hat{k}$ be from Lemma 3 such that

$$D^{\hat{k}}_{\text{FS}}(S_1 S_2 \cdots S_j) \leq 2D^{k'}_{\text{FS}}(S_1 \cdots S_{j-1}) + D^{k'}_{\text{FS}}(S_j) + 2 \leq 2|S_1 S_2 \cdots S_{j-1}| + \frac{|S_j|}{|r_k|} + 2.$$

Therefore for infinitely many prefixes of the form $S_1 S_2 \cdots S_j$ where S_j is devoted to k

$$\begin{aligned}
D^k_{\text{FS}}(S_1 S_2 \cdots S_j) - D^{\hat{k}}_{\text{FS}}(S_1 S_2 \cdots S_j) &\geq \frac{|S_j|}{|r_k|}(|r_k| - 4k - 1) \\
&\quad - 2|S_1 S_2 \cdots S_j| - 2 \\
&= |S_j|(1 - \frac{4k}{|r_k|} - \frac{1}{|r_k|}) - 2\log|S_j| - 2 \\
&\geq |S_j|(1 - \delta) \qquad (\delta > 17\epsilon) \\
&\geq |S_1 S_2 \cdots S_j|(1 - \alpha) \qquad (\alpha > \delta)
\end{aligned}$$

as $\frac{4k}{|r_k|}$ is maximum for $k = 4$.

For $0 \leq k \leq 3$, the above result follows from the fact that $D^0_{\mathrm{FS}}(S \upharpoonright m) \geq \ldots \geq D^3_{\mathrm{FS}}(S \upharpoonright m) \geq D^4_{\mathrm{FS}}(S \upharpoonright m)$, when we take $S \upharpoonright m = S_1 \cdots S_j$ to be such that S_j is devoted to $k = 4$.

Furthermore S has a non-zero finite-state strong dimension as for $k \geq 4$, there exists infinitely many prefixes of the form $S \upharpoonright m = S_1 \cdots S_j$ such that

$$D^k_{\mathrm{FS}}(S \upharpoonright m) \geq |S_j|(1 - \frac{4k}{|r_k|}) \geq |S_j|(1 - 16\epsilon) = (m - \log|S_j|)(1 - 8\epsilon) \geq m(1 - \hat{\epsilon}),$$

where $\hat{\epsilon} > 16\epsilon$ for j large. Therefore S is i.o. finite-state deep.

Next we show S is not a.e. ILPDC-deep.

Let $C_1, C_2, \ldots$ be an enumeration of all ILPDCs such that for a pair of ILPDCs C_p, C_q, if $p \leq q$, C_q has at least as many states as C_p. As the number of machines with k states is bigger than k, we can say that C_k has at most k states.

Henceforth we assume j is odd and examine S_j. Each odd j can be written in the form $2^k - 1 + n2^{k+1}$. Then for ILPDC C_k, C_k is devoted to every interval of the form $2^k - 1 + n2^{k+1}$, $n \geq 0$.

If S_j is devoted to C_k, we set S_j to be a string y of length $|I_j|$ from Lemma 1 that satisfies

$$|C_k(S_1 \cdots S_j)| - |C_k(S_1 \cdots S_{j-1})| \geq |S_j| - \log k - 1.$$

Say $S \upharpoonright m = S_1 \cdots S_j$. Hence we have that for $\beta_1, \beta_2 > 0$, for j large

$$\begin{aligned} |C_k(S \upharpoonright m)| &\geq |S_j| - \log k - 1 > |S_j|(1 - \beta_1) \\ &= (m - O(\log m))(1 - \beta_1) = m(1 - \beta_2). \end{aligned}$$

Hence for all k, for j large and for infinitely many prefixes of the form $S_1 \cdots S_j$ we have

$$|I_{PD}(S_1 \cdots S_j)| - |C_k(S_1 \cdots S_j)| < |S_1 \cdots S_j| - |S_1 \cdots S_j|(1 - \beta_2) = |S_1 \cdots S_j|\beta_2.$$

As β_1, β_2 can be made arbitrarily small, S is not a.e. ILPDC-deep.

□

6 Final Remarks

We introduced pushdown depth and showed our notion is a well behaved depth notion that satisfies all basic depth properties. We showed that is different from i.o. finite-state depth [6]. This gives more weight to the thesis that there is no perfect depth notion, but rather a "best for the job" notion.

It would be interesting to see whether a converse can be proven, i.e. a sequence that is ILPDC-deep but not i.o. finite state deep.

Acknowledgements. The authors would like to thank the anonymous referees for their useful comments, specifically to explore how the chosen binary representations of FSTs affects FS-depth.

References

1. Antunes, L., Fortnow, L., van Melkebeek, D., Vinodchandran, N.: Computational depth: concept and applications. Theoret. Comput. Sci. **354**, 391–404 (2006)
2. Bennett, C.H.: Logical depth and physical complexity. In: Bennett, C. (ed.) The Universal Turing Machine, A Half-Century Survey, pp. 227–257. Oxford University Press, New York (1988)
3. Calude, C.S., Salomaa, K., Roblot, T.K.: Finite state complexity. Theoret. Comput. Sci. **412**(41), 5668–5677 (2011)
4. Calude, C.S., Staiger, L., Stephan, F.: Finite state incompressible infinite sequences. Inf. Comput. **247**, 23–36 (2016)
5. Dai, J., Lathrop, J., Lutz, J., Mayordomo, E.: Finite-state dimension. Theoret. Comput. Sci. **310**, 1–33 (2004)
6. Doty, D., Moser, P.: Feasible depth. In: Cooper, S.B., Löwe, B., Sorbi, A. (eds.) CiE 2007. LNCS, vol. 4497, pp. 228–237. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-73001-9_24
7. Huffman, D.A.: Canonical forms for information-lossless finite-state logical machines. IRE Trans. Circ. Theory CT-6 (Special Supplement) **5**(5), 41–59 (1959)
8. Kohavi, Z.: Switching and Finite Automata Theory, 2nd edn. McGraw-Hill, New York (1978)
9. Mayordomo, E., Moser, P., Perifel, S.: Polylog space compression, pushdown compression, and Lempel-Ziv are incomparable. Theory Comput. Syst. **48**(4), 731–766 (2011)
10. Moser, P.: Polynomial depth, highness and lowness for E. Inf. Comput. (2019, accepted)
11. Moser, P.: On the polynomial depth of various sets of random strings. Theor. Comput. Sci. **477**, 96–108 (2013)
12. Moser, P., Stephan, F.: Depth, highness and DNR degrees. Discrete Math. Theor. Comput. Sci. **19**(4) (2017)