Robustness of Logical Depth

L. Antunes and A. Souto and A. Teixeira

Instituto de Telecomunicações [⋆]
Faculdade de Ciências Universidade do Porto

Abstract. Usually one can quantify the subjective notion of useful information in two different perspectives: static resources – measuring the amount of planing required to construct the object; or dynamic resources – measuring the computational effort required to produce the object. We study the robustness of logical depth measuring dynamic resources, proving that small variations in the significance level can cause large changes in the logical depth.

Classification: Kolmogorov Complexity, Information Measures.

1 Introduction

Philosophy and meaning of information has a long history, however recently interest in this area has increased and researchers have tackled this from different approaches and perspectives, namely its meaning, quantification and measures of information and complexity. In this paper we are interested in measures of meaningful or useful information. In the past there have been several proposals to address this question: sophistication [Kop87,Kop95], logical depth [Ben88], effective complexity [GML96], meaningful information [Vit06], self-dissimilarity [WM07], computational depth [AFvMV06], facticity [Adr12]. Pieter Adiraans [Adr12] divided the several approaches to defined a string as interesting in: i) some amount of computation resources are required to construct the object (Sophistication, Computational Depth). ii) exists a trade-off between the model and the data code under two part code optimization (meaningful information, effective complexity, facticity) and finally iii) it has internal phase transitions (self-dissimilarity).
Solomonoff [Sol64], Kolmogorov [Kol65] and Chaitin [Cha66] independently defined a rigorous measure of the information contained in an individual object x, as the length of the shortest program that produces the object x. This measure is usually called Kolmogorov complexity of x and is denoted by $K(x)$. A randomly generated string, with high probability, has high Kolmogorov complexity, so contains near maximum information. However by being randomly generated,

[⋆] This work was supported by FCT projects PEst-OE/EEI/LA0008/2011 and PTDC/EIA-CCO/099951/2008. The authors are also supported by the grants SFRH/BPD/76231/2011 and SFRH/BD/33234/2007 of FCT.

makes it unlikely to have useful information as we can obtain a similar one by flipping fair coins.
Usually one can quantify the subjective notion of useful information, as in item i) previously defined, in two different perspectives: static resources – measuring the amount of planing required to construct the object; or dynamic resources – measuring the computational effort required to produce the object.
Regarding *dynamic resources*, the Kolmogorov complexity of a string x does not take into account the time effort necessary to produce the string from a description of length $K(x)$. Bennett [Ben88] called this effort *logical depth*. Intuitively, a computationally deep string takes a lot of computational effort to be recovered from its shortest description while shallow strings are trivially constructible from their $K(x)$, i.e., the shortest program for x does not require lots of computational power to produce it. After some attempts, Bennett [Ben88] formally defined the s-significant logical depth of an object x as the time required by a standard universal Turing machine to generate x by a program p that is at most s bits longer than its Kolmogorov complexity. Thus, an object is logically deep if it takes a lot of time to be generated from any short description.
Bennett[Ben88] claimed that the significance level in logical depth is due to stability reasons. In this paper we study its robustness, i.e., if small variations in the significance level can cause large changes in the logical depth. In this sense we show that logical depth is not robust.
The rest of the paper is organized as follows: in the next section, we introduce some notation, definitions and basic results needed for the comprehension of the rest of the paper. In Section 3, we prove that logical depth is not a stable measure.

2 Preliminaries

In this paper we use the binary alphabet $\Sigma = \{0,1\}$. $\Sigma^* = \{0,1\}^*$ is the set of all finite binary strings that are normally represented by x, y and z. Σ^n and $\Sigma^{\leq n}$ are the set of strings of length n and the set of strings of length at most n, respectively. We denote the initial segment of length k of a string x with length $|x|$ by $x_{[1:k]}$ and its i^{th} bit by x_i. The function log will always mean the logarithmic function of base 2. $\lfloor k \rfloor$ represents the largest integer smaller or equal than k. All resource-bounds used in this paper are *time constructible*, *i.e.*, there is a Turing machine whose running time is exactly $t(n)$ on every input of size n, for some time t. Given a program p, we denote its running time by $time(p)$. Given two functions f and g, we say that $f \in O(g)$ if there is a constant $c > 0$, such that $f(n) \leq c \cdot g(n)$, for almost all $n \in \mathbb{N}$.

2.1 Kolmogorov Complexity

We refer the reader to the book of Li and Vitányi [LV08] for a complete study on Kolmogorov complexity.

Definition 1 (Kolmogorov complexity). *Let U be a universal prefix-free Turing machine. The* prefix-free Kolmogorov *complexity of $x \in \Sigma^*$ given $y \in \Sigma^*$ is,*

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

The t-time-bounded prefix-free Kolmogorov complexity *of $x \in \Sigma^*$ given $y \in \Sigma^*$ is,*

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

The default value for the axillary input y for the program p, is the empty string ϵ and to avoid overloaded notation we usually drop this argument in those cases. We choose as a reference universal Turing machine a machine that affects the running time of a program on any other machine by at most a logarithmic factor and the program length by at most a constant number of extra bits.

Definition 2 (c-incompressible). *A string x is* c-incompressible *if and only if $K(x) \geq |x| - c$.*

A simple counting argument can show the existence of c-incompressible strings. In fact,

Theorem 1. *There are at least $2^n \cdot (1 - 2^{-c}) + 1$ strings $x \in \Sigma^n$ that are c-incompressible.*

Bennett [Ben88] said that a string x is logically deep if it takes a lot of time to be generated from any short description. The c-significant logical depth of an object x is the time that a universal Turing machine needs to generate x by a program that is no more than c bits longer than the shortest description of x.

Definition 3 (Logical depth). *Let x be a string, c a significance level. A string's* logical depth *at significance level c, is:*

$$ldepth_c(x) = \min_p \{time(p) : |p| \leq K(x) + c \wedge U(p) = x\}.$$

One can however, scale down the running time for program length by using a Busy Beaver function, similar to the notion of Busy Beaver computational depth introduced in [AF09].

Definition 4 (Busy Beaver function). *The* Busy Beaver function, *BB, is defined by*

$$\begin{aligned} BB : \mathbb{N} &\to \mathbb{N} \\ n &\to \max_{p:|p|\leq n} \{\textit{running time of } U(p) \textit{ when defined}\} \end{aligned}$$

Definition 5 (Busy Beaver logical depth). *The* Busy Beaver logical depth, *with significance level c, of $x \in \Sigma^n$ is defined as:*

$$ldepth_c^{BB}(x) = \min_l \{\exists p : |p| \leq K(x) + c \textit{ and } U(p) = x \textit{ in time } \leq BB(l)\}$$

Notice that this is a rescaling of Definition 3, since $BB^{-1}(\text{ldepth}_c(x)) = \text{ldepth}_c^{BB}(x)$. From this new definition it is clear that $\text{ldepth}_c^{BB}(x) \le K(x)+O(1)$. In fact, one can simulate any computation, keeping track of the amount of computation steps and thus, $K(\text{time}(p)) \le |p| + O(1)$ which implies $\text{ldepth}_c^{BB}(x) \le K(x)+O(1)$. Notice also that the Busy Beaver logical depth is a static measure based on programs length.

3 Instability of logical depth

In this section we prove that even slightly changes of the constant on the significance level c of logical depth determines large variation of this measure.

Theorem 2. *For every sufficiently large n there are constants c, k_1 and k_2 and a string x of length n such that $ldepth_{k_2}(x) \ge 2^n$ and $ldepth_{2c+k_1}(x) \le O(n \cdot \log n)$.*

Proof. Consider the following set:

$$A = \{x \in \Sigma^n : (\exists p)|p| < n + K(n) - c \wedge U(p) = x \text{ in time } \le 2^n\}$$

Considering $B = \Sigma^n - A$, we know that B has at least $2^n(1-2^{-c})$ elements. Let $x \in B$ such that $n + K(n) - c - k_1 \le K(x) \le n + K(n) - c - k_2$ for some constants k_1 and k_2. We show that these strings exist in Lemma 3.1 bellow. Thus,

- $\text{ldepth}_{k_2}(x) \ge 2^n$.
  Assume that $\text{ldepth}_{k_2}(x) < 2^n$, then, by definition of logical depth, there is a program p of size at most $K(x)+k_2 \le n+K(n)-c-k_2+k_2 = n+K(n)-c$ such that $U(p) = x$ in time $< 2^n$, which implies that x would be an element of A, which contradicts the choice of x.
- $\text{ldepth}_{2c+k_1}(x) \le O(n)$.
  Since the significance level is $2c+k_1$, then we can consider programs to define its logical depth of length at least $n+K(n)-c-k_1+2c+k_1 = n+K(n)+c$ (and of course of length at most $n+K(n)-c-k_2+2c+k_1 = n+K(n)+c+k_1-k_2$). So, if c is sufficiently large to allow a prefix free version of the program `print` to be one of the possible programs, then we conclude that $\text{ldepth}_{2c+k_1}(x)$ is at most the running time of `print`(x), which is at most $O(n \cdot \log n)$.

Lemma 3.1. *Let c be a constant and B be the set described in the last proof. There are constants k_1 and k_2 and strings in B such that $n - c - k_1 \le K(x) \le n - c - k_2$.*

Proof. Consider the set $S = \{x \in \Sigma^n : K(x) \ge n + K(n) - c - k_1\}$.
It is easy to see that every element in S is in B. Let p be the program of size $\le n + K(n) - c - a$ where a is a constant to be defined later that has the longest running time. Notice that $K(p) \ge n + K(n) - c - a - l$ for some l. In fact, if $K(p) < n + K(n) - c - a - l$ for all l then we could consider the program that runs p^*, the 1st program in the lexicographic order that produces p to obtain

p and then run it again and that would be a smaller program that would have longer running time. More formally, consider the program $q = RUN(\cdot)$ where RUN describes the universal Turing machine with some data and run it on U. Since RUN is describable by a constant number of bit, say s, then, if the data is p^*, $|q| = |p^*| + s \leq n + K(n) - a - l + s \leq n + K(n) - a$ for sufficiently large l. Furthermore, $time_U(q) \geq time_U(RUN(p^*)) = time_U(p^*) + time_U(p) \geq time_U(p)$ which contradicts the choice of p.
Let t be the running time of p and let x be the first string in the lexicographic order such that $K^t(x) \geq n + K(n)$. Thus,

- $K(x) \leq K(p) + b \leq n + K(n) - c - a + b$ for some constant b since from p we can compute t and then compute x.
- $K(x) \geq n + K(n) - c - a$. In fact, if $K(x) < n + K(n) - c - a$ then considering q the prefix-free program that witnesses the Kolmogorov complexity of x we would have that $|q| < n - c - a$ and then $U(q) = x$. Thus, by definition of p we get $time(q) < time(p)$ and hence $K^t(x) \leq n + K(n) - a$ contradicting the choice of x.

Just take $a > b$ and also $k_1 = a$ and $k_2 = a - b$.

Theorem 3. *For every sufficiently large there are constants c, k_1 and k_2 and a string x of length n such that $ldepth^{BB}_{k_2}(x) \geq n$ and $ldepth^{BB}_{2c+k_1}(x) \leq O(\log n)$.*

Proof. The idea is similar to Theorem 2. We rewrite the proof has the reasoning of the conclusions changes a bit.
Consider the following set:

$$A = \{x \in \Sigma^n : (\exists p)|p| < n + K(n) - c \wedge U(p) = x \text{ in time } \leq BB(n)\}$$

Considering $B = \Sigma^n - A$, we know that B has at least $2^n(1 - 2^{-c})$ elements. Let $x \in B$ such that $n + K(n) - c - k_1 \leq K(x) \leq n + K(n) - c - k_2$ for some constants k_1 and k_2. Thus,

- $\text{ldepth}^{BB}_{k_2}(x) \geq n$.
  Assume that $\text{ldepth}^{BB}_{k_2}(x) < n$, then, by definition of busy beaver logical depth, there is a program p of size at most $K(x)+k_2 \leq n+K(n)-c-k_2+k_2 = n + K(n) - c$ such that $U(p) = x$ in time $< BB(n)$, which implies that x would be an element of A, contradicting the choice of x.
- $\text{ldepth}_{2c+k_1}(x) \leq O(\log n)$.
  Since the significance level is $2c+k_1$, then we can consider programs to define its logical depth of length at least $n+K(n)-c-k_1+2c+k_1 = n+K(n)+c$ (and of course of length at most $n+K(n)-c-k_2+2c+k_1 = n+K(n)+c+k_1-k_2$). So, if c is sufficiently large to allow a prefix free version of the program `print` to be one of the possible programs, then we conclude that $\text{ldepth}^{BB}_{2c+k_1}(x)$ is at most $BB^{-1}(time(\texttt{print}(\text{x}))) = BB^{-1}(n \log n) \leq O(\log n)$ (since the busy beaver function grows faster than any computable function, in particular exponential).

We can adapt the argument presented above to prove that if we allow logarithmic terms on the significance level of logical depth we get a similar result.

Corollary 3.1. *For every n and sufficiently large there are constants c, k_1 and k_2 and a string x of length n such that $ldepth^{BB}_{k_2 \log n}(x) \geq n$ and $ldepth^{BB}_{(2c+k_1)\log n}(x) \leq O(\log n)$.*

Proof. The proof is equal to the previous one with the following adaptations:

$$A = \{x \in \Sigma^n : (\exists p)|p| < n + K(n) - c\log n \wedge U(p) = x \text{ in time } \leq BB(n)\}$$

and with a similar reasoning to Lemma 3.1 we can show the existence of a string in the complement of A satisfying $n + K(n) - c\log n - k_1 \log n \leq K(x) \leq n + K(n) - c\log n - k_2 \log n$.

4 Conclusions

Our major contribution in this paper is the proof that the most commonly used definition of logical depth in the literature is not stable, since small variations in the significance level can cause drastic changes in the value of Logical depth even if we correct it with a Busy Beaver function.

Acknowledgments

We thanks Bruno Bauwens for helpful discussions and comments.
This work was supported by FCT projects PEst-OE/EEI/LA0008/2011 and PTDC/EIA-CCO/099951/2008. The authors are also supported by the grants SFRH/BPD/76231/2011 and SFRH/BD/33234/2007 of FCT.

References

[Adr12] Pieter Adriaans. Facticity as the amount of self-descriptive information in a data set. 2012.

[AF09] L. Antunes and L. Fortnow. Sophistication revisited. *Theory of Computing Systems*, 45(1):150–161, Springer–Verlag, 2009.

[AFvMV06] L. Antunes, L. Fortnow, D. van Melkebeek, and N. Vinodchandran. Computational depth: concept and applications. *Theoretical Computer Science*, 354(3):391–404, Elsevier Science Publishers Ltd., 2006.

[Ben88] C. Bennett. Logical depth and physical complexity. In *A half-century survey on The Universal Turing Machine*, pages 227–257, New York, NY, USA, 1988. Oxford University Press, Inc.

[Cha66] G. Chaitin. On the length of programs for computing finite binary sequences. *Journal of ACM*, 13(4):547–569, ACM Press, 1966.

[GML96] Murray Gell-Mann and Seth Lloyd. Information measures, effective complexity, and total information. *Complexity*, 2(1):44–52, 1996.

[Kol65] A. Kolmogorov. Three approaches to the quantitative definition of information. *Problems of Information Transmission*, 1(1):1–7, Springer–Verlag, 1965.

[Kop87] M. Koppel. Complexity, depth, and sophistication. *Complex Systems*, 1:1087–1091, Complex Systems Publication Inc., 1987.

[Kop95] M. Koppel. Structure. *The universal Turing machine (2nd ed.): a half-century survey*, pages 403–419, Springer–Verlag, 1995.

[LV08] Ming Li and P. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer-Verlag, 2008.

[Sol64] R. Solomonoff. A formal theory of inductive inference, Part I. *Information and Control*, 7(1):1–22, Academic Press Inc., 1964.

[Vit06] Paul M. B. Vitányi. Meaningful information. *IEEE Transactions on Information Theory*, 52(10):4617–4626, 2006.

[WM07] David H Wolpert and William Macready. Using self-dissimilarity to quantify complexity. *Complexity*, 12(3):77–85, 2007.