Logical depth for reversible Turing machines with an application to the rate of decrease in logical depth for general Turing machines

Paul M.B. Vitányi[*]

August 29, 2019

Abstract

The logical depth of a *reversible* Turing machine equals the shortest running time of a shortest program for it. This is applied to show that the result in [1] is valid notwithstanding the error noted in Corrigendum [7].

keywords: Logical depth, Kolmogorov complexity, compression

1   Introduction

A book on number theory is difficult, or 'deep.' The book lists a number of difficult theorems of number theory. However, it has very low Kolmogorov complexity, since all theorems are derivable from the initial few definitions. Our estimate of the difficulty, or 'depth,' of the book is based on the fact that it takes a long time to reproduce the book from part of the information in it. The existence of a deep book is itself evidence of some long evolution preceding it.

The logical depth of a (finite) string 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.

Logical depth as defined in [4] for a string 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. Since both are approximately the same ([5, Theorem 7.7.1] based on [4, Lemma 3]) it is no restriction to use the compressibility version.

The used notions of computability, resource-bounded computation time, self-delimiting strings, big-O notation, and Kolmogorov complexity are well-known and the properties, notations, are treated in [5].

---

[*]CWI and University of Amsterdam. Address: CWI, Science Park 123, 1098XG Amsterdam, The Netherlands. Email: Paul.Vitanyi@cwi.nl

2　Preliminaries

All Turing machines in this paper are prefix Turing machines. A *prefix Turing machine* 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 one-way infinite and divided into cells capable of containing one symbol out of a finite set. Initially the program tape is inscribed with an infinite sequence of 0's and 1's and the head is scanning the leftmost cell. When the computation terminates the sequence of bits scanned on the input tape is the *program*. For every fixed finite 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). Let $T_0, T_1, \ldots$ be the standard enumeration of prefix Turing machines. A *universal prefix Turing machine* simulates every prefix Turing machine given its index number. We also require it to be *optimal* which means that the simulation program is as short as possible. We choose a *reference optimal universal prefix Turing machine* and call it U.

The prefix Kolmogorov complexity is based on the prefix Turing machine similar to the (plain) Kolmogorov complexity based on the (plain) Turing machine. Let x, y be finite binary 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 binary string of length n then $K(x|y) \leq n + O(\log n)$. Restricting the computation time resource is indicated by a superscript giving the allowed number of steps, usually denoted by d. The notation $U^d(p, y) = x$ means that $U(p, y) = x$ within d steps. 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 *b-incompressible* if $|x^*| \geq |x| - b$.

3　Reversible Turing Machines

A Turing machine behaves according to a finite list of rules. These rules determine, from the current state of the finite control and the symbol contained in the cell under scan, the operation to be performed next and the state to enter at the end of the next operation execution.

The device is (*forward*) *deterministic*. Not every possible combination of the first two elements has to be in the set; in this way we permit the device to perform *no* operation. In this case we say that the device *halts*. Hence, we can define a Turing machine by a *transition function*.

DEFINITION 1. A *reversible* Turing machine [3, 2] is a Turing machine that is forward deterministic (any Turing mchine as defined is) but also *backward deterministic*, that is, the transition function has a single-valued inverse. The

details of the formal definition are intricate [3, 2] and need not concern us here. This definition extends in the obvious manner to multitape Turing machines.

In [3] for every 1-tape ordinary Turing machine T a 3-tape reversible Turing machine T_{rev} is constructed that emulates T in linear time such that with input p the output is $T_{\mathrm{rev}}(p) = (p, T(p))$. The reversible Turing machine that emulates U is called U_{rev}.

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

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

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

THEOREM 1. *The logical depth of a string x at significance level $b \in \mathcal{N}$ for reversible Turing machines is equal to*

$$depth_b(x) = \min\left\{d : p \in \{0, 1\}^* \wedge U_{\mathrm{rev}}^d(p) = (p, x) \wedge |p| \leq K(x) + b\right\},$$

the least number of steps to compute x by U_{rev} from a program of length at most $K(x) + b$.

Proof. Since a reversible Turing machine is backwards deterministic, and an incompressible program cannot be computed from a shorter program, the length of an incompressible program for x can only be the length of a shortest program for x. The logical depth at significance b is then the least number of steps to compute x by U_{rev} from a program p of length $K(x) + b$. $\square$

4   The Rate of Decrease of Logical Depth

In [1, Section 4] it is assumed that, for all $x \in \{0, 1\}^*$, the string x^* is the only incompressible string such that $U(x^*) = x$. That is, logical depth according to Theorem 1 is used. However, this assumption is wrong for general Turing machines in that for many x there may be an incompressible string p with $|x| \geq |p| > |x^*|$ such that $U(p) = x$. The computation of $U(p) = x$ may be faster than that of $U(x^*) = x$. For example, the function from $x \in \{0, 1\}^*$ to the least number of steps in a computation $U(p) = x$ for an incompressible string p may be computable. The argument in the paper is, however, correct for the set of reversible Turing machines. These Turing machines are a subset of the set of all Turing machines [3, 2] and emulate them in linear time. This implies the correctness of [1, Theorem 2] as we shall show.

LEMMA 1. *Let ψ be defined by*

$$\psi(n) = \max_{|x|=n} \min_d \{d : U_{\mathrm{rev}}^d(x^*) = (x^*, x)\}.$$

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

3

Proof. If a function ψ as in the lemma were computable, then for an x of length n we could run U_{rev} emulating U [3] forward for $\psi(n)$ steps on all programs of length $n+O(\log n)$. Among those programs that halt within $\psi(n)$ steps, we could select the programs p which output (p,x). Subsequently, we could select from that set a program p of minimum length, say x^*. Such a program x^* has length $K(x)$ since U_{rev} is emulating U. This would imply that K would be computable. But the function K is incomputable [6, 5]: contradiction. Therefore ψ cannot be computable. Since this holds for every function majoring ψ, the function ψ must grow faster than any computable function. $\qquad\square$

COROLLARY 1. The set of reversible Turing machines is a subset of the set of all Turing machines. The emulation of $U(p)$ by $U_{\mathrm{rev}}(p)$ is linear time for all binary inputs p by [3]. Therefore, replacing in the lemma U_{rev} by U changes $\psi(n)$ to $\phi(n) = \Omega(\psi(n))$. Hence the lemma holds with ψ replaced by ϕ and U_{rev} by U. This gives us [1, Lemma 1] and therefore [1, Theorem 2] (the Busy Beaver upper bound is proved as it is in [1]):

THEOREM 2. *The function*

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

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

References

[1] L.F. Antunes, A. Souto, and P.M.B. Vitányi, On the Rate of Decrease in Logical Depth, *Theor. Comput. Sci.*, 702(2017), 60–64.

[2] H.B. Axelsen and R. Glück, A simple and efficient universal reversible Turing machine, *Proc. 5th Int. Conf. Language and Automata Theory and Applications*, Lecture notes in computer science, Vol. 6638, Springer, 2011, 117–128.

[3] C.H. Bennett, Logical reversibility of computation, *IBM J. Research and Development*, 17:6(1973), 525–532.

[4] C.H. Bennett, Logical depth and physical complexity, pages 227–257 in *The Universal Turing Machine A Half-Century Survey*, R. Herken Ed., Oxford University Press, 1988.

[5] M. Li and P.M.B. Vitányi, *An Introduction to Kolmogorov Complexity and Its Applications*, Springer, New York, 2008.

[6] A.N. Kolmogorov, Three approaches to the quantitative definition of information. *Problems Inform. Transmission*, 1:1(1965), 1–7.

[7] P.M.B. Vitányi, Corrigendum to "On the rate of decrease in logical depth" by L.F. Antunes, A. Souto, and P.M.B. Vitányi [Theoret. Comput. Sci. 702 (2017) 60–64], *Theoret. Comput. Sci.*, https://doi.org/10.1016/j.tcs.2018.07.009 (In Press, Corrected Proof).