Kolmogorov Complexity for Possibly Infinite Computations

CACIC 2003 - IX Argentine Congress on Computer Science
(no Workshop in particular)

Verónica Becher   Santiago Figueira [*]

vbecher@dc.uba.ar   sfigueir@dc.uba.ar

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

July 28, 2003

Keywords: Kolmogorov complexity, Program-size complexity, Turing machines, monotone machines, infinite computations, non-effective computations.

Abstract

In this paper we study a variant of the Kolmogorov complexity for non-effective computations, that is, either halting or non-halting computations on Turing machines. This complexity function is defined as the length of the shortest inputs that produce a desired output via a possibly non-halting computation. Clearly this function gives a lower bound of the classical Kolmogorov complexity. In particular, if the machine is allowed to overwrite its output, this complexity coincides with the classical Kolmogorov complexity for halting computations relative to the first jump of the halting problem. However, on machines that cannot erase their output –called *monotone* machines–, we prove that our complexity for non effective computations and the classical Kolmogorov complexity separate as much as we want. We also consider the prefix-free complexity for possibly infinite computations. We study several properties of the graph of these complexity functions and specially their oscillations with respect to the complexities for effective computations.

[*]supported by a grant of Fundación Antorchas, Argentina.

1 Introduction

The Kolmogorov or *program-size* complexity [6] classifies strings with respect to a static measure for the difficulty of computing them: the length of the shortest program that computes the string. A low-complexity string has a short algorithmic description from which one can reconstruct the string and write it down. Conversely, a string has maximal complexity if it has no algorithmic description shorter than its full length. Due to an easy but consequential theorem of invariance, program-size complexity is independent of the Turing machine (or programming language) being considered, up to an additive constant. Thus, program-size complexity counts as an absolute measure of complexity.

The prefix-free version of program-size complexity, independently introduced by Chaitin [2] and Levin [8], also serves as a measure of quantity of information, being formally identical to Shanon's information theory [2].

In this paper we study a variant of the Kolmogorov complexity for non-effective computations, that is, either halting or non-halting computations on Turing machines. This complexity function, notated with K^{∞}, is defined as the length of the shortest inputs that produce a desired output via a possibly non-halting computation.

Clearly this function gives a lower bound of the classical Kolmogorov complexity. In particular, if the machine is allowed to overwrite its output, this complexity coincides with the classical Kolmogorov complexity for halting computations relative to the first jump of the halting problem. However, on machines that cannot erase their output –called *monotone* machines–, we prove that our complexity for non effective computations and the classical Kolmogorov complexity separate as much as we want.

We also consider the prefix-free complexity for possibly infinite computations. We study several properties of the graph and specially the oscillations of these functions with respect to the complexities for effective computations.

2 Definitions

$\mathbb{N}$ is the set of natural numbers, and we work with the binary alphabet $\{0,1\}$. As usual, a string is a finite sequence of elements of $\{0,1\}$, λ is the empty string and $\{0,1\}^*$ is the set of all strings. $\{0,1\}^{\omega}$ is the set of all infinite sequences of $\{0,1\}$, i.e. the Cantor space. $\{0,1\}^{\leq\omega} = \{0,1\}^* \cup \{0,1\}^{\omega}$ is the set of all finite or infinite sequences of $\{0,1\}$. For any $n \in \mathbb{N}$, $\{0,1\}^n$ is the set of all strings of length n.

For $a \in \{0,1\}^*$, $|a|$ denotes the length of a. If $a \in \{0,1\}^*$ and $A \in \{0,1\}^{\omega}$ we denote $a \upharpoonright n$ the prefix of a with length $\min(n, |a|)$ and $A \upharpoonright n$ the length n prefix of the infinite sequence A. For $a, b \in \{0,1\}^*$, we write $a \preceq b$ if a is a prefix of b.

We assume the recursive bijection $str : \mathbb{N} \to \{0,1\}^*$ such that $str(i)$ is the i-th string in the length-lexicographic order over $\{0,1\}^*$. We also assume the recursive bijection $\overline{\cdot} : \{0,1\}^* \to \{0,1\}^*$ which for every string $s = b_1 b_2 \ldots b_{n-1} b_n$, $\overline{s} = 0b_1 0b_2 \ldots 0b_{n-1} 1b_n$. This function will be useful to code inputs to Turing machines which require more than one argument.

If f is any partial function then, as usual, we write $f(p)\downarrow$ when it is defined, and $f(p)\uparrow$ otherwise.

2.1 Possibly infinite computations on monotone machines

We work with Turing machines with a one-way read-only input tape, some work tapes, and an output tape. The input tape contains a first dummy cell (representing the empty input) and then a one-way infinite sequence of 0's and 1's representing the input, followed with a special end-marker indicating the end of the input. Notice that the end-marker allows the machine to know exactly where the input ends and the machine is not able to read beyond the end of the end-marker.

We shall refer to two architectures of Turing machines, regarding the input and output tapes. A *monotone* Turing machine has a one-way write-only output tape. A *prefix* machine is a Turing machine but the input tape contains no blank end-marker in the rightmost cell. Since there are no external delimitation of the input tape, the machine can eventually read the entire input tape. A *prefix monotone* machine contains no blank end-marker in the input tape and it has a one-way write-only output tape.

A computation starts with the input head scanning the leftmost dummy cell. The output tape is written one symbol of $\{0,1\}$ at a time (the output grows monotonically with respect to the prefix ordering in $\{0,1\}^*$ as the computational time increases).

A *possibly infinite computation* is either a halting or a non halting computation. If the machine halts, the output of the computation is the finite string written on the output tape. Else, the output is either a finite string or an infinite sequence written on the output tape as a result of a never ending process. This leads to consider $\{0,1\}^{\leq\omega}$ as the output space. We define formally the possibly infinite computations on a (prefix) monotone machine:

Definition 2.1. *Let $\mathcal{M}$ be a Turing machine. $M(p)[t]$ is the current output of $\mathcal{M}$ on input p at stage t. Notice that $M(p)[t]$ does not require that the computation on input p halts.*

Remark 2.2. $M(p)[t]$ is a total primitive recursive function.

Definition 2.3. *Let $\mathcal{M}$ be a prefix machine. $M(p)[t]$ is the current output of $\mathcal{M}$ on input p at stage t* if it has not read beyond the end of p. *Otherwise, $M(p)[t]\uparrow$. Again, notice that $M(p)[t]$ does not require that the computation on input p halts.*

When we write $M(p)[t]$ we will use Definition 2.1 or 2.3 depending on whether $\mathcal{M}$ is prefix or not.

Remark 2.4. Notice that if $\mathcal{M}$ is a prefix machine then:

1. If $M(p)[t]\uparrow$ then $M(q)[u]\uparrow$ for all $q \preceq p$ and $u \geq t$.
2. If $M(p)[t]\downarrow$ then $M(q)[u]\downarrow$ for any $q \succeq p$ and $u \leq t$. Also, if at stage t, $\mathcal{M}$ reaches a halting state, then $M(p)[u]\downarrow = M(p)[t]$ for all $u \geq t$.
3. If $\mathcal{M}$ is prefix monotone then $M(p)[t] \preceq M(p)[t+1]$, in case $M(p)[t+1]\downarrow$.
4. $M(p)[t]$ is partial recursive and it has a recursive domain.

We introduce maps for the possibly infinite computation on a monotone machine (resp. prefix monotone machine). We restrict ourselves to possibly infinite computations which just read finitely many symbols from the input tape.

Definition 2.5. *1. Let $\mathcal{M}$ be a Turing machine (resp. prefix machine). The input/output behavior of $\mathcal{M}$ for* halting computations *is the partial recursive map $M : \{0,1\}^* \to \{0,1\}^*$ given by the usual computation of $\mathcal{M}$, i.e., $M(p)\downarrow$ iff $\mathcal{M}$ enters into a halting state on input p (resp. iff $\mathcal{M}$ enters into a halting state on input p without reading beyond p). If $M(p)\downarrow$ then $M(p) = M(p)[t]$ for some stage t at which $\mathcal{M}$ entered a halting state.*

2. Let $\mathcal{M}$ be a monotone machine (resp. prefix monotone machine). The input/output behavior of $\mathcal{M}$ for possibly infinite computations *is the map $M^\infty : \{0,1\}^* \to \{0,1\}^{\leq\omega}$ given by $M^\infty(p) = \lim_{t\to\infty} M(p)[t]$, where $M(p)[t]$ is as in Definition 2.1 (resp. Definition 2.3).*

Remark 2.6. 1. If $\mathcal{U}$ is any universal Turing machine with the ability of overwriting the output then by Shoenfield's Limit Lemma [10] follows that U^∞ computes all $\emptyset'$-recursive functions. This explains why we concentrate on monotone machines.

2. If $\mathcal{M}$ is a monotone machine (but not prefix monotone) then M^∞ is total and extends M.

Proposition 2.7. *Let $\mathcal{M}$ be a prefix monotone machine.*

1. domain(M) is closed under extension and its syntactical complexity is Σ^0_1.

2. domain(M^∞) is closed under extensions and its syntactical complexity is Π^0_1.

3. M^∞ extends M.

Proof. 1. is trivial.

2. $M^\infty(p)\downarrow \Leftrightarrow \forall t\ \mathcal{M}$ on input p does not read $p0$ and does not read $p1$. Clearly, *domain*(M^∞) is closed under extensions since if $M^\infty(p)\downarrow$ then $M^\infty(q)\downarrow = M^\infty(p)$ for every $q \succeq p$.

3. Since the machine $\mathcal{M}$ is not required to halt, M^∞ extends M.

□

Remark 2.8. Let $\mathcal{M}$ be a prefix monotone machine. An alternative definition of M and M^∞ would be to consider them with prefix-free domains (instead of closed under extensions).

- $M(p)\downarrow$ iff at some stage t $\mathcal{M}$ enters a halting state having read *exactly* p. If $M(p)\downarrow$ then its value is $M(p)[t]$ for such stage t.

- $M^\infty(p)\downarrow$ iff $\exists t$ at which $\mathcal{M}$ has read exactly p and for every $t' > t$ $\mathcal{M}$ does not read $p0$ nor $p1$. If $M^\infty(p)\downarrow$ then its value is $\sup\{M(p)[t] : t \geq 0\}$.

All properties of the complexity functions we study in this paper hold for this alternative definition.

We fix an effective enumeration of all tables of instructions of (prefix) monotone machines. This gives an effective $(\mathcal{M}_i)_{i\in\mathbb{N}}$. We fix the usual (prefix) monotone universal machine $\mathcal{U}$, which defines the functions $U(0^i1p) = M_i(p)$ and $U^\infty(0^i1p) = M_i^\infty(p)$ for halting and possibly infinite computations respectively. Recall that U^∞ is an extension of U. We also fix $U^{\emptyset'}$ a (prefix) monotone universal machine with an oracle for $\emptyset'$.

By Shoenfield's Limit Lemma [10] every $M^\infty : \{0,1\}^* \to \{0,1\}^*$ is recursive in $\emptyset'$. However, possibly infinite computations on monotone machines can not compute all $\emptyset'$-recursive functions.

For instance, the characteristic function of the halting problem can not be computed in the limit by a monotone machine. But there are other non-recursive functions that are obtainable by a possibly infinite computation. As an example, consider the Busy Beaver function in unary notation $bb : \mathbb{N} \to 1^*$, which on input $n \in \mathbb{N}$ it gives the maximum number of 1's produced by any Turing machine with n states which halts with no input. bb is not recursive, just $\emptyset'$-recursive. However $bb(n)$ is the output of a non halting computation which on input n, it simulates every Turing machine with n states and for each one that halts it updates, if necessary, the output with more 1's.

2.2 Program size complexities

Let's consider inputs as programs. The Kolmogorov or *program size* complexity [6] relative to a Turing machine $\mathcal{M}$ is the function $K_{\mathcal{M}} : \{0,1\}^* \to \mathbb{N}$ which maps a string s to the length of the shortest programs that output s. That is (recall Definition 2.5 item 1),

$$K_{\mathcal{M}}(s) = \begin{cases} \min\{|p| : M(p) = s\} & \text{if } s \text{ is in the range of } M \\ \infty & \text{otherwise} \end{cases}$$

In case $\mathcal{M}$ is a prefix machine, and M has a domain under extensions (or equivalently a prefix-free domain), we denote it $H_{\mathcal{M}}$ rather than $K_{\mathcal{M}}$ and we call it *prefix* complexity.

Since the subscript $\mathcal{M}$ can be any machine, even one equipped with an oracle, this is a definition of program size complexity for both effective or relative computability. In general, this program size complexity function is not recursive.

The invariance theorem (Kolmogorov, 1965 [6]) states that the universal Turing machine $\mathcal{U}$ is *asymptotically optimal* for program-size complexity, i.e.

$$\forall \text{ Turing machine } \mathcal{M}\ \exists c\ \forall s\ K_{\mathcal{U}}(s) \leq K_{\mathcal{M}}(s) + c$$

Its prefix variant (Chaitin, 1975 [2], Levin, 1974 [8]) states that the same result for prefix program size complexity.

For any pair of asymptotically optimal machines $\mathcal{M}$ and $\mathcal{N}$ there is a constant c such that $|K_{\mathcal{M}}(s) - K_{\mathcal{N}}(s)| \leq c$ for every string s. Thus, program size complexity on asymptotically optimal machines counts as an absolute measure of complexity, up to an additive constant. Exactly the same holds for prefix machines.

We shall write K (resp. H) for $K_{\mathcal{U}}$ (resp. $H_{\mathcal{U}}$) where $\mathcal{U}$ is some universal Turing (resp. universal prefix) machine.

The complexity for a universal machine (resp. prefix machine) with oracle A is notated as K^A (resp. H^A). As expected, the help of oracles leads to shorter programs.

Proposition 2.9. *There exists c such that $K^{\emptyset'}(s) \leq K(s) + c$ for all strings s. The same holds for H, $H^{\emptyset'}$ instead of $K, K^{\emptyset'}$.*

2.3 Program-size complexity for possibly infinite computations

Let $\mathcal{M}$ be a monotone machine, and M, M^∞ the respective maps for input/output behavior of $\mathcal{M}$ for halting computations and possibly infinite computations (see Definition 2.5).

Definition 2.10. $K^\infty_{\mathcal{M}} : \{0,1\}^{\leq\omega} \to \mathbb{N}$ *is the program size complexity for functions M^∞:*

$$K^\infty_{\mathcal{M}}(x) = \begin{cases} \min\{|p| : M^\infty(p) = x\} & \textit{if } x \textit{ is in the range of } M^\infty \\ \infty & \textit{otherwise} \end{cases}$$

For the universal $\mathcal{U}$ we drop subindexes and we simply write K^∞.

Remark 2.11. From Remark 2.6 (item 1) it is immediate that when $\mathcal{U}$ is a Turing machine with the ability of overwriting the output K^∞ coincides with $K^{\emptyset'}$, up to an additive constant.

The Invariance Theorem holds for K^∞: for all monotone machine $\mathcal{M}$ there is a c such that $\forall s \in \{0,1\}^{\leq\omega}$ $K^\infty(s) \leq K^\infty_{\mathcal{M}}(s) + c$.

In case $\mathcal{M}$ is prefix monotone machine we write $H^\infty_{\mathcal{M}}$ and $H^\infty_{\mathcal{U}} = H^\infty$. The Invariance Theorem also holds for H^∞.

We shall mention some known results that will be used constantly used in the next sections:

Proposition 2.12.

1. *$\exists c\ \forall s \in \{0,1\}^*$ $K(s) \leq |s| + c$*
2. *$\exists c\ \forall s \in \{0,1\}^*$ $K^{\emptyset'}(s) - c < K^\infty(s) < K(s) + c$, (see [1])*
3. *$\forall n\ \exists s \in \{0,1\}^*$ of length n such that $K(s) \geq n$. The same holds for $K^{\emptyset'}$ and K^∞*

Proposition 2.13. *Items 2 and 3 or Proposition 2.12 are still valid considering H, H^∞ and $H^{\emptyset'}$, but item 1 has to be restated as $\exists c\ \forall s \in \{0,1\}^*$ $H(s) \leq |\overline{s}| + c = 2|s| + c$, since the machine cannot read beyond the end of the input. It also holds (see [2]) $\exists c\ \forall s \in \{0,1\}^*$ $H(s) \leq H(|s|) + |s| + c$.*

3 Oscillations of K^∞

In this section we study some properties of the complexity function K^∞ and compare it with K and $K^{\emptyset'}$. In this section we work with monotone machines. We know $K^{\emptyset'} \leq K^\infty \leq K$ up to additive constants. The following results show that K^∞ is really in between $K^{\emptyset'}$ and K.

There are strings that separate the three complexity functions K, $K^{\emptyset'}$ and K^∞ arbitrarily:

Proposition 3.1. *For every c there is a string $s \in \{0,1\}^*$ such that*

$$K^{\emptyset'}(s) + c < K^\infty(s) < K(s) - c.$$

Proof. We know that for every n there is a string s of length n such that $K(s) \geq n$. Let d_n be the first string of length n in the lexicographic order satisfying this inequality, i.e., $d_n = \min\{s \in \{0,1\}^n : K(s) \geq n\}$. Let $f : \mathbb{N} \to \mathbb{N}$ be any recursive function with infinite range, and consider a machine $\mathcal{C}$ for infinite computations, which on input i does the following:

```
$j := 0$
Repeat
  Write $f(j)$
  Find a program $p$, $|p| \leq 2i$, such that $U(p) = f(j)$
  $j := j + 1$
```

The machine $\mathcal{C}$ on input i outputs (in the limit) the string $c_i = f(0)f(1)\ldots f(j_i)$, where $K(f(j_i)) > 2i$ and $\forall z, 0 \leq z < j_i : K(f(z)) \leq 2i$. For each i, we define $e_i = d_i c_i$.

Let's fix k and see there is an i_1 such that $\forall i \geq i_1 : K^\infty(e_i) - K^{\emptyset'}(e_i) > k$. On the one hand, we can compute d_i from i and a minimal program p such that $U^\infty(p) = e_i$ by simulating $U(p)$ until it outputs i bits into the output. If we codify the input as $\bar{i}p$ we obtain

$$i \leq K(d_i) \leq K^\infty(e_i) + 2\,|i| + \mathcal{O}(1). \tag{1}$$

On the other hand, with the help of the $\emptyset'$ oracle, we can compute e_i from i. Hence

$$K^{\emptyset'}(e_i) \leq |i| + \mathcal{O}(1). \tag{2}$$

From (1) and (2) we have $K^{\infty}(e_i) - K^{\emptyset'}(e_i) + \mathcal{O}(1) \geq i - 3\,|i|$ and then, there is i_1 such that for all $i \geq i_1$, $K^{\infty}(e_i) - K^{\emptyset'}(e_i) > k$.

Let's see now there is i_2 such that $\forall i \geq i_2 : K(e_i) - K^{\infty}(e_i) > k$. Given i and a shortest program p such that $U(p) = e_i$ we construct a machine that computes $f(j_i)$. Indeed, if we codify the input as $\bar{i}p$ the following machine does the work:

```
Obtain $i$
Compute $e := U(p)$
$s := e\lceil i$
$j := 0$
Repeat
  $s := sf(j)$
  If $s = e$ then write $f(j)$ and halt
  $j := j + 1$
```

Hence, for all i

$$2i < K(f(j_i)) \leq K(e_i) + 2\,|i| + \mathcal{O}(1). \tag{3}$$

Using the machine $\mathcal{C}$ we can construct a machine for infinite computations which computes e_i from a minimal program p such that $U(p) = d_i$. Then, for every i

$$K^{\infty}(e_i) \leq K(d_i) + \mathcal{O}(1) \leq i + \mathcal{O}(1). \tag{4}$$

From (3) and (4) we get $K(e_i) - K^{\infty}(e_i) + \mathcal{O}(1) > i - 2\,|i|$ so the difference between $K(e_i)$ and $K^{\infty}(e_i)$ can grow arbitrarily as we increase i. Let i_2 such that for all $i \geq i_2$ such that $K(e_i) - K^{\infty}(e_i) > k$.

Taking $i_0 = \max\{i_1, i_2\}$, we obtain $\forall i \geq i_0 : K^{\emptyset'}(e_i) + k < K^{\infty}(e_i) < K(e_i) - k$ and this completes the proof. □

The three complexity functions K, $K^{\emptyset'}$ and K^{∞} get close infinitely many times:

Proposition 3.2. *There is a constant c such that for every n:*

$$\exists s \in \{0,1\}^n : |K^{\emptyset'}(s) - K^{\infty}(s)| \leq c \wedge |K^{\infty}(s) - K(s)| \leq c.$$

Proof. Let s_n of length n such that $K^{\emptyset'}(s_n) \geq n$. Hence for each n, $K(s_n) \leq n + \mathcal{O}(1) \leq K^{\emptyset'}(s_n) + \mathcal{O}(1)$, and from Proposition 2.12, $K^{\infty}(s_n) - K^{\emptyset'}(s_n) \leq \mathcal{O}(1)$. Hence $|K^{\emptyset'}(s_n) - K^{\infty}(s_n)| \leq \mathcal{O}(1)$. In the same way, $K(s_n) - K^{\infty}(s_n) \leq K^{\emptyset'}(s_n) - K^{\infty}(s_n) + \mathcal{O}(1) \leq \mathcal{O}(1)$ and so $|K(s_n) - K^{\infty}(s_n)| \leq \mathcal{O}(1)$. We conclude that for all n $|K^{\emptyset'}(s_n) - K^{\infty}(s_n)| \leq \mathcal{O}(1)$ and $|K^{\infty}(s_n) - K(s_n)| \leq \mathcal{O}(1)$. □

For infinitely many strings, K and K^{∞} get close but they separate from $K^{\emptyset'}$ as much as we want:

Proposition 3.3. *There is a constant c such that for all m*

$$\exists s \in \{0,1\}^* : K(s) - K^{\emptyset'}(s) > m \wedge |K^{\infty}(s) - K(s)| < c.$$

Proof. We know that $\#\{s \in \{0,1\}^{n+2|n|} : K(s) < n\} < 2^n$ and then

$$\#\{s \in \{0,1\}^{n+2|n|} : K(s) \geq n\} > 2^{n+2|n|} - 2^n.$$

Let $S_n = \{\overline{|w|}w : w \in \{0,1\}^n\}$. Notice that, if $s \in S_n$, $|s| = n + 2|n|$. Clearly, $\#S_n = 2^n$. Assume by contradiction that there is n such that $S_n \cap \{s \in \{0,1\}^{n+2|n|} : K(s) \geq n\} = \emptyset$. Then $2^{n+2|n|} \geq \#S_n + \#\{s \in \{0,1\}^{n+2|n|} : K(s) \geq n\} > 2^{n+2|n|}$ which is impossible. For every n, let's define s_n

$$s_n = \min\{s \in S_n : K(s) \geq n\}. \tag{5}$$

Given a minimal program p such that $U^\infty(p) = s_n$, we can compute s_n in an effective way. The idea is to take advantage of the structure of s_n to know when U^∞ stops writing in its output tape: we simulate $U^\infty(p)$ until we detect $\overline{n}$ and we continue the simulation of U^∞ until we see it writes exactly n more bits. Then for each n, $K(s_n) \leq K^\infty(s_n) + \mathcal{O}(1)$ and from Proposition 2.12 we have that for all n the difference $|K(s_n) - K^\infty(s_n)|$ is bounded by a constant.

Using the $\emptyset'$ oracle, we can compute s_n from n. Hence $K^{\emptyset'}(s_n) \leq |n| + \mathcal{O}(1)$. From (5) we conclude $K(s_n) - K^{\emptyset'}(s_n) + \mathcal{O}(1) \geq n - |n|$. Thus, the difference between $K(s_n)$ and $K^{\emptyset'}(s_n)$ can be made arbitrarily large. □

Infinitely many times K^∞ and $K^{\emptyset'}$ get close but they separate from K arbitrarily:

Proposition 3.4. *There is a constant c such that for each m*

$$\exists s \in \{0,1\}^* : K(s) - K^\infty(s) > m \wedge |K^\infty(s) - K^{\emptyset'}(s)| < c.$$

Proof. As in the proof of Proposition 3.1, consider a recursive f with infinite range, let $c_n = \overline{n}f(0)f(1)\dots f(j_n)$, and slightly modify the machine $\mathcal{C}$ such that on input i, it first writes $\overline{i}$ and then it continues writing $f(j)$ until it finds a j_i such that $K(f(j_i)) > 2i$ and $\forall z, 0 \leq z < j_i$: $K(f(z)) \leq 2i$. Thus, given $str(n)$, we can compute n and then c_n in the limit. Hence for every n

$$K^\infty(c_n) \leq |str(n)| + \mathcal{O}(1). \tag{6}$$

Given an $\emptyset'$ oracle minimal program for c_n, we can compute $str(n)$ in an oracle machine. Then for every n

$$K^{\emptyset'}(str(n)) \leq K^{\emptyset'}(c_n) + \mathcal{O}(1). \tag{7}$$

We define $m_n = \min\{s \in \{0,1\}^n : K^{\emptyset'}(s) \geq n\}$ and $s_n = c_{str^{-1}(m_n)}$. From (7) we know

$$n \leq K^{\emptyset'}(m_n) \leq K^{\emptyset'}(s_n) + \mathcal{O}(1) \tag{8}$$

and from (6) we have

$$K^\infty(s_n) \leq |m_n| + \mathcal{O}(1). \tag{9}$$

From (8) and (9) we obtain $K^\infty(s_n) - K^{\emptyset'}(s_n) \leq \mathcal{O}(1)$ and by Proposition 2.12 we conclude that for all n, $|K^\infty(s_n) - K^{\emptyset'}(s_n)| \leq \mathcal{O}(1)$. In the same way as we did in Proposition 3.1, we construct an effective machine that outputs $f(j_n)$ from a shortest program such that $U(p) = c_n$, but in this case the machine gets n from the input itself (we don't need to pass it as a distinct parameter). Hence for all n, $2n < K(f(j_n)) \leq K(c_n) + \mathcal{O}(1)$ and in particular for $n = str^{-1}(m_n)$ we have $2str^{-1}(m_n) < K(s_n) + \mathcal{O}(1)$. Since for each string s, $|s| \leq str^{-1}(s)$ we have $2|m_n| < K(s_n) + \mathcal{O}(1)$. From (9) and recalling that $|m_n| = n$, we have $K(s_n) - K^\infty(s_n) + \mathcal{O}(1) > n$. Thus, the difference between $K(s_n)$ and $K^\infty(s_n)$ grows as n increases. □

It is known that the complexity function K is nearly continuous in the length and lexicographic order on $\{0,1\}^*$, i.e. for all n $|K(str(n)) - K(str(n+1))| = \mathcal{O}(1)$.

We have the following result.

Proposition 3.5. *For all n*

$$|K^\infty(str(n)) - K^\infty(str(n+1))| \leq 2K(|str(n)|) + \mathcal{O}(1)$$

Proof. Consider the following monotone machine $\mathcal{M}$ with input $\overline{p}q$:

```
Obtain $y = U(p)$
Simulate $z = U^\infty(q)$ till it outputs $y$ bits
Write $str(str^{-1}(z) + 1)$
```

Let $p, q \in \{0,1\}^*$ such that $U(p) = |str(n)|$ and $U^\infty(q) = str(n)$. Then, $M^\infty(\overline{p}q) = str(n+1)$ and $K^\infty(str(n+1)) \leq K^\infty(str(n)) + 2K(|str(n)|) + \mathcal{O}(1)$.

Similarly, if $\mathcal{M}$ above instead of writing $str(str^{-1}(z)+1)$, it writes $str(str^{-1}(z)-1)$, we conclude $K^\infty(str(n)) \leq K^\infty(str(n+1)) + 2K(|str(n+1)|) + \mathcal{O}(1)$.

Since, $|K(str(n)) - K(str(n+1))| \leq \mathcal{O}(1)$, we conclude

$$|K^\infty(str(n)) - K^\infty(str(n+1))| \leq 2K(|str(n)|) + \mathcal{O}(1).$$

□

The advantage of K^∞ over K can be seen along the initial segments of every recursive sequence: if $A \in \{0,1\}^\omega$ is recursive then there are infinitely many n's such that $K(A\restriction n) - K^\infty(A\restriction n) > c$, for an arbitrary c.

Proposition 3.6. *Let $A \in \{0,1\}^\omega$ be a recursive sequence. Then*

$$\limsup_{n\to\infty} K(A\restriction n) - K^\infty(A\restriction n) = \infty.$$

Proof. Let $f : \mathbb{N} \to \{0,1\}$ a total recursive function such that $f(n)$ is the n-th bit of A. Let's consider the following monotone machine $\mathcal{M}$ with input p:

```
Obtain $n := U(p)$
Write $A\restriction(str^{-1}(0^n) - 1)$
For $s := 0^n$ to $1^n$ in lexicographic order
    Write $f(str^{-1}(s))$
    Search for a program $p$ such that $|p| < n$ and $U(p) = s$
```

If $U(p) = n$, then $M^\infty(p)$ outputs $A\restriction k_n$ for some k_n such that $2^n \leq k_n < 2^{n+1}$, since for all n there is a string of length n with K-complexity greater than or equal to n. Let us fix n. On one hand, $K^\infty(A\restriction k_n) \leq |n| + \mathcal{O}(1)$. On the other, $K(A\restriction k_n) + \mathcal{O}(1) \geq n$, because we can compute the first string of length n in the lexicographic order with K-complexity $\geq n$ from a program for $A\restriction k_n$. Hence, for each n, $K(A\restriction k_n) - H^\infty(A\restriction k_n) + \mathcal{O}(1) \geq n - |n|$. □

4 Program size complexity for possibly infinite computations on prefix monotone machines

We turn now to the analysis of H^∞ and we work with prefix monotone machines. We will se that the H^∞-version of Propositions 3.1 and 3.3 are still valid.

There are strings that separate the three complexity functions H, $H^{\emptyset'}$ and H^∞ arbitrarily:

Proposition 4.1. *For every c there is a string s such that*

$$H^{\emptyset'}(s) + c < H^\infty(s) < H(s) - c.$$

Proof. The proof is essentially the same that Proposition 3.1 but using prefix monotone machines. Let $c_n = f(0)f(1)\ldots f(j_n)$ and slightly change the instructions of machine $\mathcal{C}$ putting $H(f(j_n)) > 3n$ and $\forall z, 0 \le z < j_n : H(f(z)) \le 3n$. Let $d_n = \min\{s \in \{0,1\}^n : H(s) \ge n\}$ and $e_n = d_n c_n$. Assume p is a shortest program such that $U^\infty(p) = e_i$. Consider the effective machine which on input $\bar{i}p$ does the following:

```
Obtain $i$
Simulate $x \restriction i = U^\infty(p)$ until it outputs $i$ bits
Print $x$ and halt
```

Then, we have

$$i \le H(d_i) \le H^\infty(e_i) + 2\,|i| + \mathcal{O}(1). \tag{10}$$

If we codify the input of the oracle computation of Proposition 3.1 by duplicating the bits (now we can't use just $|i|$ bits to codify i), inequality (2) becomes

$$H^{\emptyset'}(e_i) \le 2\,|i| + \mathcal{O}(1). \tag{11}$$

From (10) and (11) we have $H^\infty(e_i) - H^{\emptyset'}(e_i) + \mathcal{O}(1) \ge i - 4\,|i|$, and so we can make the difference between $H^\infty(e_i)$ and $H^{\emptyset'}(e_i)$ as large as we want. To show that the difference between $H(e_i)$ and $H^\infty(e_i)$ can also be made arbitrarily large, we replace (3) by

$$3i < H(f(j_i)) \le H(e_i) + 2\,|i| + \mathcal{O}(1) \tag{12}$$

and recalling that for each string s, $H(s) \le 2\,|s| + \mathcal{O}(1)$, inequality (4) is replaced by

$$H^\infty(e_i) \le H(d_i) + \mathcal{O}(1) \le 2i + \mathcal{O}(1). \tag{13}$$

From (12) and (13) we get $H(e_i) - H^\infty(e_i) + \mathcal{O}(1) > i - 2\,|i|$. □

For infinitely many strings, H and H^∞ get close but they separate from $H^{\emptyset'}$ as much as we want:

Proposition 4.2. *There is a constant c such that for all m*

$$\exists s \in \{0,1\}^* : H(s) - H^{\emptyset'}(s) > m \wedge |H^\infty(s) - H(s)| \le c.$$

Proof. The idea of the proof is the same as the one in Proposition 3.3. We redefine s_n (see (5)):

$$s_n = \min\{s \in S_n : H(s) \ge n\}. \tag{14}$$

As in Proposition 3.3, we consider the same program, but using prefix monotone machines. Identically we obtain becomes $H(s_n) \le H^\infty(s_n) + \mathcal{O}(1)$ and from Proposition 2.13 we have $|H(s_n) - H^\infty(s_n)| \le \mathcal{O}(1)$. Instead of $K^{\emptyset'}(s_n) \le |n| + \mathcal{O}(1)$ we obtain $H^{\emptyset'}(s_n) \le 2\,|n| + \mathcal{O}(1)$ and from (14) we conclude $H(s_n) - H^{\emptyset'}(s_n) \ge n - 2\,|n| + \mathcal{O}(1)$. Thus, the difference between $H(s_n)$ and $H^{\emptyset'}(s_n)$ grows as n increases. □

We can show the following weaker H^∞-version of Proposition 3.4:

Proposition 4.3. *There is a sequence* $(s_n)_{n\in\mathbb{N}}$ *such that* $\lim_{n\to\infty} H(s_n) - H^\infty(s_n) = \infty$ *and* $|H^\infty(s_n) - H^{\emptyset'}(s_n)| \leq H(n) + \mathcal{O}(1)$

Proof. The idea is similar to the proof of Proposition 3.4, but making j_i such that $H(f(j_i)) > 3i$ and $\forall z, 0 \leq z < j_i : H(f(z)) \leq 3i$. We replace (6) by

$$H^\infty(c_n) \leq H(str(n)) + \mathcal{O}(1) \tag{15}$$

since the machine for infinite computations can compute n and c_n from a shortest program p such that $U(p) = str(n)$. There is an oracle machine that computes $str(n)$ from a minimal oracle program for c_n. Then, restating (7), we have for every n

$$H^{\emptyset'}(str(n)) \leq H^{\emptyset'}(c_n) + \mathcal{O}(1). \tag{16}$$

Let $m_n = \min\{s \in \{0,1\}^n : H^{\emptyset'}(s) \geq n\}$ and $s_n = c_{str^{-1}(m_n)}$. From (15) and (16) we have

$$H^\infty(s_n) - H^{\emptyset'}(s_n) \leq H(m_n) - H^{\emptyset'}(m_n) + \mathcal{O}(1) \leq H(m_n) - n + \mathcal{O}(1)$$

and, since $H(m_n) \leq H(|m_n|) + |m_n| + \mathcal{O}(1)$ we conclude $H^\infty(s_n) - H^{\emptyset'}(s_n) \leq H(n) + \mathcal{O}(1)$. We can construct an effective machine that computes $f(j_n)$ from a minimal program for U which outputs c_n. From (15) we have $H(s_n) - H^\infty(s_n) + \mathcal{O}(1) > 3n - H(m_n)$. Since for all n, $H(m_n) \leq 2\,|m_n| + \mathcal{O}(1) = 2n + \mathcal{O}(1)$, we get $H(s_n) - H^\infty(s_n) + \mathcal{O}(1) > n$ and hence the difference can be made arbitrarily large. □

It is easy to see that Proposition 3.6 is also true for H^∞. Proposition 3.5 for H^∞ is still valid considering $H(|str(n)|) + \mathcal{O}(1)$ as the upper bound. Let's see some other properties that are only valid for prefix monotone machines:

Proposition 4.4. *For all strings s and t*

1. $H(s) \leq H^\infty(s) + H(|s|) + \mathcal{O}(1)$.
2. $H^\infty(ts) \leq H^\infty(s) + H(t) + \mathcal{O}(1)$.
3. $H^\infty(s) \leq H^\infty(st) + H(|t|) + \mathcal{O}(1)$.
4. $H^\infty(s) \leq H^\infty(st) + H^\infty(|s|) + \mathcal{O}(1)$.

Proof.

1. Let $p, q \in \{0,1\}^*$ such that $U^\infty(p) = s$ and $U(q) = |s|$. Then there is a machine that first simulates $U(q)$ to obtain $|s|$, then it starts a simulation of $U^\infty(p)$ writing its output on the output tape, until it has written $|s|$ symbols, and then halts.

2. Let $p, q \in \{0,1\}^*$ such that $U^\infty(p) = s$ and $U(q) = t$. Then there is a machine that first simulates $U(q)$ until it halts and prints $U(q)$ on the output tape. Then , it starts a simulation of $U^\infty(p)$ writing its output on the on the output tape.

3. Let $p, q \in \{0,1\}^*$ such that $U^\infty(p) = st$ and $U(q) = |t|$. Then there is a machine that first simulates $U(q)$ until it halts to obtain $|t|$. Then it starts a simulation of $U^\infty(p)$ such that at each stage n of the simulation it writes the symbols needed to leave $U(p)[n] \upharpoonright |U(p)[n]| - |t|$ on the output tape.

4. Consider the following monotone machine:

```
$t := 1$ ; $v := \lambda$ ; $w := \lambda$
Repeat
  if $U(v)[t]$ asks for reading then $v := vb$
  if $U(w)[t]$ asks for reading then $w := wb$
      where $b$ is the next bit in the input
  extend the actual output to $U(w)[t] \upharpoonright (U(v)[t])$
```

If p and q are shortest programs such that $U^{\infty}(p) = |s|$ and $U^{\infty}(q) = st$ respectively, then we can interleave p and q in a way such that at each stage t, $v \preceq p$ and $w \preceq q$ (notice that eventually $v = p$ and $w = q$). Thus, this machine will compute s and will never read more than $H^{\infty}(st)+H^{\infty}(|s|)$ bits.

□

References

[1] V. Becher, S. Daicz, & G. Chaitin. A highly random number. In C. S. Calude, M. J. Dineen, and S. Sburlan, editors, *Combinatorics, Computability and Logic: Proceedings of the Third Discrete Mathematics and Theoretical Computer Science Conference (DMTCS'01)*, 55–68. Springer-Verlag London, 2001.

[2] G. J. Chaitin. A theory of program size formally identical to information theory, *J. ACM*, vol.22, 329–340, 1975.

[3] G. J. Chaitin. Algorithmic entropy of sets, *Computers & Mathematics with Applications*, vol.2, 233–245, 1976.

[4] G.J. Chaitin. Information-theoretical characterizations of recursive infinite strings. *Theoretical Computer Science*, 2:45–48,1976.

[5] H.P. Katsef. and M. Sipser Several results in program size complexity. *Theoretical Computer Science*, 15, 291–309, 1981.

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

[7] L.A. Levin. On the Concept of a Random Sequence. *Doklady Akad. Nauk SSSR*,

[8] 14(5), 1413–1416, 1973.L.A. Levin. Laws of Information Conservation (Non-growth) and Aspects of the Foundations of Probability Theory. *Problems of Information Transmission*, 10:3, pp. 206–210, 1974.

[9] M. Li and P. Vitanyi. *An introduction to Kolmogorov complexity and its applications*, Springer, Amsterdam, 1997 (2d edition).

[10] J.R. M. Shoenfield. On degrees of unsovability. *Annals of Mathematics*, vol. 69, 644–653, 1959.