LOWNESS AND LOGICAL DEPTH

ROD DOWNEY AND MICHAEL MACINERNEY AND KENG MENG NG

1. Abstract

Bennett's concept of logic depth [3] seeks to capture the idea that a language has a lot of useful information. Thus we would expect that neither sufficiently random nor sufficiently computationally trivial sequences are deep. A question of Moser and Stephan [11] explores the boundary of this assertion, asking if there is a low computably enumerable (Bennett) deep language. We answer this question affirmatively by constructing a superlow computably enumerable Bennett deep language.

2. Introduction

Which sets (sequences/languages) contain a lot of information? When is this information useful? The area of algorithmic information theory would suggest that a random set would have a lot of information, but a sufficiently random set would have very little *useful* information. In [3], Bennett introduced a computational method of assigning meaning to having a lot of useful information.

Bennett's intuition was that sets with a lot of useful information, *deep sets*, were those with the following property. A set should be deep is one for which the more time a compressor is given the more the compressor can compress the sequence. That is, in no computably time bounded way, can we understand the complexity of the sets initial segments.

To be more precise,

Definition 2.1 (Bennett [3])**.** *Let K denote prefix-free Kolmogorov complexity*[1]*, and K^t be a time bounded version, for a computable time bound $t : \mathbb{N} \to \mathbb{N}$.*

*We say that a language L is (Bennett)-*deep *(or simply "deep" when the context is clear) if for each constant c and each computable time bound t, for almost all n,*

$$K^t(L \upharpoonright n) - K(L \upharpoonright n) > c.$$

The first author is partially supported by the Marsden Fund of New Zealand. The second author is partially supported by MOE2015-T2-2-055. The third author is partially supported by MOE-RG26/13 and MOE2015-T2-2-055.

[1] We assume that the reader is familiar with the basics of Kolmogorov Complexity, and refer the reader to Downey and Hirschfeldt [5], Li and Vitanyi [9] or Nies [13] for background material.

Here $A \upharpoonright n$ denotes the initial segment of A consisting of the first $n+1$ bits, following the notation of Soare [14].

Bennett proved that as we would expect, computable languages and sufficiently random ones are shallow, that is, not deep. The notion of depth has proven quite fruitful in giving insight into intrinsic information in languages, and several further variations on the notion, mainly involving orders (in place of c) and plain complexity in place of K) have been studied. See, for instance, [1, 2, 4, 8, 9, 10], etc. As Moser [10] showed, *all* of these notion have a common interpretation in terms of computable time bounds and compression ratios.

The goal of our paper is to answer a question raised in Moser and Stephan [11]. In [11], those authors gave a systematic analysis of the computational power of sets (as measured by the apparatus of classical computability theory, using tools like the jump operator), against notions of logical depth.

For example, Moser and Stephan extended an earlier result of Bennett by showing that a degree $\mathbf{a}$ is is high (meaning $\mathbf{a}' \geq \mathbf{0}''$) if and only if $\mathbf{a}$ contains a "strongly" deep set; one with depth ration ϵn.

One key property of deep sets is that easy sets should not be deep. Bennett proved that computable sets (and 1-random sets) are shallow, although there can be deep computably enumerable sets like the halting problem. Moser and Stephan showed that all *K-trivial* sets are shallow, where A is K-trivial iff $K(A \upharpoonright n) \leqslant^+ K(n+1)$ for all n. K-trivial sets resemble computable sets in terms of Kolmogorov complexity. They are also low in that if A is K-trivial then $A' \equiv_T \emptyset'$. In fact, that are all *superlow* in that $A' \equiv_{tt} \emptyset'$, where this denoted truth-table equivalence. (Nies [12, 13], also Downey and Hirschfeldt [5], and Kučera and Terwijn [7] for a related concept).

On the other hand it was known that, at least in terms of Kolmogorov complexity, there are deep sets quite close to being computable , at least in terms of Kolmogorov complexity. That is, Lathrop and Lutz [8] showed that there are *ultracompressible* deep sets. A is ultracompressible if and only if for all computable orders[2] g,

$$K(A \upharpoonright n) \leqslant^+ K(n+1) + g(n+1).$$

For sets in general, Moser and Stephan showed that PA degrees contain deep sets, and hence there are superlow deep sets by the Superlow Basis Theorem.

The question Moser and Stephan raise is whether such low deep sets can be computably enumerable. The thing is that enumerability has a big effect on the initial segment complexity of sets. For instance, there are superlow 1-random sets R and hence superlow sets with $K(R \upharpoonright n) \geq^+ n$ for all n, but if A is c.e. then $K(A \upharpoonright n) \leqslant^+ 3\log n$. Moreover, a recurrent theme in classical computability theory is that low c.e. sets have many properties very much like computable sets. (Soare [14] CH IX.3: "Low sets Resemble

[2]That is, $g(n)$ is nondecreasing and is unbounded.

Recursive Sets") So it would be reasonable to guess that all low c.e. sets are shallow. Nevertheless, we will prove the following.

Theorem 2.2. *There is a superlow c.e. Bennett deep set.*

The remainder of this paper is devoted to proving Theorem 2.2. Notation is more or less standard and generally follows Soare [14] or Downey-Hirschfeldt [5].

3. The Proof

Proof. We construct a c.e. set A. To make A Bennett deep, we meet for every $i \in \omega$ the requirement

R_i : *if* φ_i *is an order function, then*

$$(\forall c)(\forall^\infty m)\ K^{\varphi_i}(A \upharpoonright m) > K(A \upharpoonright m) + c,$$

where $\langle \varphi_i \rangle_{i<\omega}$ is an acceptable listing of all partial computable functions. We assume that we have some approximation $\langle \varphi_{i,s} \rangle_{s<\omega}$ to each φ_i such that for all s, the domain of $\varphi_{i,s}$ is an initial segment of ω. To make A low, we meet for every $e \geqslant 1$ the requirement

$$L_e : (\exists^\infty s)(\Phi_e^A(e)[s]\downarrow) \implies \Phi_e^A(e)\downarrow$$

where $\langle \Phi_e \rangle_{e<\omega}$ is an acceptable listing of all Turing functionals. We will later show that A is superlow by computably bounding the number of injuries to each L-requirement.

We first consider the strategy to meet the R-requirements without any L-requirements. We follow an approach from [6], where it is shown that every high degree contains a Bennett deep set.

We partition ω into consecutive intervals $I_0, I_1, \dots$ where interval I_j has length 2^j. We assign partial computable functions to intervals in the following way. Assign φ_0 to every second interval including the first one, φ_1 to every second interval including the first one of the *remaining* intervals, and so on for $\varphi_2, \varphi_3, \dots$. This way, φ_i will be assigned to every 2^{i+1}th interval. Therefore, if φ_i is assigned to I_j, then $I_{j+2^{i+1}}$ is the least interval above I_j to which φ_i is also assigned. We often write I_{j^+} instead of $I_{j+2^{i+1}}$. If φ_i is assigned to I_j, then we will also say that I_j is *dedicated* to R_i.

Suppose that φ_i is an order function. For each interval I_j to which φ_i is assigned, we would like to enumerate numbers from I_j into A in such a way that the complexity at time φ_i of $A \cap I_j$, considered as a string, is as high as possible. Then because the lengths of the intervals are rapidly increasing, and the intervals to which φ_i is assigned occur regularly, we will be able to show that almost every initial segment of A has high complexity at time φ_i, and so R_i is met.

More precisely, for the interval I_j, we look above to the interval I_{j^+}. We wait until a stage s where we see $\varphi_i(\max I_{j^+})[s]\downarrow$. Then at stage s, we choose the leftmost string τ of length $|I_j|$ which maximises $K_{\varphi_i(\max I_{j^+})}(\tau)$, and enumerate numbers from I_j into A so that $A_s \upharpoonright\upharpoonright \max I_j = A_{s-1} \upharpoonright \min I_j \hat{}\, \tau$. We say that we *move* in I_j at stage s. We will show that there are constants c_i and d_i such that for sufficiently large j, if m is such that $\max I_j < m \leqslant \max I_{j+}$, and we move in I_j at some stage, then $K^{\varphi_i}(A \upharpoonright\upharpoonright m) \geqslant c_i m - d_i$. It is important to note that moving in I_j will not allow us to given a lower bound on $K^{\varphi_i}(A \upharpoonright\upharpoonright m)$ for $m \in I_j$, but only for m such that $\max I_j < m \leqslant \max I_{j+}$. We make A c.e., and so there is a constant d such that for all $m \in \omega$, $K(A \upharpoonright\upharpoonright m) \leqslant 4\log(m+1) + d$. Therefore, the limit infimum as m tends to infinity of the difference between the true complexity $K(A \upharpoonright\upharpoonright m)$ and the time-bounded complexity $K^{\varphi_i}(A \upharpoonright\upharpoonright m)$ is infinite, and R_i will be met.

We now consider how this strategy could cope with the introduction of finitely many lowness requirement $L_1, \ldots, L_n$. Suppose at stage s we see $\Phi_e^\sigma(e)[s]\downarrow$ for some $\sigma \prec A_{s-1}$. At some later stage t we see that $\varphi_i(\max I_{j+})[t]\downarrow$ for some interval I_j such that $\sigma \succ A_{t-1} \upharpoonright \min I_j$. We say that I_j is *restrained* by L_e at stage t. We would like to move in I_j at stage t, but doing so would destroy the computation $\Phi_e^\sigma(e)$ and injure L_e. We are only allowed to destroy $\Phi_e^A(e)$ computations finitely many times, so we must eventually respect the restraint from a lowness requirement. Notice though that a lowness requirement imposes only finite restraint on A. In this simplified case with only finitely many lowness requirements, we simply respect each restraint; eventually there will be no further restraint on A, we will be able to move in almost every interval, and the strategy from above will succeed.

The situation is much more complicated with infinitely many lowness requirements. Now, the L-requirements will attempt to impose restraint cofinally along A. If we simply respect each restraint, then we will make A computable, and so will not be able to make A Bennett deep. Therefore, we need a strategy that will sometimes injure L-requirements in order to move, while still injuring each L-requirement only finitely often.

We arrange the L-requirements in the priority ordering

$$L_1 < L_2 < \cdots < L_e < \cdots .$$

We must from time to time respect the restraint from an L-requirement, while making

$$\liminf_{m\to\infty} K^{\varphi_i}(A \upharpoonright\upharpoonright m) - K(A \upharpoonright\upharpoonright m) = \infty.$$

Our idea is that we will not move in an interval I_j restrained by the L-requirement L_e if $e > i$, and if we are able to make the difference between $K^{\varphi_i}(A \upharpoonright\upharpoonright m)$ and $K(A \upharpoonright\upharpoonright m)$ at least e. We will attempt to do so by using the KC theorem to actively compress the initial segments of A. If I_j is restrained by L_e and $e \leqslant i$, then we will respect the restraint, and neither compress strings because of restraint from L_e, nor move in I_j even if we would like to. The L-requirements with index less than i will impose only finitely much restrain on A, and so we will be able to act in all but finitely many intervals dedicated to R_i.

So suppose as above that at stage s we see $\Phi_e^\sigma(e)[s]\downarrow$ for some $\sigma \prec A_{s-1}$, and at some later stage t we see that $\varphi_i(\max I_{j^+})[t]\downarrow$ for some interval I_j such that $\sigma \succ A_{t-1} \upharpoonright \min I_j$. Suppose that we moved in the previous interval dedicated to R_i. Then the strings we need to compress are the initial segments of σ of length greater than $\max I_j$. We enumerate a set of requests D. Suppose for the moment that we are only concerned with compressing strings due to restraint from L_e, so that we are willing to enumerate weight of 1 into our set D to compress these strings. Also suppose we have $\varphi_i(|\sigma|)[t]\downarrow$. We let

$$N_t = \{\nu : \nu \prec A_{t-1} \wedge \max I_j < |\nu| \leqslant |\sigma|\}.$$

The weight of these strings at time φ_i is

$$w_t = \sum_{\nu \in N_t} 2^{-K^{\varphi_i}(\nu)}.$$

In order to compress each of these strings by e bits, we would need to enumerate the request $(K^{\varphi_i}(\nu) - e, \nu)$ into D for every $\nu \in N_t$. In doing so, we would enumerate weight of $2^e.w_t$ into D. Therefore, if $2^e.w_t \leqslant 1$, then we are able to enumerate requests into D and use the KC theorem to compress these strings by e bits. If $2^e.w_t > 1$, then we are unable to compress these strings. In this case, we would like to fall back on the first strategy and move in I_j, but as this would injure L_e, we need some way to guarantee that we injure L_e at most finitely many times.

The key is the following. Because we have not moved in I_j by the beginning of stage t, we have not yet enumerated any numbers from I_j into A, and $A_{t-1} \cap I_j = \emptyset$. Then because $\sigma \succ A_{t-1} \upharpoonright \min I_j$, each string in N_t extends $A_{t-1} \upharpoonright \min I_j \,\hat{}\, 0$. When we move in I_j at stage t, we make sure to enumerate $\min I_j$ into A. As we make A c.e., no later approximation to A will extend $A_{t-1} \upharpoonright \min I_j \,\hat{}\, 0$. Then the weight $2^e.w_t$ is "lost" forever, in the following sense.

Suppose at some later stage u that L_e restrains an interval $I_{j'}$ above I_j. Let $\sigma' \prec A_{u-1}$ be least such that $\Phi_e^{\sigma'}(e)[u]\downarrow$. Then we consider the set N_u of initial segments of σ' of length greater than $\max I_{j'}$. Because each string in N_u extends $A_{u-1} \upharpoonright \min I_j \,\hat{}\, 1$, the sets N_u and N_t are disjoint. Therefore, the descriptions that the universal prefix-free machine $\mathcal{U}$ used to describe the strings in N_t cannot be used to describe the strings in N_u. So for $\mathcal{U}$ to describe the strings in N_u, it must add more weight in addition to the weight w_t already used. The weight of the domain of $\mathcal{U}$ is at most 1, and so if $\mathcal{U}$ loses weight at least w_t every time we injure L_e, then we can injure L_e at most $(w_t)^{-1}$ many times. We define a *threshold* $l_e = 2^{-e}$ for L_e. If the weight w_t we calculate is less than or equal to the threshold, then $2^e.w_t \leqslant 1$, and we can compress the set N_t by e bits. If the weight w_t is greater than the threshold, then we decide to move.

Now that we have compressed σ, as well as some of its initial segments, we may later want to act in intervals dedicated to R_i which are above σ. Suppose that I_k is the least interval dedicated to R_i with $\min I_k > |\sigma|$. If we see $\varphi_i(\max I_{k^+})[u]\downarrow$ at some later stage u, and no L-requirement restrains I_k at stage u, then we will want to move in I_k. However, we have no way of ensuring the difference between $K(\nu)$ and $K^{\varphi_i}(\nu)$ for strings $\nu \prec A_{u-1}$ with $|\sigma| < |\nu| \leqslant \max I_k$ is bounded below. The solution to this is the following. At the stage t where we compress σ and its initial segments, we will make sure that we have already seen $\varphi_i(\max I_{k^+})$ converge. Then we compress *all* strings $\nu \prec A_{t-1}$ such that $\max I_j < |\nu| \leqslant \max I_k$, and move in I_k at stage t. Then we can either move in I_{k^+} or compress strings above $\max I_{k^+}$ at some later stage, without having to worry about intervals below. Compressing these extra strings will not interfere with our way of ensuring the number of injuries to L_e is bounded.

We must now decide how to handle all L-requirements. Suppose that I_j is dedicated to R_i, and at stage s we see $\varphi_i(\max I_{j^+})[s]\downarrow$. If there is no L-requirement which restrains I_j at stage s, then we move in I_j. If there is some L_e which restrains I_j at stage s, we let $e = e_s$ be the least such. We now have a threshold for every L-requirement. Then, using the treshold l_e for L_e, we decide whether we would like to stay and compress a set of strings due to L_e, or move in I_j and injure L_e at stage s. Choosing e_s to be the least e such that L_e restrains I_j at stage s will allow us to ensure that if we do move in I_j and injure L_e at stage s, then no L-requirement of stronger priority than L_e is injured at stage s. This will be important when it comes to verifying that A is superlow.

Suppose that we compress some strings due to L_e at stage s and also move in some further interval, as described above. We say that I_j is *happy* at the end of stage s. Now suppose some L_d with $d < e$ restrains I_j at some later stage $t > s$. We consider the set N_t of strings we would like to compress, and decide using the threshold l_d for L_d whether we would like to stay and compress strings due to L_d, or move in I_j and injure L_d at stage t. In either case, I_j will again be happy at the end of stage t.

In general, we say that I_j is happy at stage s if we have either moved in I_j, or by the beginning of stage s, if e is least such that L_e restrains I_j at stage s, we have compressed all necessary strings due to L_e, and moved in the following interval. In full, if $\sigma \prec A_{s-1}$ is least such that $\Phi_e^\sigma(e)[s]\downarrow$, and k is least such that I_k is dedicated to R_i and $\min I_k > |\sigma|$, then we have compressed all strings $\nu \prec A_{s-1}$ such that $\max I_j < |\nu| \leqslant \max I_k$ by e bits, and moved in I_k.

The goal of the construction can then be summarised rather simply: if the interval I_j dedicated to R_i is unhappy at some stage, and we have seen enough convergence of φ_i, we act to make I_j happy.

We now turn to the definition of the thresholds l_e for all $e \geqslant 1$. When considering all L-requirements, we will still want to enumerate a single set D of requests. We must of course make sure that the weight of D is less than 1. To do so, we will set aside weight of 2^{-e} in D to requests that we enumerate when L_e is the L-requirement of strongest priority which restrains us. If we can manage to stick to this, then D will have weight less than 1.

We calculate these thresholds inductively, beginning with l_1. Recall that if L_1 restrains some interval I_j, then we will only want to act in I_j if I_j is dedicated to R_0. If I_j is dedicated to R_0 and L_1 restrains I_j at stage s, then will consider the set of strings N_s as above, and calculate its weight at time φ_0. We will wish to compress each string in N_s by 1 bit, and have set aside weight of 2^{-1} in our set D in order to do so. If the set N_s has weight w_s, then the weight we enumerate into D would be $2.w_s$. Therefore, if $w_s < 2^{-1}.2^{-1}$, then we will be able to compress each string in N_s by 1 bit while enumerating weight at most 2^{-1} into D. Therefore, we set $l_1 = 2^{-2}$.

Calculating l_2 is much more involved. Now if L_2 restrains some interval I_j, then we will want to act in I_j if I_j is either dedicated to R_0 or R_1. Suppose that I_{j_0} is dedicated to R_0 and that at stage s we see that L_2 is the strongest priority L-requirement which restrains I_{j_0}. At some later stage t we see $\varphi_0(\max I_{j_0+})[t]\downarrow$. Let $\sigma_2 \prec A_{t-1}$ be least such that $\Phi_2^{\sigma_2}(2)[t]\downarrow$. We consider the set N_t as usual, and then calculate the weight w_t of these strings at time φ_0. If we do not move in I_{j_0}, then we wish to compress each string in N_t by 2 bits. To compress each string in N_t by 2 bits, we will need to enumerate weight of $2^2.w_t$ into D. Suppose we naively calculate l_2 based on the method we used before to calculate l_1. We have set aside weight of 2^{-2} in our set D in order to compress strings when L_2 is the strongest priority L-requirement which restrains us. So we set $l_2 = 2^{-2}.2^{-2}$.

Let's say that we do compress the strings in N_t at stage t. At some much later stage u, we see that I_{j_1}, an interval dedicated to R_1, is also restrained by L_2, and that $\varphi_1(\max I_{k^+})[u]\downarrow$, where I_k is the first interval dedicated to R_1 with $\min I_k > |\sigma_2|$. We will then want to act in I_{j_1} at stage u. As usual, we consider the set N_u of strings we would like to compress. Because I_{j_0} and I_{j_1} are both restrained by L_2, we have already compressed many of the strings in N_u at stage t. Note though that the values of $\varphi_1(m)$ for $m \leqslant |\sigma_2|$ may be much larger than the values of $\varphi_0(m)$. Therefore, the complexity of the strings ν in N_u at time φ_1, $K^{\varphi_1}(\nu)$, may be much lower than $K^{\varphi_0}(\nu)$. So the weight w_u of the strings in N_u measured at time φ_1 may be much larger than the weight w_t. If $w_u \geqslant l_2$, then we will want to move in I_{j_1} at time u. If $w_u < l_2$, then we will want to compress the strings in N_u by 2 bits. Both situations are bad for us. If we do want to move, then this will make I_{j_0} unhappy, and furthermore, we have "wasted" some of the weight in D, in that the strings in N_t are no longer all initial segments of A_u. If we do compress the strings in N_u, then we will end up enumerating more than the agreed upon weight of 2^{-2} into D.

We will instead define two thresholds, $l_{0,2}$ and $l_{1,2}$. It is not important what the priority ordering between R_0 and R_1 is. (Indeed, we will not define a priority ordering between the R-requirements.) Rather, what is important is the order in which the functions converge. We use the threshold $l_{0,2}$ when the first of the functions φ_0 and φ_1 converges, and the threshold $l_{1,2}$ when the second converges.

Assume for the moment that we never see restraint from L_1. Suppose we see the sequence of events as before, but now use the two thresholds. Then at stage t we see that $w_t \leqslant l_{0,2}$, and we compress the set N_t. If at stage u we also have $w_u \leqslant l_{1,2}$, then we will want to compress the strings in N_u. If we do have $w_u \leqslant l_{1,2}$ and compress the strings in N_u, then we will not need to act in another interval restrained by L_2 again. This is because at stage t, every interval dedicated to R_0 below $|\sigma_2|$ is made happy when we compress the strings in N_t, and at stage u, every interval dedicated to R_1 below $|\sigma_2|$ is made happy when we compress the strings in N_u. Of the weight 2^{-2} in D that we set aside for compressing strings when L_2 is the strongest priority L-requirement which restrains us, we reserve half for compressing strings when the second order function converges. Therefore, we would like $l_{1,2}$ to be such that $2^2.l_{1,2}$, the upper bound on the weight we would enumerate into D, is at most $2^{-1}.2^{-2}$. So we set $l_{1,2} = 2^{-2}.2^{-1}.2^{-2}$.

We must be careful that the amount of weight that we "waste" as above is small. So suppose at stage t we see that $w_t \leqslant l_{0,2}$ (whatever value this may be) and we compress the set N_t, but at stage u we see that $w_u \geqslant l_{1,2}$. We move at stage u, and will potentially waste all the weight w_t. Because we move only when we see weight of at least $l_{1,2}$, we can use the same reasoning as before to show that we can do this at most $(l_{1,2})^{-1}$ many times. Looking ahead to calculating the thresholds for larger values of e, of the weight 2^{-2} in D that we set aside for when restrained by L_2, we reserve 2^{-2} for when the first function converges. We enumerate weight of $2^2.l_{0,2}$ into D every time we compress strings when the first order function converges, and can be interrupted at most $(l_{1,2})^{-1}$ many times. Then including weight we may enumerate before we are interrupted the first time, and weight we enumerate after we are interrupted for the last time, we enumerate weight of at most $(1 + (l_{1,2})^{-1}).2^2.l_{0,2}$ into D when compressing strings when the first order function converges. So we would like $l_{0,2}$ to be such that $(1 + (l_{1,2})^{-1}).2^2.l_{0,2} \leqslant 2^{-2}.2^{-2}$, and we let $l_{0,2}$ be some rational number which satisfies this inequality.

Assuming that we never see restraint from any L-requirement of stronger priority than L_e, then we calculate the thresholds for L_e in much the same way. Now that we allow any interval dedicated to any requirement R_i with $i < e$ to act if restrained by L_e, we have e many thresholds $l_{0,e}, l_{1,e}, \ldots, l_{e-1,e}$. We have set aside weight of 2^{-e} for compressing strings when L_e is the strongest priority L-requirement which restrains us, and of this weight, we reserve 2^{e-c} for when the c^{th} order function converges. We first calculate $l_{e-1,e}$ like we did for $l_{1,2}$, and then use these values to calculate the rest of the thresholds recursively until we get to $l_{0,e}$.

Now suppose that we have been moving and compressing strings when restrained by L_e, and later see that some interval I_j, in which we have acted and compressed strings, is restrained by L_d with $d < e$. Then I_j will become unhappy, and we will either move in I_j, or compress some strings below the restraint imposed by L_d. Both actions will require us to move in some interval: either we move in I_j, or we compress some set of strings and move in the following interval. Moving in an interval will again mean that some of the strings we have already compressed when we were restrained by L_e are now of no use to us, and so are wasted. However, we can compute an upper bound on the number of times that we may act when L_e is the strongest L-requirement which restrains us. Whenever we move when L_e is the strongest L-requirement which restrains us, we see some weight which is lost forever. As $l_{0,e}$ is the smallest threshold of those we use when L_e is the strongest L-requirement which restrains us, we lose a set of weight at least $l_{0,e}$ every time we move. Therefore, we can do this at most $(l_{0,e})^{-1}$ many times. If we compress strings when L_e is the strongest L-requirement which restrains us, then we can only do this for the sake of some R-requirement R_i with $i < e$. We can then use these facts to compute an upper bound on the number of times we can act when L_e is the strongest L-requirement which restrains us.

Suppose that a_e is an upper bound on the number of times we may act when restrained by any L-requirement of stronger priority than L_e. We will want to take this into account when defining the thresholds for L_e. Before, we enumerated weight of at most 2^{-e} when L_e was the strongest priority L-requirement that restrained us. If this could be wasted every time we act for an L-requirement of stronger priority than L_e, then including weight we enumerate before we are interrupted the first time we act for such an L-requirement, and the weight we enumerate after the last time we are interrupted when we act for such an L-requirement, we will want to enumerate weight of at most $(1 + a_e)^{-1}.2^{-e}$ into D when L_e is the strongest priority L-requirement that restrains us. This is the last concern we need to consider in the calculation of the thresholds.

There is one last change we make to the set of strings we compress. Suppose I_j is happy at the beginning of stage s because for L_e the strongest L-requirement which restrains I_j, $\sigma \prec A_{s-1}$ least such that $\Phi_e^\sigma(e)[s]\downarrow$, and k least such that I_k is dedicated to R_i and $\min I_k > |\sigma|$, we have compressed all strings $\nu \prec A_{s-1}$ such that $\max I_j < |\nu| \leqslant \max I_k$ by e bits, and moved in I_k. Because the intervals dedicated to R_i occur only once every 2^{i+1} intervals, there may be many intervals I_m with $\min I_m > |\sigma|$ that are below I_k. If we were to move in one of these intervals at some later stage t, then I_j would become unhappy, because then we would not have compressed all strings $\nu \prec A_{t-1}$ such that $\max I_j < |\nu| \leqslant \max I_k$. So we would like to act again in I_j and compress some strings. The problem is that the intervals I_m may be dedicated to R-requirements R_i with $i > e$, and so we would not be able to compute in advance a bound on the number of times we may need to act and compress strings when L_e is the strongest L-requirement which restrains us. We could choose to not act for any R-requirement R_i with $i > e$ in an interval in which we have already compressed strings due to L_e. Even with this restriction, we would need to act again in I_j if we moved in any interval I_m as above dedicated to some R-requirement R_i with $i < e$.

Instead, we simply compress *all* strings ν such that $\nu \succ \sigma$ and $|\nu| \leqslant \max I_k$. Then no matter how we move in intervals I_m with $\min I_m > |\sigma|$, I_j will be happy. Again, compressing these extra strings will not interfere with our way of ensuring the number of injuries to L_e is bounded.

4. Definitions

For each pair (c, e) with $e \geqslant 1$ and $c < e$, we define the threshold $l_{c,e}$. We do this by recursion. We let $l_{0,1} = 2^{-2}$. Now suppose that for all $d < e$, we have defined $l_{c,d}$ for all $c < d$. We let $a_e = \sum_{i=1}^{e-1}(i+2).(l_{0,i})^{-1}$. Let $l_{e-1,e}$ be the greatest rational number of the form 2^{-p} with $p \in \omega$ such that $l_{e-1,e} \leqslant 2^{-1}.(1+a_e)^{-1}.2^{-e}$. Suppose we have defined $l_{i,e}$ for some i with $0 < i < e$. We let $l_{i-1,e}$ be the greatest rational number of the form 2^{-p} with $p \in \omega$ such that $(1 + (l_{i,e})^{-1}).2^e.l_{i-1,e} \leqslant 2^{i-e}.(1+a_e)^{-1}.2^{-e}$.

If we say "move in I_j" at stage s of the construction, then we do the following. Suppose that φ_i is assigned to I_j. We will have $\varphi_i(\max I_{j^+})[s]\downarrow$. We run the universal prefix-free machine $\mathcal{U}$ on all inputs of length strictly less than $|I_j| - 1$ for $(\varphi_i(\max I_{j^+}))^3$ many steps each. Suppose τ is the leftmost string of length $|I_j| - 1$ that was not output during this procedure. We enumerate $\min I_j$ into A, and for all $x < |\tau|$, if $\tau(x) = 1$, then we enumerate $\min I_j + 1 + x$ into A. Note that $K_{\varphi_i(\max I_{j^+})^3}(\tau) \geqslant |I_j| - 1$.

If $e < s$ and $\Phi_e^\sigma(e)[s]\downarrow$ for some $\sigma \prec A_{s-1}$, then with σ the least such, we let $r_{e,s}$ be the maximum of $|\sigma|$, and the length of any string in any set N_t, where $t < s$ is some stage of the construction at which we acted in Case 2 with $e_t \leqslant e$. We say that I_j *is restrained by* L_e *at stage* s if $\min I_j < r_{e,s}$.

Suppose that the partial computable function φ_i is assigned to the interval I_j. We say that I_j is *open at stage* s if I_j is not restrained by any L_e with $e < i$ at stage s. We say that I_j is *happily restrained by* L_e *at stage* s if I_j is restrained by L_e at stage s, and by the beginning of stage s, if k is least such that φ_i is assigned to I_k and $\min I_k > r_{e,s}$, then we have compressed the strings $\nu \prec A_{s-1}$ such that $\max I_j < |\nu| \leqslant \max I_k$ for the sake of R_i due to L_e, and moved in I_k.

We say that I_j *is happy at stage* s if we have either moved in I_j before stage s, or for L_e the strongest priority L-requirement which restrains I_j at stage s, I_j is happily restrained by L_e at stage s.

We say that we *want to act in* I_j *at stage* s if I_j is open and not happy at stage s, and if φ_i is assigned to I_j, then either

(1) I_j is not restrained by any L-requirement at stage s, $\varphi_i(\max I_{j^+})[s]\downarrow$, and φ_i is nondecreasing on the interval $[0, \max I_{j^+}]$, or

(2) I_j is restrained by some L-requirement at stage s, and for
 (a) $L_{e_s} = L_e$ the strongest such L-requirement, and
 (b) $k_s = k$ the least such that φ_i is assigned to I_k and $\min I_k > r_{e,s}$,
we have $\varphi_i(\max I_{k^+})[s]\downarrow$, and φ_i is nondecreasing on the interval $[0, \max I_{k^+}]$.

5. The construction

Construction

Stage 0: Let $A_0 = \emptyset$ and let $D_0 = \emptyset$. We proceed to the next stage.

Stage s, $s \geqslant 1$: Let $j < s$ be least such that we want to act in I_j at stage s. (If there is no such j, we proceed to the next stage.) We say that I_j *receives attention at stage s*. Suppose that φ_i is assigned to I_j. There are two cases.

Case 1: We want to act in I_j at stage s and (1) applies. We move in I_j, and proceed to the next stage.

Case 2: We want to act in I_j at stage s and (2) applies. Let $L_{e_s} = L_e$ and $k_s = k$ be as above. Let

$$N_s = \{\, \nu : (\nu \prec A_{s-1} \wedge \max I_j < |\nu| \leqslant r_{e,s}) \vee \\ (\nu \succ A_{s-1} \restriction r_{e,s} \,\wedge\, |\nu| \leqslant \max I_k)\}$$

and let $w_s = \sum_{\nu \in N_s} 2^{-K^{\varphi_i}(\nu)}$. Let c_s be the number of requirements R_d with $d < e$ such that all open intervals which are dedicated to R_d and restrained by L_e are happy at the beginning of stage s. There are two subcases.

Subcase 2a: $w_s \leqslant l_{c_s,e}$. Then for every $\nu \in N_s$ we enumerate the request $(K^{\varphi_i}(\nu) - e, \nu)$ into D. For every such ν, we say that we have *compressed ν for the sake of R_i due to L_e*. We move in I_k.

Subcase 2b: $w_s > l_{c_s,e}$. We move in I_j.

If we act in Case 2, we also move in every open interval which was happy at the beginning of stage s, but is no longer happy after moving due to Case 2.

End of Construction

6. The verification

Recall the natural numbers e_s and c_s defined during the construction.

Lemma 6.1. *Let $c < e$. We can act in Subcase 2b of the construction at a stage s with $e_s = e$ and $c_s \geqslant c$ at most $(l_{c,e})^{-1}$ many times.*

Proof. Suppose that s and t are two such stages, with $s < t$. Consider the sets N_s and N_t. We will show that they are disjoint.

Suppose that I_{j_s} receives attention at stage s. As we have not moved in I_{j_s} before stage s, $A_{s-1}(\min I_{j_s}) = 0$. The strings in N_s all extend $A_{s-1} \upharpoonleft \max I_{j_s}$, and so must extend $A_{s-1} \restriction \min I_{j_s} \hat{\ } 0$. When we move at stage s, we enumerate $\min I_{j_s}$ into A. As A is c.e., for all $s' \geqslant s$, $A_{s'}$ does not extend $A_{s-1} \restriction \min I_{j_s} \hat{\ } 0$.

Suppose that I_{j_t} receives attention at stage t. The strings in N_t are either initial segments of A_{t-1} of length at least $\max I_{j_t}$, or properly extend $A_{t-1} \upharpoonright r_{e,t}$. If $\nu \in N_t$ is an initial segment of A_{t-1} of length less than or equal to $\min I_{j_s}$, then it cannot be in N_s, as all strings in N_s have length at least $\max I_{j_s}$. If $\nu \in N_t$ is an initial segment of A_{t-1} of length greater than $\min I_{j_s}$, then it cannot be in N_s, because ν must extend $A_{t-1} \upharpoonright\!\upharpoonright \min I_{j_s}$, which cannot extend $A_{s-1} \upharpoonright \min I_{j_s} \,\hat{}\, 0$. Lastly, suppose $\nu \in N_t$ properly extends $A_{t-1} \upharpoonright r_{e,t}$. By the choice of $r_{e,t}$, ν must be longer than any string in N_s. So ν cannot be in N_s. Therefore N_s and N_t are disjoint.

We act in Subcase 2b of the construction at stages s and t and $e_s = e_t = e$, and so we have $w_s, w_t > l_{c_s,e}$. The thresholds satisfy $l_{0,e} < l_{1,e} < \cdots < l_{e-1,e}$. As $c_s \geqslant c$, we have $l_{c_s,e} \geqslant l_{c,e}$. Therefore, if we act in Subcase 2b of the construction at more than $(l_{c,e})^{-1}$ many stages s with $e_s = e$ and $c_s \geqslant c$, then we will have more than $(l_{c,e})^{-1}$ many pairwise disjoint sets, each with weight greater than $l_{c,e}$. This contradicts the fact that $\sum_\sigma 2^{-K(\sigma)} < 1$. □

Lemma 6.2. *Suppose that L_e is the strongest priority L-requirement which restrains I_j at stage s, and I_j is happily restrained by L_e at stage s. Then I_j is happy at all later stages unless we either move in some interval restrained by L_e at some later stage, or see that some L_d with $d < e$ restrains I_j at some later stage.*

Proof. Suppose φ_i is assigned to I_j and k is least such that φ_i is assigned to I_k and $\min I_k > r_{e,s}$. Suppose at stage $s+1$ we neither move in any interval restrained by L_e, nor see some L_d with $d < e$ restrain I_j, but move in some interval I_m with $\min I_m > r_{e,s}$. If I_m is above I_k then it is clear that we have already compressed the strings $\nu \prec A_s$ such that $\max I_j < |\nu| \leqslant \max I_k$ for the sake of R_i due to L_e. Now consider the case where I_m is below I_k. Suppose we compressed $A_{s-1} \upharpoonright r_{e,s}$ for the sake of R_i due to L_e at stage $r \leqslant s$. Then at stage r we compressed all strings extending $A_{s-1} \upharpoonright r_{e,s}$ of length at most $\max I_k$. Therefore we have compressed all strings $\nu \prec A_s$ such that $\max I_j < |\nu| \leqslant \max I_k$ for the sake of R_i due to L_e. □

Lemma 6.3. *A is superlow.*

Proof. Suppose we act in Case 2 of the construction at some stage t. We choose some e_t. We show that we do not move in any interval restrained by any L-requirement of stronger priority than L_{e_t} at stage t. Suppose I_j receives attention at stage t.

First suppose that we act in Subcase 2a at stage t. Then we do not move in I_j at stage t. By the choice $k = k_t$ at stage t, I_k is not restrained by L_{e_t}. Suppose for contradiction that I_k is restrained by some L_d with $d < e_t$. As $j < k$, L_d must restrain I_j at stage t. But then $e_t \leqslant d$, which is a contradiction. We now consider the intervals which were happy at the beginning of stage t, but are not happy after we move in I_k. Suppose I_r is such an interval. Suppose for contradiction that the strongest priority L-requirement which restrains I_r is L_d with $d < e_t + 1$. Then by Lemma

6.2, I_k must be restrained by L_d. This is a contradiction. So the strongest priority L-requirement which restrains I_r is L_d for some $d \geqslant e_t + 1$.

Now suppose that we act in Subcase 2b at stage t. By the choice of e_t, the strongest priority L-requirement which restrains I_j is L_{e_t}. Suppose I_r is an interval which was happy at the beginning of stage t, but not after we moved in I_j. Suppose for contradiction that the strongest priority L-requirement which restrains I_r is L_d with $d < e_t$. Then by Lemma 6.2, I_j must be restrained by L_d. This is a contradiction. So the strongest priority L-requirement which restrains I_r is L_d for some $d \geqslant e_t$.

Suppose that after stage s^*, we do not act in Case 2 at a stage s with $e_s < e$. We calculate an upper bound on the number of stages $s > s^*$ at which we can injure L_e, and an upper bound on the number of stages $s > s^*$ at which we act in Case 2 with $e_s = e$.

We say that L_e is *injured* at stage s if there is some $\sigma \prec A_{s-1}$ such that $\Phi_e^\sigma(e)[s]\downarrow$, but $\sigma \not\prec A_s$. By the choice of $r_{e,s}$, in order to injure L_e at stage s, we must move in some interval that is restrained by L_e at stage s. Suppose we injure L_e at stage $s > s^*$. We must have $e_s = e$. By the choice of k_s, if we act in Subcase 2a of the construction at stage s and $e_s = e$, then we do not injure L_e at stage s. Therefore we must act in Subcase 2b at stage s. We have $c_s \geqslant 0$ at any such stage, and so by Lemma 6.1, we can injure L_e at a stage $s > s^*$ at most $(l_{0,e})^{-1}$ many times.

Suppose we do not move in any interval restrained by L_e between stages s_0 and s_1, where $s^* < s_0 < s_1$. We calculate an upper bound on the number of times we act in Case 2 at a stage s with $s_0 < s < s_1$ and $e_s = e$. As we do not move in any interval restrained by L_e between stages s_0 and s_1, if we do act at a stage s with $s_0 < s < s_1$ and $e_s = e$, then we must act in Subcase 2a at stage s. Suppose u with $s_0 < u < s_1$ is the first stage after stage s_0 at which we act in Subcase 2a with $e_u = e$. If I_j receives attention at stage u and φ_i is assigned to I_j, then we compress a set of strings for the sake of R_i due to L_e. At the end of stage u, each open interval I_m restrained by L_e that is dedicated to R_i is happy, and so we cannot act in any such interval at another stage before stage s_1. By the definition of *open*, the only intervals restrained by L_e which may later receive attention are those dedicated to R-requirements R_i with $i < e$. Therefore, we can act in Case 2 at a stage s with $s_0 < s < s_1$ and $e_s = e$ at most e many times. We could also act in Subcase 2a before we first move in an interval restrained by L_e, and after we last move in an interval restrained by L_e. So, as we can move in an interval restrained by L_e at a stage $s > s^*$ at most $(l_{0,e})^{-1}$ many times, we can act in Subcase 2a at a stage $s > s^*$ with $e_s = e$ at most $(e+1).(l_{0,e})^{-1}$ many times. Finally, including the stages in which we act in Subcase 2b, we can act in Case 2 at a stage $s > s^*$ with $e_s = e$ at most $(e+2).(l_{0,e})^{-1}$ many times.

We now calculate inductively a bound on the number of times we injure each L-requirement. Note that if we act in Case 1 of the construction at some stage s, then we cannot injure any L-requirement at stage s. As L_1

is the L-requirement of strongest priority, we could never act in Case 2 at a stage s with $e_s < 1$. Therefore, in order to injure L_1, we must act in Subcase 2b at a stage s with $e_s = 1$, and so by Lemma 6.1, we can injure L_1 at most $(l_{0,1})^{-1}$ many times. We injure L_2 either at a stage s in which we act in Case 2 with $e_s = 1$, of which there are at most $3.(l_{0,1})^{-1}$, or at a stage s with $e_s = 2$ and in which we act in Subcase 2b, of which there are at most $(l_{0,2})^{-1}$. Therefore, we injure L_2 at most $3.(l_{0,1})^{-1} + (l_{0,2})^{-1}$ many times. In general, we injure L_e at most

$$3.(l_{0,1})^{-1} + 4.(l_{0,2})^{-1} + \ldots + (e+1).(l_{0,e-1})^{-1} + (l_{0,e})^{-1}$$

many times, and we act in Case 2 at a stage s with $e_s \leqslant e$ at most

$$\sum_{i=1}^{e}(i+2).(l_{0,i})^{-1}$$

many times.

The function which takes the pair (c, e) with $e \geqslant 1$ and $c < e$ to the threshold $l_{c,e}$ is computable. So for all $e \geqslant 1$, we can computably bound the number of times L_e is injured, and A is superlow. □

Lemma 6.4. *The weight of D is less than 1.*

Proof. For all $e \geqslant 1$, let D_e be the set of requests $(l, \nu) \in D$ where ν was compressed due to L_e. We show that for all $e \geqslant 1$, the weight of D_e is at most 2^{-e}, which shows that $D = \cup_{e \geqslant 1} D_e$ has weight at most 1.

Let $a_1 = 0$, and for all $e \geqslant 2$, let $a_e = \sum_{i=1}^{e-1}(i+1).(l_{0,i})^{-1}$. Then for all $e \geqslant 1$, we can move in an interval restrained by L_e when we act in Case 2 at a stage s with $e_s < e$ at most a_e many times. For all $e \geqslant 1$ and all $k \leqslant a_e$, let $D_{e,k}$ be the set of requests in D_e that were enumerated at a stage s such that before s, there were k many stages t where we acted in Case 2 with $e_t < e$. We show that for all $k \leqslant a$, the weight of $D_{e,k}$ is at most $(1 + a_e)^{-1}.2^{-e}$. Then we will have that $D_e = \cup_{k \leqslant a_e} D_{e,k}$ has weight at most 2^{-e}.

Suppose that after stage s^*, we do not act in Case 2 at a stage s with $e_s < e$, and that before stage s^*, there were k many stages t where we acted in Case 2 with $e_t < e$. If we enumerate weight into D_e at some stage s, then we determine the natural number $c_s \leqslant e$. Let $D_{e,k,c}$ be the set of all requests in $D_{e,k}$ that were enumerated at a stage s at which $c_s = c$. We claim that we must have $c_s < e$, and that for all $c < e$, the set $D_{e,k,c}$ has weight at most $2^{c-e}.(1 + a_e)^{-1}.2^{-e}$. Then we will have that $D_{e,k} = \cup_{c<e} D_{e,k,c}$ has weight at most

$$\sum_{c<e} 2^{c-e}.(1 + a_e)^{-1}.2^{-e} < (1 + a_e)^{-1}.2^{-e}.$$

Suppose at stage $s > s^*$ that I_j requires attention, and that we act in Subcase 2a with $e_s = e$. Then I_j must be unhappy at the beginning of stage s. Suppose I_j is dedicated to R_i. As I_j is open at stage s, we must have

$i < e$. Then R_i is not included among the c_s many requirements at stage s, and $c_s < e$.

We now show that our claim above holds for $c = e - 1$. So suppose at stage $s > s^*$ that I_j requires attention, and that we act in Subcase 2a with $e_s = e$ and $c_s = e - 1$, and compress a set N_s of strings due to L_e. Then N_s has weight at most $l_{e-1,e}$, and we enumerate weight at most $2^e.l_{e-1,e}$ into $D_{e,k,e-1}$ at stage s. The threshold $l_{e-1,e}$ was chosen so that $2^e.l_{e-1,e} \leqslant 2^{-1}.(1 + a_e)^{-1}.2^{-e}$, so we enumerate weight at most $2^{-1}.(1 + a_e)^{-1}.2^{-e}$ into D_e at stage s. We show that we cannot act at a stage $t > s$ at which $e_t = e$, which then proves the claim above for $c = e-1$.

As $c_s = e-1$, there are $e-1$ many requirements R_d with $d < e$ such that before stage s, all open intervals which are dedicated to R_d and restrained by L_e are happy at the beginning of stage s. As I_j receives attention at stage s, I_j must be unhappy at the beginning of stage s. Furthermore, I_j is restrained by L_e at the beginning of stage s. Therefore R_i is not included among the c_s many requirements at stage s. We compress the set N_s for the sake of R_i due to L_e at stage s, and all intervals above I_j which are dedicated to R_i and restrained by L_e, as well as I_j itself, are happy at the end of stage s. As I_j receives attention at stage s, no open interval below I_j which is dedicated to R_i is unhappy at the beginning of stage s. Therefore, at the end of stage s, there are exactly e many requirements R_d such that all open intervals which are dedicated to R_d and restrained by L_e are happy at the end of stage s. As we showed above, we cannot act at a stage $t > s$ with $c_s = e$, and so we cannot act at a stage $t > s$ with $e_t = e$.

Now let $c < e-1$, and suppose at stage $s > s^*$ that I_j requires attention, and that we act in Subcase 2a with $e_s = e$ and $c_s = c$, and compress a set N_s of strings due to L_e. Then N_s has weight at most $l_{c,e}$, and we enumerate weight at most $2^e.l_{c,e}$ into D_e. Suppose that we do not move in any interval restrained by L_e after stage s. We show that if at some stage $t > s$ we have $e_t = e$, then we must have $c_t > c$.

As $c_s = c$, there are c many requirements R_d with $d < e$ such that before stage s, all open intervals which are dedicated to R_d and restrained by L_e are happy at the beginning of stage s. As I_j receives attention at stage s, I_j must be unhappy at the beginning of stage s. Furthermore, I_j is restrained by L_e at the beginning of stage s. Therefore R_i is not included among the c_s many requirements at stage s. We compress the set N_s for the sake of R_i due to L_e at stage s, and all intervals above I_j which are dedicated to R_i and restrained by L_e, as well as I_j itself, are happy at the end of stage s. As I_j receives attention at stage s, no open interval below I_j which is dedicated to R_i is unhappy at the beginning of stage s. Therefore, at the end of stage s, there are exactly $c+1$ many requirements R_d such that all open intervals which are dedicated to R_d and restrained by L_e are happy at the end of stage s. Suppose $t > s$ is least such that $e_t = e$. Then at the beginning of stage t, there are $c+1$ many requirements R_d such that all open intervals which are dedicated to R_d and restrained by L_e are happy, and $c_t = c + 1 > c$.

Therefore, in order to enumerate any more weight into $D_{e,k,c}$ after stage s, we must move in some interval restrained by L_e after stage s. Suppose we move in some interval restrained by L_e at some stage after s, and that $t > s$ is the least such. Then, as above, we have $c_t \geqslant c + 1$. By Lemma 6.1, we can move in some interval restrained by L_e at some stage u with $e_u = e$ and $c_u \geqslant c + 1$ at most $(l_{c+1,e})^{-1}$ many times. In between such stages u, we can enumerate weight at most $2^e.l_{c,e}$ into D_e. We could also enumerate this much weight into D_e before the first such stage u, and after the last such stage u. Therefore, we enumerate weight at most $(1 + (l_{c+1,e})^{-1}).2^e.l_{c,e}$ into $D_{e,k,c}$. By the choice of $l_{c,e}$, we have $(1 + (l_{c+1,e})^{-1}).2^e.l_{c,e} \leqslant 2^{c-e}.(1 + a_e)^{-1}.2^{-e}$. This establishes the claim. □

Lemma 6.5. *Suppose that φ_i is an order function. Then there are constants c_i and d_i such that the following holds. Suppose φ_i is assigned to the interval I_j and we move in I_j at some stage. Then if j is sufficiently large, and m is such that $\max I_j < m \leqslant \max I_{j^+}$, then*

$$K^{\varphi_i}(A \upharpoonright\upharpoonright m) \geqslant c_i m - d_i.$$

Proof. Let M be the following machine. On input γ, run the universal prefix-free machine $\mathcal{U}$ on input γ. If $\mathcal{U}(\gamma)\downarrow = \delta$ and $|\delta| = \max I_j$ for some j, then M outputs the string ρ such that $\delta \upharpoonright \min I_j \,\hat{}\, \rho = \delta$. Then $\mathcal{U}$ simulates the machine M, and if M runs in time t, then $\mathcal{U}$ simulates M in time $O(t^2)$. Suppose $\mathcal{U}$ simulates M in time at most et^2. Let f be the coding constant for M.

Let $u = \varphi_i(\max I_{j^+})$. We claim that if j is sufficiently large, then $K_u(A \upharpoonright\upharpoonright \max I_j) > \max I_j/3$. Let j be large enough so that $\varphi_i(n) \geqslant e$ for all $n \geqslant \max I_j$, and so that $\max I_j/3 + f \leqslant |I_j| - 2$. Such a j exists because φ_i is unbounded and nondecreasing. Then we have $(\varphi_i(\max I_m))^3 \geqslant e(\varphi_i(\max I_m))^2$ for all $m \geqslant j$. We move in I_j at stage s, and so if we move to the string $A_s \upharpoonright \min I_j \,\hat{}\, \tau$ at stage s, we have $K_{u^3}(\tau) \geqslant |I_j| - 1$. Suppose for contradiction that $l = K_u(A \upharpoonright\upharpoonright \max I_j) \leqslant \max I_j/3$. Then there is a string γ of length $l \leqslant \max I_j/3$ such that $\mathcal{U}_u(\gamma)\downarrow = A \upharpoonright\upharpoonright \max I_j$. As $\mathcal{U}$ simulates M in time et^2, and by the choice of j, there is a string γ' of length at most $l + f$ such that $\mathcal{U}_{u^3}(\gamma')\downarrow = \tau$. But then $K_{u^3}(\tau) \leqslant l + f \leqslant \max I_j/3 + f \leqslant |I_j| - 2$, which is a contradiction. This establishes the claim.

We claim that there is c_i such that for j and m as in the statement of the lemma, $c_i m \leqslant \max I_j/4$. Using the fact that $\max I_j = 2^{j+1} - 2$ and the fact that φ_i is assigned to every 2^{i+1}th interval, it is straightforward to verify that $c_i = 2^{-(2^{i+1}+2)}$ suffices.

We now show the conclusion of the lemma. Let N be the following machine. One input γ, run $\mathcal{U}$ on input γ. If $\mathcal{U}(\gamma)\downarrow = \delta$ and $|\delta| \neq \max I_k$ for any interval I_k to which φ_i is assigned, then we let k be greatest such φ_i is assigned to I_k and $\max I_k < |\delta|$. Then N outputs the string $\delta \upharpoonright\upharpoonright \max I_k$.

Suppose that if N runs in time t, then $\mathcal{U}$ simulates N in time at most gt^2. Let h be the coding constant for N.

Let j be large enough so that $\varphi_i(n) \geqslant g$ for all $n \geqslant \max I_j$. Then we have $(\varphi_i(\max I_m))^3 \geqslant g(\varphi_i(\max I_m))^2$ for all $m \geqslant j$. Such a j exists because φ_i is unbounded and nondecreasing. Suppose for contradiction that $l = K^{\varphi_i}(A \upharpoonright\upharpoonright m) < c_i m - h$. Then as $u \geqslant \varphi_i(m)$, we have $l = K^{\varphi_i}(A \upharpoonright\upharpoonright m) \leqslant K_u(A \upharpoonright\upharpoonright m) < c_i m - h$. Then there is a string γ of length $l < c_i m - h$ such that $\mathcal{U}_u(\gamma) = A \upharpoonright\upharpoonright m$. As $\mathcal{U}$ simulates N in time gt^2, and by the choice of j, there is a string γ' of length at most $l + h$ such that $\mathcal{U}_{u^3}(\gamma')\downarrow = A \upharpoonright\upharpoonright \max I_j$. But then $K_{u^3}(A \upharpoonright\upharpoonright \max I_j) \leqslant l + h < c_i m - h + h = c_i m \leqslant \max I_j/4 < \max I_j/3$, which is a contradiction. So with $d_i = h$, the lemma holds. □

Lemma 6.6. *Each interval receives attention at only finitely many stages.*

Proof. Suppose by induction that no interval below I_j receives attention after stage s, and that A does not change below $\min I_j$ after stage s. That is, $A_{s'} \upharpoonright \min I_j = A_s \upharpoonright \min I_j$ for all $s' \geqslant s$. We show that I_j receives attention at only finitely many stages after stage s. Suppose that φ_i is assigned to I_j, and that I_j is open at all stages after stage s.

If we move in I_j at some stage, then I_j will not receive attention at any later stage. So assume that I_j receives attention and we act in Subcase 2a at stage $s_1 > s$. Then for $e = e_{s_1}$, we compress a set of strings at stage s_1, and I_j is happy at the end of stage s_1. By Lemma 6.2, I_j will be happy at all later stages, unless we either move in some interval restrained by L_e at some stage $t > s_1$, or see that some L_d with $d < e$ restrains I_j at some stage $t > s_1$.

Suppose I_j is restrained by some L_d with $d < e$ at stage $s_2 > s_1$. Then I_j will be unhappy at stage s_2. If at some later stage s_3 we see sufficient convergence of φ_i, we will want to act in I_j. By assumption, no interval below I_j can receive attention at stage s_3, and so I_j will receive attention at stage s_3. If we act in Subcase 2a at stage s_3, then I_j is happy at the end of stage s_3. This cycle can repeat at most e many times, as there are only e many requirements L_d with $d < e$. If we act in Subcase 2b at stage s_3, then we move in I_j at stage s_3, after which I_j can no longer receive attention.

So suppose we move in some interval restrained by L_e at some stage $t > s_1$. Then by Lemma 6.2, I_j will become unhappy at stage t. At the end of stage t, the construction will move in I_j, after which I_j can no longer receive attention. □

Lemma 6.7. *A is Bennett deep.*

Proof. By Lemma 6.4, the set D is a set of requests of weight less than 1. So by the KC theorem, there is a constant f such that if $(l, \nu) \in D$, then $K(\nu) \leqslant l + f$. Suppose that φ_i is an order function. We show that for all $c > \max\{i - f, 0\}$, there is some n such that $K^{\varphi_i}(A \upharpoonright\upharpoonright m) > K(A \upharpoonright\upharpoonright m) + c$ for all $m \geqslant n$, and therefore that R_i is satisfied.

Fix $c > \max\{i - f, 0\}$. By Lemma 6.3, we injure each L-requirement at most finitely many times. Let a be least such that no interval above I_a is every restrained by any L_e with $e \leqslant c + f$.

Let c_i and d_i be the constants as in Lemma 6.5, and let b_0 be sufficiently large so that the Lemma 6.5 holds for all $j \geqslant b_0$. As A is a c.e. set, there exists some d such that for all $m \in \omega$, $K(A \upharpoonright\upharpoonright m) \leqslant 4\log(m+1) + d$. Let $b \geqslant b_0$ be least such that φ_i is assigned to I_b, and for all $m \geqslant \min I_b$, $(c_i m - d_i) - (4\log(m+1) + d) > c$. We will show that $K^{\varphi_i}(A \upharpoonright\upharpoonright m) > K(A \upharpoonright\upharpoonright m) + c$ for all $m > \max I_b$.

We show that each interval I_j dedicated to R_i with $j \geqslant b$ is happy at all but finitely many stages. Suppose that I_j is dedicated to R_i with $j \geqslant b$, and that for each interval I_p below I_j that is dedicated to R_i with $p \geqslant b$ is happy at every stage after stage s. It is clear that I_j is happy at all but finitely many stages if we move in I_j at some stage. So suppose we never move in I_j. Let s be such that no interval below I_j receives attention after stage s, and $A_t \upharpoonright \min I_j = A_s \upharpoonright \min I_j$ for all $t \geqslant s$. Such a stage exists by Lemma 6.6 and because A is c.e. As φ_i is an order function, if I_j is unhappy at some stage $s_1 > s$, then we will eventually want to act in I_j at some stage $s_2 \geqslant s_1$. As $s_2 > s$, I_j will receive attention at stage s_2. Then I_j will be happy at the end of stage s_2. If I_j becomes unhappy at some later stage, then it will later receive attention, and again become happy. By the previous lemma, I_j will eventually stop receiving attention, after which it will be happy at all later stages.

Let $m > \max I_b$. Let I_j be the interval dedicated to R_i such that $\max I_j < m \leqslant \max I_{j^+}$. We know that I_j is happy at all but finitely many stages. Suppose I_j is happy at all stages after stage t. First suppose I_j is happy because we move in I_j at some stage. Then because $j > b \geqslant b_0$ and by the choice of b_0, we have $K^{\varphi_i}(A \upharpoonright\upharpoonright m) \geqslant c_i m - d_i$, and so by the choice of b, $K^{\varphi_i}(A \upharpoonright\upharpoonright m) > K(A \upharpoonright\upharpoonright m) + c$. Now suppose I_j is happy because at all stages $u \geqslant t$, for L_e the strongest priority L-requirement which restrains I_j at stage u, I_j is happily restrained by L_e. We know by Lemma 6.6 that I_j receives attention at most finitely many times, so suppose I_j does not receive attention after stage v. Let L_d be the strongest priority L-requirement which restrains I_j at stage v. Let $\sigma \prec A_{v-1}$ be least such that $\Phi_d^\sigma(d)[v]\downarrow$, and k be least such that φ_i is assigned to I_k and $\min I_k > r_{d,v}$. Then at all stages $w \geqslant v$, we have compressed the strings $\nu \prec A_w$ such that $\max I_j < |\nu| \leqslant \max I_k$ for the sake of R_i due to L_d. In particular, we have compressed $A \upharpoonright\upharpoonright m$ for the sake of R_i due to L_d. Then by choice of a and because $j > b \geqslant a$, we have that $d > c + f$. Therefore, $K^{\varphi_i}(A \upharpoonright\upharpoonright m) > K(A \upharpoonright\upharpoonright m) + c$. □

□

References

[1] Luis Antunes, Lance Fortnow, Dieter van Melkelbeck, Variyam Vinochandram, Computational Depth: Concept and Applications, *Theoretical Computer Science*, Vol. 354 (2006), 391-404.

[2] Luis Antunes, Armato Matos, Andre Souto, and Paul Vitanyi, Depth as randomness deficiency, *Theory of Computing Systems*, Vol. 45 (2009), 724-739.

[3] Charles Bennett, Logical depth and physical complexity, *The Universal Turing Machine, a Half Century Survey*, (1988) 227-257.

[4] David Doty and Phillipe Moser, Feasible depth, In *Computability in Europe, 2007*, LNCS 4497, Springer-Verlag (2007), 228-237.

[5] Rodney Downey and Debis Hirscheldt, *Algorithmic Randomness and Complexity*, Springer-Verlag, 2010.

[6] Rupert Hölzl, Thorsten Kräling, and Wolfgang Merkle. *Time-bounded Kolmogorov complexity and Solovay functions*. Theory Comput. Syst., 52:80–94, 2013.

[7] Antonín Kučera and Sebastiaan Terwijn. *Lowness for the class of random sets*. Journal of Symbolic Logic, 64(4):1396–1402, 1999.

[8] James Lathrop and Jack lutz, recursive computational depth, *Information and Computation*, Vol. 153 (1999), 139-172.

[9] Ming Li and Paul Vitanyi, *Introduction to Kolmogorov Complexoity and Its Applications,* Springer-Verlag, 2008.

[10] Phillippe Moser, On the polynomial septh of various sets of random strngs, *Theoretical Computer Science*, Vol. 477 (2013), 96-108.

[11] Philippe Moser and Frank Stephan. *Depth, Highness and DNR degrees*. Fundamentals of Computation Theory, Twentieth International Symposium, FCT 2015, Gdansk, Poland, August 17–19, 2015, Proceedings. Springer LNCS 9210, 81–94, 2015.

[12] Andre Nies, *Lowness properties and randomness*, Adv. Math. 197 (2005), 274–305.

[13] Andre Nies, *Computability and Randomness*, Oxford University Press, 2009.

[14] Robert Soare, *Recursively Enumerable Sets and Degrees*, Springer-Verlag, 1987.

School of Mathematics and Statistics, Victoria University of Wellington, PO Box 600, Wellington, New Zealand

Mathematics Department, Nanyang University of Technology, Singapore

School of Mathematics, Statistics and Computer Science, Victoria University of Wellington, PO Box 600, Wellington, New Zealand