Theoretical Computer Science ••• (••••) •••–•••

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

On the rate of decrease in logical depth

L.F. Antunes[a,1], A. Souto[b,c,2], P.M.B. Vitányi[d,e,*,3]

[a] *University of Porto & CRACS/INESC-TEC, Rua do Campo Alegre, 1021, 4169-007 Porto, Portugal*
[b] *LaSIGE, Faculdade de Ciências, Universidade de Lisboa, Portugal Campo Grande, 1749-016 Lisboa, Portugal*
[c] *Departamento de Informática, Faculdade de Ciências, Universidade de Lisboa, Campo Grande, 1749-016 Lisboa, Portugal*
[d] *CWI, Science Park 123, 1098 XG Amsterdam, The Netherlands*
[e] *University of Amsterdam, The Netherlands*

ARTICLE INFO

Article history:
Received 15 March 2017
Received in revised form 26 July 2017
Accepted 19 August 2017
Available online xxxx
Communicated by L.M. Kirousis

Keywords:
Logical depth
Kolmogorov complexity
Compression

ABSTRACT

The logical depth with significance b of a string x is the shortest running time of a program for x that can be compressed by at most b bits. Another definition is based on algorithmic probability. We give a simple new proof for the known relation between the two definitions. We also prove the following: Given a string we can consider the maximal decrease in logical depth when the significance parameter increases by 1. There exists a sequence of strings of lengths $n = 1, 2, \dots,$ such that this maximal decrease as a function of n rises faster than any computable function but not as fast as the Busy Beaver function. This holds also for the computation times of the shortest programs of these strings.

1. Introduction

The logical depth is related to complexity with bounded resources and measures the tradeoff between program sizes and running times. Computing a string x from one of its shortest programs may take a very long time, but computing the same string from a simple "`print(x)`" program of length about $|x|$ bits takes very little time.

A program p for x, i.e. such that $U(p) = x$ where U is a universal Turing machine, of larger length than a given program q for x may require less computation time than q does. This need not always be the case, as a longer program might simply perform some pointless redundant steps.

In general we associate longer computation times with shorter programs for x. As a consequence, one may raise the question of how much time can be saved by computing a given string from a longer program.

* Corresponding author.
E-mail addresses: antunes.lfa@gmail.com (L.F. Antunes), andrenunosouto@gmail.com (A. Souto), Paul.Vitanyi@cwi.nl (P.M.B. Vitányi).

1 This work was supported by "Project *NanoSTIMA: Macro-to-Nano Human Sensing: Towards Integrated Multimodal Health Monitoring and Analytics*/NORTE-01-0145-FEDER-000016" which is financed by the North Portugal Regional Operational Programme (NORTE 2020), under the PORTUGAL 2020 Partnership Agreement, and through the European Regional Development Fund (ERDF).

2 This work was supported by LaSIGE Research Unit, ref. UID/CEC/00408/2013 and the FCT project Confident PTDC/EEI-CTP/4503/2014, tSQIG – Security and Quantum Information Group, the Instituto de Telecomunicações (IT) Research Unit, ref. UID/EEA/50008/2013, and the FCT Post-doc scholarship SFRH/BPD/76231/2011 during which the major part of the work was done.

3 CWI is the National Research Institute for Mathematics and Computer Science in the Netherlands.

http://dx.doi.org/10.1016/j.tcs.2017.08.012
0304-3975/© 2017 Published by Elsevier B.V.

1.1. Previous work

The above question was first considered by C. Bennett in [1]. The minimum time to compute a string by a b-incompressible program was called the *logical depth at significance* b of the string considered. Bennett also provided variations of this definition and studied their basic properties and relations. A more formal treatment, as well as an intuitive approach, was given in [4], Section 7.7.

1.2. Results

Section 2 introduces notations, definitions and results needed in the remainder. Section 3 presents two versions of logical depth and gives a simple new proof of quantitative relations between them. Section 4 shows that slight variations on the significance level may cause drastic variations of logical depth. Section 5 presents conclusions.

2. Preliminaries

We use *string* for a finite binary string. Strings are denoted by lower case letters, such as x, y and z. The *length* of a string x (the number of occurrences of bits in it) is denoted by $|x|$, and the *empty* string by ϵ. Thus, $|\epsilon| = 0$. The nth string in the lexicographic length-increasing order is viewed also as the natural number n and vice versa.

Computability, resource-bounded computation time, self-delimiting strings, big-O notation, and (prefix) Kolmogorov complexity are well-known and the properties, notations, are treated in [4].[4] Originally Kolmogorov complexity was introduced in 1965 in [2] and the prefix version in 1974 in [3]. The length of a self-delimiting version of a string of length n will be $n + 2\log n + 1$ where log denotes the logarithm base 2. That is, the self-delimiting version of x is $x' = 1^{||x||}0|x|x$ where $||x||$ denotes the length of $|x|$. Restricting the computation time resource is indicated by a superscript giving the allowed number of steps, usually denoted by d.

We choose a *reference optimal universal prefix Turing machine* and call it U. Let x, y be strings. The *prefix Kolmogorov complexity* $K(x|y)$ of x with auxiliary y is defined by

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

If x is a string of length n then $K(x|y) \le n + O(\log n)$. The notation $U^d(p, y) = x$ means that $U(p, y) = x$ within d steps. The *d-time-bounded prefix Kolmogorov complexity* $K^d(x|y)$ is defined by[5]

$$K^d(x|y) = \min_p\{|p| : U^d(p, y) = x\}.$$

If the auxiliary string y is the empty string ϵ, then we usually drop it. Similarly, we write $U(p)$ for $U(p, \epsilon)$. The string x^* is a *shortest program* for x if $U(x^*) = x$ and $K(x) = |x^*|$. A string x is *c-incompressible* if $|x^*| \ge |x| - c$ and it is *c-compressible* if $|x^*| \le |x| - c$.

Define $Q(x) = \sum_{p:U(p)=x} 2^{-|p|}$. In [3] L.A. Levin proved in the Coding Theorem that (see [4] Theorem 4.3.3 for details):

$$-\log Q(x) = K(x) + O(1)$$

3. Different versions of logical depth

The logical depth as defined in [1] for a string (the finite case) comes in two versions: one based on the compressibility of programs of prefix Turing machines and the other using the ratio between algorithmic probabilities with and without time limits.

The algorithmic probability version is based on the so-called *a priori* probability [4] and its time-bounded version:

$$Q(x) = \sum_{U(p)=x} 2^{-|p|}, \quad Q^d(x) = \sum_{U^d(p)=x} 2^{-|p|}.$$

[4] The reader may be less familiar with the *prefix Turing machine*. It is a Turing machine with a one-way read-only program tape, an auxiliary tape, one or more work tapes and an output tape. All tapes are linear and divided in cells capable of containing one out of a finite set of symbols. Initially the program tape is inscribed with an infinite sequence of 0's and 1's and the head is scanning the leftmost cell (the program tape is semi-infinite). When the computation terminates the sequence of bits scanned is the *program*. For every fixed contents of the auxiliary tape the set of programs for such a machine is a prefix code (no program is a proper prefix of another program). Another consequence is that if the computation of a prefix Turing machine takes d steps then for its program p holds that $d \ge |p|$. The prefix Kolmogorov complexity is based on the prefix Turing machine similar to the (plain) Kolmogorov complexity based on the (plain) Turing machine.

[5] Since we deal with running times of computations the following can happen. Two different reference optimal universal prefix Turing machines may have different computation times for the same combination of input, auxiliary string, and output. It can also be the case that they have different sets of programs. Let U and U' be two optimal universal prefix Turing machines in the standard enumeration of prefix Turing machines. For every auxiliary y and every program q there is a program p with $|q| = |p| + O(1)$ such that $U^d(q, y) = U'^{d'}(p, y)$ for integers d, d' and $d' = g(d)$ with g a computable function. Therefore, although U and U' have the same length of shortest programs for a string x (with $O(1)$ precision, that is, up to a constant additive term), the time-limited prefix Kolmogorov complexity of a string x may differ by a non-constant additive term.

Definition 1. Let x be a string and b a nonnegative integer. The logical depth, version 1, of x at significance level $\varepsilon = 2^{-b}$ is

$$\mathrm{depth}^{(1)}_{\varepsilon}(x) = \min\left\{ d : \frac{Q^d(x)}{Q(x)} \geq \varepsilon \right\}.$$

The definition of logical depth based on the prefix Kolmogorov complexity version is based on the minimal time the reference optimal prefix Turing machine needs to compute x from a program which is b-incompressible.

Definition 2. Let x be a string and b a nonnegative integer. The logical depth, version 2, of x at significance level b, is

$$\mathrm{depth}^{(2)}_{b}(x) = \min\left\{ d : p \in \{0,1\}^* \wedge U^d(p) = x \wedge |p| \leq K(p) + b \right\},$$

the least number of steps to compute x by a b-incompressible program.

Remark 1. The relation of almost equality between those definitions is known [4, Theorem 7.7.1] based on the more informal [1, Lemma 3]. As far as we know that is the only existing proof. We give a simple new proof based on algorithmic probability. ◇

Theorem 1. *Let x be a string and b, d positive integers. Then*

(i) $\mathrm{depth}^{(1)}_{2^{-b}}(x) = d$ implies $\mathrm{depth}^{(2)}_{b-O(1)}(x) = d$, and
(ii) $\mathrm{depth}^{(2)}_{b}(x) = d$ implies $\mathrm{depth}^{(1)}_{2^{-b'}}(x) \leq d$ for some $b' \leq b + K(b) + O(1)$.

Proof. (i): Let $Q^d(x) \geq 2^{-b}Q(x)$ with d least. Let c be the greatest integer such that all programs computing x within d steps are c-compressible. Therefore every p such that $U^d(p) = x$ satisfies $U(r) = p$ for some r. To compute string x by U using program r requires that r is concatenated with an extra program q of constant length $c' \geq 0$ to restart U after it has computed p. Then $U^d(U(rq)) = x$ and $|r| \leq |p| - c - c'$. For every p there are possibly more than one such r. Let P be the set of these p's and R be the set of the r's. Hence $\sum_{r\in R} 2^{-|r|} \geq \sum_{p \in P} 2^{-(|p|-c-c')}$. That is, the reference optimal universal Turing machine computes x with probability at least $2^{c+c'}Q^d(x) \geq 2^{c+c'-b}Q(x)$ from the programs in R. Then $Q(x) \geq 2^{c+c'-b}Q(x) + 2^{-b}Q(x)$ (the left-hand side comprises all halting programs for x no matter what is the running time and the right-hand side comprises a lower bound on the contribution of the programs in R plus the contribution of the programs in P). This means that $2^{c+c'-b} < 1$ and therefore $c + c' - b < 0$, that is $c + c' < b$. Among the programs in P there is a program which is c-incompressible (otherwise c is not greatest as assumed above) and since $c + c' < b$ it is $(b - c') = (b - O(1))$-incompressible.

(ii): Assume that d is the least number of steps to compute x by a b-incompressible program. By way of contradiction let $Q^d(x) < 2^{-B}Q(x)$ with $B = b + K(b) + e > 0$ for a large enough constant e. Let an auxiliary distribution R be defined as follows. Computably enumerate the set P of all programs p such that $U^d(p) = x$. Let $R(x) = \sum_{p\in P} 2^{B-|p|}$. Then[6] $R(x) = 2^B Q^d(x) < Q(x)$. The function R given B is lower semicomputable. Since Q is a semimeasure and $R(x) < Q(x)$ for all x the function R is a semimeasure. A program $p \in P$ adds $2^{-|p|+B}$ probability to $R(x)$ and p is therefore $(B - O(1))$-compressible given B, and hence $(B - K(B) - O(1))$-compressible. It remains to bound the constant term. For sufficiently large e,

$$\begin{aligned} B - K(B) - O(1) &= b + K(b) + e - K(b + K(b) + e) - O(1) \\ &\geq b + K(b) + e - K(b + K(b)) - K(e) - O(1) \\ &> b. \end{aligned}$$

The first inequality follows from $K(u + v) \leq K(u) + K(v) + O(1)$, and the second inequality holds since $K(b + K(b)) \leq K(b, K(b)) + O(1) \leq K(b) + O(1)$ and $K(e) < e/2$ for e large enough. Hence all programs that compute x in d steps are $(b + 1)$-compressible contradicting the assumption. Therefore $Q^d(x) \geq 2^{-B}Q(x)$. □

Remark 2. With b, d as in the theorem a slight modification in the proof yields

$$\frac{1}{2^{b+K(b)+O(1)}} \leq \frac{Q^d_U(x)}{Q_U(x)} \leq \frac{1}{2^{b-O(1)}},$$

where (i) corresponds to the right inequality and (ii) to the left inequality. ◇

[6] We assume that $R(x) > 0$. If $R(x) = 0$ then there is no program p such that $U^d(p) = x$ and (ii) is vacuously correct.

Remark 3. Notice that it is possible to use $K(d)$ instead of $K(b)$ in the above proof by changing the construction of the semiprobability as follows: knowing d generate all programs p computing x within d steps and let the semiprobabilities be proportional to $2^{-|p|}$ and the sum be less than $Q(x)$. In this way, $K(b)$ in Theorem 1 can be improved to $\min\{K(b), K(d)\} + O(1)$. ◇

4. The graph of logical depth

Slight changes of the significance level b can drastically change the value of the logical depth.

Lemma 1. *Let ϕ be defined by*

$$\phi(n) = \max_{|x|=n} \min_{d} \{d : U^d(x^*) = x\}.$$

Then ϕ is not computable and grows faster than any computable function.

Proof. If a function ϕ as in the lemma were computable, then for an x of length n we could run U for $\phi(n)$ steps on any program of length $n + O(\log n)$. Among those programs that halt within $\phi(n)$ steps, we could select the ones which output x. Subsequently, we could select from that set a program of minimum length from which x can be computed. Such a program has length $K(x)$. This would imply that K would be computable. But the function K is incomputable: contradiction. Therefore ϕ cannot be computable. Since this holds for every function majoring ϕ, the function ϕ must grow faster than any computable function. □

Definition 3. The *busy beaver function* $BB : \mathbb{N} \to \mathbb{N}$ is defined by

$$BB(n) = \max\{d : |p| \le n \wedge U^d(p) < \infty\}.$$

This is a variant in terms of number of steps from the original Busy Beaver function in [5]. The following result was mentioned informally in [1].

Lemma 2. *The running time of a program p of length n that halts is at most $BB(n)$. The running time of a shortest program (shortest in prefix Kolmogorov complexity) for a string x of length n is at most $BB(n + O(\log n))$.*

Proof. The first statement of the lemma follows from Definition 3. The second statement follows from the observation that $K(x) \le n + O(\log n)$ for every x of length n. □

Theorem 2. *The function*

$$f(n) = \max_{|x|=n,\ 0 \le b \le n} \{x : depth_b^{(2)}(x) - depth_{b+1}^{(2)}(x)\}$$

grows faster than any computable function but not as fast as the Busy Beaver function.

Proof. Let ϕ be the incomputable function of Lemma 1. For every $n \ge 1$ select x_n of length n such that $U^{\phi(n)}(x_n^*) = x_n$. Since a shortest program can be compressed by at most $O(1)$ bits we have $\mathrm{depth}_{O(1)}^{(2)}(x) = \phi(n)$. The optimal reference prefix Turing machine U can compute x from its self-delimiting version x' of length $n + O(\log n)$ using an $O(1)$-bit program in time $g(n)$ with g a computable function.

For every n we can consider the finite sequence $L_0^n, L_1^n, \dots, L_{n-K(x)+O(\log n)}^n$ with $L_m^n = \mathrm{depth}_m^{(2)}(x_n) - \mathrm{depth}_{m+1}^{(2)}(x_n)$ $(0 \le m \le n - K(n) + O(\log n))$, and $\phi(n) = \sum_{i=0}^{n-K(x)+O(\log n)} L_i^n$. Then, the average $(\phi(n) - g(n))/(n - K(n) + O(\log n))$ is incomputable (as it grows faster than any computable function), and therefore the function f defined by

$$f(n) = \max_m \{L_n^m : 0 \le m \le n - K(x) + O(\log n)\}$$

also grows faster than any computable function, since $f(n) \ge (\phi(n) - g(n))/(n - K(n) + O(\log n))$. □

5. Conclusion

In [1] one version of logical depth is based on compression and one is based on algorithmic probability. These are closely related [1,4]. We gave a new proof of this relation based on incompressibility and algorithmic probability. For a string x the logical depth with significance b is the minimum number of steps d in a computation of x from a program p that is less than b bits longer than $K(p)$. This b is called the significance. There is a sequence of strings such that for each string changing the significance parameter by only 1 can change the depth by a large amount. As a function of the position of the string in the sequence this amount grows faster than any computable function but not as fast as the Busy Beaver function. The computation times of shortest programs can similarly rise faster than any computable function but not as fast as the Busy Beaver function.

Acknowledgements

We thank an anonymous person for comments and suggesting the new proof of Theorem 1 and the referees for comments.

References

[1] C. Bennett, Logical depth and physical complexity, in: R. Herken (Ed.), The Universal Turing Machine: A Half-Century Survey, Oxford University Press, 1988, pp. 227–257.
[2] A.N. Kolmogorov, Three approaches to the quantitative definition of information, Probl. Inf. Transm. 1 (1) (1965) 1–7.
[3] L.A. Levin, Laws of information conservation (non-growth) and aspects of the foundation of probability theory, Probl. Inf. Transm. 10 (1974) 206–210.
[4] M. Li, P.M.B. Vitányi, An Introduction to Kolmogorov Complexity and Its Applications, Springer, New York, 2008.
[5] T. Rado, On non-computable functions, Bell Syst. Tech. J. 41 (3) (1962) 877–884.