Prefix-like Complexities of Finite and Infinite Sequences on Generalized Turing Machines

Alexey Chernov
Jürgen Schmidhuber

Technical Report No. IDSIA-11-05-2005
March 2005

IDSIA / USI-SUPSI
Dalle Molle Institute for Artificial Intelligence
Galleria 2, 6928 Manno, Switzerland

This work was sponsored by SNF grant 200020-100259/1.

Prefix-like Complexities of Finite and Infinite Sequences on Generalized Turing Machines

Alexey Chernov
Jürgen Schmidhuber*

March 2005

Abstract

Generalized Turing machines (GTMs) are a variant of non-halting Turing machines, by computational power similar to machines with the oracle for the halting problem. GTMs allow a definition of a kind of descriptive (Kolmogorov) complexity that is uniform for finite and infinite sequences. There are several natural modifications of the definition (as there are several monotone complexities). This paper studies these definitions and compares complexities defined with the help of GTMs and complexities defined with the help of oracle machines.
Keywords: Kolmogorov complexity, generalized Turing machine, infinite sequence, non-halting computation.

1 Introduction

Recently, Schmidhuber [7, 6] introduced several modifications of Turing machines, in particular, generalized Turing machines (GTMs), and defined appropriate complexities. GTMs are machines that never halt and are allowed to change their output with the only requirement being that each bit eventually stabilizes. GTMs are a natural model for computability in the limit, and thus GTMs are closely related to Turing machines with the oracle for the halting problem. These two models are equivalent in the case of computing functions on naturals. However, the situation is different in the case of functions on binary sequences with additional requirements such as the self-delimiting property. Functions of this kind are used in algorithmic information theory for defining prefix and monotone complexity (see [2, 1], and for comprehensive presentation and more references [3]).

Originally, Kolmogorov defined the complexity of an object as the minimal size of its description with respect to some effective specifying method (mode of description), i. e. mapping from the set of descriptions to the set of objects. For the specifying method, one can assume various computing devices (as Turing machines, maybe, non-halting, supplied with an oracle, etc.), implementing this method. Every assumption leads to a variant of complexity. Restricting oneself to values defined up to a bounded additive term, one can speak about complexity with respect to a certain class of machines (containing a universal one).

For a machine of a fixed computational power, one can deal with the machine input and the input size in various ways. To give an example, let us consider an abstract (Turing) machine that has one (infinite in one direction) input tape containing only zeros and ones, can read the input tape bit by bit, and generates some object. There were studied at least three variants of 'description' (actually, definitions of the machine input and its size) of a generated object with respect to such a machine.

*IDSIA & TU Munich, Boltzmannstr. 3, 85748 Garching, München, Germany

1. Prefix mode of description.

   Informally, the machine must separate the description ("a significant part of the input") from the rest of the input and generate the object. Formally, the description is the beginning of the input tape actually read during generating the object. The size of the description is its length. In this case, the set of descriptions (of all objects) is prefix-free: any element is not a prefix of another element.

2. Weakly prefix mode of description.

   Informally, the machine must generate the object given a description coupled with other information, as before, but now the machine does not have to separate the description explicitly. Formally, the description is a finite sequence such that the machine generates the object if the input tape contains any prolongation of this sequence; the size of description is its length. The set of descriptions is not prefix-free, but if one description is a prefix of another, they describe the same object. Any prefix mode of description is also a weakly prefix one, but the converse does not hold. In the weakly prefix case, the set of "minimal descriptions" (that are not prolongations of other descriptions) is prefix-free, but, generally speaking, this set cannot be enumerated in an effective way, in contrast to the prefix case.

   For machines with halting computations, the weakly prefix mode of description can be interpreted with the help of an "interactive" model of input. A machine does not have an input tape and does not read input, but obtains its input bit by bit from the user. The input sequence may be finite or infinite, the user decides when the next bit is provided. (One can imagine that the input is given by some physical or computational process.) It is required that the result of any computation does not depend on the timing of obtaining input bits, but depends on the input sequence only. Clearly, if the machine generates some object on input x, the machine generates the same object on all prolongations of x (since the user can provide x at the beginning, and the rest only when the machine has halted). On the other hand, one can assume (slightly modifying a machine) the following property: if the machine generates some object on all prolongations of x, then it generates the same object also on x (the proof idea: consider the set of y such that the machine halts on xy, but does not halt on any beginning of xy; this set contains a beginning of any infinite prolongation of x and is finite, since otherwise the machine does not halt on some infinite prolongation of x; so, one can enumerate the set of all x with the required property). Now it is easy to see that the weakly prefix descriptions are exactly the input sequences of this new machine.

3. Probabilistic mode of description.

   Informally, the input tape is interpreted as a source of random bits, and the probability of generating some object serves as measure of its complexity (complex objects are rare). Formally, a description is any set of infinite sequences such that the machine generates the object when the input tape contains this sequence. The size of the description is the negative logarithm of its uniform measure. If x is a weakly prefix description of size n, then the set of all prolongations of x is a probabilistic description of size n. On the other hand, for any collection of non-overlapping probabilistic descriptions one can find a prefix-free set of sequences (like prefix descriptions), but not effectively, generally speaking.

For any machine model, one can consider these three input modes and get three complexities, the prefix-mode complexity is the greatest one, and the probabilistic-mode complexity is the least one. Actually, two important results in algorithmic complexity theory can be interpreted as comparing these input modes for specific machine models. These results concern prefix and monotone complexity, and they show that the three kinds of complexity may coincide or may be different depending on the machine model.

In both cases the standard Turing machine is used, but the result of the computation is defined differently.

For prefix complexity, the machine is said to generate an object if it prints the object and halts (thus, the object must be finite; the objects can be identified with finite sequences). Levin's coding theorem [2] implies that in this case all three input modes lead to the same complexity (up to an additive constant). Clear distinction between prefix and weakly prefix definitions of prefix complexity is provided in [8].

The monotone complexity provides an opposite example. The objects are finite and infinite sequences, the machine prints its output bit by bit, and may halt or non halt. We say that the machine generates a finite sequence if it appears at some moment on the output tape (later the machine may prolong the output); the machine generates an infinite sequence if it generates all its finite prefixes during the computation. Now the probabilistic mode gives the value known as the logarithm of a priori semimeasure on continuous space. The weakly prefix mode is used for the main definition of monotone complexity in [1] (which is referred to as type 2 monotone complexity in [3, pp. 312–313]). The prefix mode is used for definition of monotone complexity in the main text of [3] (where also referred to as type 3 monotone complexity at pp. 312–313). All three values coincide up to a logarithmic additive term. Gács [1] proved that the difference between the monotone complexity (under his definition) and the logarithm of a priori semimeasure (between the probabilistic and weakly prefix modes in our terms) is unbounded on finite sequences. It is unknown whether the two monotone complexities (the weakly prefix and prefix modes) coincide; all known theorems hold for both.

In this paper, the three definitions are studied for GTMs, and for finite and infinite sequences as objects. Informally speaking, it turns out that the prefix-mode complexity differs from the weakly prefix-mode complexity by a logarithmic term, for both finite and infinite sequences. For finite sequences, the weakly prefix-mode complexity coincides with the probabilistic-mode complexity up to a constant; for infinite sequences, they coincide up to a logarithm, and the question about better accuracy is open. A visual presentation of the results is given in a table in Section 6. The diagram also presents relations to complexities with the oracle for the halting problem: these relations are the main tool used in the proofs. The rest of the paper is organized as follows: Section 2 contains definitions of GTM and complexities; Section 3 provides technical lemmas connecting GTMs and oracle machines; the main results for finite and infinite sequences are proved in Sections 4 and 5, respectively.

2 Definition of GTM and Complexities

Denote by $\mathbb{B}^*$ the space of finite sequences over the binary alphabet $\mathbb{B} = \{0, 1\}$ and by $\mathbb{B}^\infty$ the space of infinite sequences. Denote by $\ell(x)$ the length of $x \in \mathbb{B}^*$, and put $\ell(x) = \infty$ for $x \in \mathbb{B}^\infty$. For $x \in \mathbb{B}^* \cup \mathbb{B}^\infty$ and $n \in \mathbb{N}$, let x_n be the n-th bit of x (0 or 1) if $n \leq \ell(x)$ and a special symbol "blank" otherwise.

A *generalized Turing machine* (GTM) is a machine with one read-only input tape, several work tapes, and one output tape; all tapes are infinite in one direction. A GTM never halts; it reads the input tape bit by bit from left to right; it can print on the output tape in any order, i.e. can print or erase symbols in any cell many times. For a machine T and an input sequence $p \in \mathbb{B}^\infty$, denote by $T_t(p)$ the finite binary sequence[1] on the output tape at the moment t. We say that a GTM T on an input $p \in \mathbb{B}^\infty$ *converges* to $x \in \mathbb{B}^* \cup \mathbb{B}^\infty$ (write $T(p) \rightsquigarrow x$) if $\forall n \, \exists t_n \, \forall t > t_n \; [T_t(p)]_n = x_n$ (informally speaking, each bit of the output stabilizes eventually). The sequence x is called the *output* of T on p, and p is called a *program* for x.

[1]For technical convenience, we assume that the content of the output tape is always a finite sequence of zeros and ones followed by blanks (without blanks inside). This assumption is not restrictive: for any T one can consider T' that emulates T but postpones printing a bit to the output tape if this violates the requirement; this modification does not affect the results of converging computations.

We say that a GTM T on an input $p \in \mathbb{B}^*$ *strongly converges* to $x \in \mathbb{B}^* \cup \mathbb{B}^\infty$ (write $T(p) \rightrightarrows x$) if $T(p0^\infty) \rightsquigarrow x$ and T reads exactly p during the computation. We say that a GTM T on an input $p \in \mathbb{B}^*$ *weakly converges* to $x \in \mathbb{B}^* \cup \mathbb{B}^\infty$ (write $T(p) \rightarrowtail x$) if $T(pq) \rightsquigarrow x$ for any $q \in \mathbb{B}^\infty$. These two kinds of convergence are straightforward implementation on GTMs of the prefix and weakly prefix modes. Clearly, if $T(p) \rightrightarrows x$, then $T(p) \rightarrowtail x$.

Recall that for weakly prefix mode we had two equivalent models in the case of halting computations. For non-halting computations, there are several (non-equivalent) possibilities to define some analog of the "interactive" machine (where the user sometimes gives a new bit). The choice of the following variant of definition is based on the relating GTMs to oracle machines below. We say that a GTM T on an input $p \in \mathbb{B}^*$ *uniformly weakly converges* to $x \in \mathbb{B}^* \cup \mathbb{B}^\infty$ (write $T(p) \twoheadrightarrow x$) if $\forall n\, \exists t_n\, \forall t > t_n \forall q \in \mathbb{B}^\infty\ [T_t(pq)]_n = x_n$. The last formula differs from the definition of weak convergence only *by the order of quantifiers* ($T(p) \rightarrowtail x$ iff $\forall q \in \mathbb{B}^\infty\, \forall n\, \exists t_n\, \forall t > t_n\ [T_t(pq)]_n = x_n$). Informally speaking, in the uniform case, the moment of stabilizing a certain output bit is determined by some finite part of the input; on the contrary, in the non-uniform case, for any beginning of the input sequence there may be prolongation where this bit will change. It is easy to see that uniform weak convergence can be implemented by some kind of "interactive" machine; for non-uniform weak convergence this is unclear. As is shown below, strong, weak, and uniform weak convergence give different classes of computable functions.

The standard reasoning shows that there is a universal GTM U. For $x \in \mathbb{B}^* \cup \mathbb{B}^\infty$, we define complexities corresponding to the prefix and weakly prefix modes:

$$
\begin{aligned}
K^G_{\rightrightarrows}(x) &= \min\{\ell(p) \mid U(p) \rightrightarrows x\}\,,\\
K^G_{\rightarrowtail}(x) &= \min\{\ell(p) \mid U(p) \rightarrowtail x\}\,,\\
K^G_{\twoheadrightarrow}(x) &= \min\{\ell(p) \mid U(p) \twoheadrightarrow x\}\,.
\end{aligned}
$$

The idea of the probabilistic modes is reflected by a priori GTM-probability

$$P^G(x) = \lambda\{p \mid U(p) \rightsquigarrow x\}\,,$$

where λ is the uniform measure on $\mathbb{B}^\infty$; we do not introduce a special sign for the corresponding complexity $-\log_2 P^G(x)$. These complexity measures are well-defined in the sense that if U is replaced by any other GTM, then $K^G_{\rightrightarrows}(x)$, $K^G_{\rightarrowtail}(x)$, $K^G_{\twoheadrightarrow}(x)$, and $-\log_2 P^G(x)$ can decrease at most by a constant, the same for all x. Clearly, for $x \in \mathbb{B}^* \cup \mathbb{B}^\infty$,

$$-\log_2 P^G(x) \le K^G_{\rightarrowtail}(x) \le K^G_{\twoheadrightarrow}(p) \le K^G_{\rightrightarrows}(x)\,. \tag{1}$$

As usual in complexity theory, many relations hold up to a bounded additive term, which is denoted by $\overset{+}{=}$, $\overset{+}{\le}$ in the sequel.

The complexity $K^G_{\rightrightarrows}(x)$ coincides with $K^G(x)$ originally defined in [7]. Poland in [4] suggested a definition of complexity for enumerable output machines similar to $K^G_{\rightarrowtail}(x)$ in our case. Poland proved that for enumerable output machines, his complexity is equal to the logarithm of a priori measure up to a constant (for GTMs such an equality was not known even with logarithmic accuracy).

3 Oracle Machines and GTMs

Recall that an oracle Turing machine is a Turing machine with one additional operation: for any number n, the machine can check whether n belongs to a fixed set called an oracle (or, equivalently, the machine can get any bit of a certain infinite sequence). The oracle is not a part of the machine, and the same machine can work with different oracles, but the result of the computation depends on the oracle used.

Denote by $\mathsf{0}'$ the oracle for the halting problem (see [5] for comprehensive consideration). Technically, we assume some fixed effective enumeration of Turing machines (without oracle) that do not read their input tape, and the n-th bit of the infinite binary sequence $\mathsf{0}'$ is 1 iff the n-th machine halts.

It is well known (see [5]) that functions computable with $\mathsf{0}'$ as an oracle are exactly the functions computable in the limit. GTM also expresses the idea of computability in the limit: it tries different answers and eventually (in the limit) gives the correct answer. But if the input is provided without delimiters, there is some difference. In the case of the prefix mode, an oracle machine can use the oracle to detect the end of input (and to stop reading in time), while a GTM does not differ from an ordinary Turing machine in this aspect. In the case of the probabilistic mode, a GTM has an advantage, since its computation is non-halting and the GTM can use the entire (infinite) input sequence.

It turns out that the uniform weak convergence for GTMs is equivalent to the weak convergence for machines with the oracle for the halting problem.

Lemma 1. *1. For any GTM T there exists an oracle machine $\tilde{T}$ with two input tapes*[2] *such that: For any $p \in \mathbb{B}^*$, if $T(p) \twoheadrightarrow x$, then $\forall q \in \mathbb{B}^\infty$ $\tilde{T}^{\mathsf{0}'}(pq, n)$ halts and prints $x_{1:n}$.*
2. For any oracle machine T with two input tapes there exists a GTM $\tilde{T}$ with the following properties. For any $p \in \mathbb{B}^\infty$, if $T^{\mathsf{0}'}(p, n)$ halts and prints $x_{1:n}$, then $\tilde{T}(p) \rightsquigarrow x$. If $T^{\mathsf{0}'}(pq, n)$ halts and prints $x_{1:n}$ for $p \in \mathbb{B}^*$ and for all $q \in \mathbb{B}^\infty$, then $\tilde{T}(p) \twoheadrightarrow x$.

Proof. 1. Recall that $T(p) \twoheadrightarrow x$ means that $\forall n\, \exists t_n\, \forall t > t_n \forall q \in \mathbb{B}^\infty\ [T_t(pq)]_n = x_n$. Note that the predicate $\forall t > t_n \forall q \in \mathbb{B}^\infty\ [T_t(pq)]_n = x_n$ is a $\mathsf{0}'$-decidable, and as well as the predicate $A_T(t_0, p, n, x) = \forall m \le n \forall t > t_0 \forall q \in \mathbb{B}^\infty [T_t(pq)]_m = x_m$.

For each $t_0 = 1, 2, \ldots$, the machine $\tilde{T}$ tries to find x of length $\le n$ such that $A_T(t_0, p, n, x)$ is true, where p is the beginning of the input of length t_0 (actually, $\tilde{T}$ reads a new input bit at each step). If x is found, $\tilde{T}$ prints this (unique) x and halts. Otherwise, $\tilde{T}$ proceeds to next t_0.
2. The machine $\tilde{T}$ computes the n-th output bit emulating the running of T on $\tilde{T}$ input and n. To answer T's oracle queries, $\tilde{T}$ enumerates $\mathsf{0}'$ and answers according to the current version. In the beginning the current version of $\mathsf{0}'$ consists of zeros, then some zeros change to ones (when the corresponding machines halt). If a changed bit was used in the emulated computation, the emulation starts, with the correct bits. To prove that any output bit stabilizes eventually observe that any finite computation uses only a finite number of oracle queries, and every bit of the oracle enumerated by $\tilde{T}$ changes at most once and then becomes correct.

Now suppose that for some $p \in \mathbb{B}^*$ and for all n the computation $T^{\mathsf{0}'}(pq, n)$ halts with the same result for all $q \in \mathbb{B}^\infty$. The construction above provides that $\tilde{T}(p) \rightarrowtail x$. To prove $\tilde{T}(p) \twoheadrightarrow x$, note that there exists m such that on any input beginning with p, $T\mathsf{0}'$ reads not more than m input bits. Thus, the quantifier over q (in the definition of convergence) is actually a finite quantifier in this case and can be moved through the quantifier over t_n. □

Remark. For any GTM, one could construct also an equivalent oracle machine, but for complexity applications Lemma 1 is sufficient.

Note that one cannot replace uniform weak convergence by weak convergence in the first statement of Lemma 1, because the behavior of the GTM can always depend on the *unread* part of the input, which is unacceptable for halting machines. Actually, there are functions computable

[2]The first tape provides the GTM input, and the second tape gives the required length of the oracle machine output (in any self-delimited form). This second tape is introduced to provide a uniform formulation for finite and infinite outputs in terms of halting machine.

on GTMs in the sense of weak convergence, but not on machines with the oracle $0'$; for example, let $f(n)$ be 0 if the n-th *oracle* machine with the oracle $0'$ halts on *all* inputs, and $f(n)$ be undefined otherwise.

The next lemma relates probabilistic GTM-descriptions to $0'$-machines. For any halting machine, as usual prefix and monotone machines, it is easy to show that the probability of generating a certain object is enumerable from below, since the pre-image of any object is a countable union of cylinder sets (sets of all infinite sequences with a fixed prefix q), see [3]. In contrast, Example 7 in [4] shows that the set $\{p \in \mathbb{B}^\infty \mid \forall n\, \exists t_n\, \forall t > t_n\; [T_t(p)]_n = x_n\}$ may contain no cylinder set for some GTM T. Nevertheless, GTM-probabilities turned out to be $0'$-enumerable from below.

Lemma 2. *For any GTM T, the value*

$$R(x,m) = \lambda\big(\{p \in \mathbb{B}^\infty \mid \forall n \le m\, \exists t_n\, \forall t > t_n\; [T_t(p)]_n = x_n\}\big)$$

is $0'$-enumerable from below for any $x \in \mathbb{B}^* \cup \mathbb{B}^\infty$ and $m \in \mathbb{N}$.

Proof. First, we eliminate the bounded quantifier taking $t_0 = \max_{n \le m} t_n$:

$$R(x,m) = \lambda\big(\{p \in \mathbb{B}^\infty \mid \exists t_0 \forall t \ge t_0\; [T_t(p)]_{1:m} = x_{1:m}\}\big)\,.$$

Then, transforming quantifiers $\exists$ and $\forall$ to union and intersection of sets, and using continuity of the measure, we get

$$R(x,m) = \sup_{t_0} \inf_{t_1 \ge t_0} \lambda\Big(\bigcap_{t=t_0}^{t_1} \{p \in \mathbb{B}^\infty \mid [T_t(p)]_{1:m} = x_{1:m}\}\Big)\,.$$

Until time t_1, the machine T can read only the first t_1 bits of input, therefore $\lambda\big(\bigcap_{t=t_0}^{t_1}\{p \in \mathbb{B}^\infty \mid [T_t(p)]_{1:m} = x_{1:m}\}\big)$ is computable. The infimum of the last value is $0'$-computable, and thus $R(x,m)$ is $0'$-enumerable from below. □

4 Finite Sequences

The prefix complexity of $x \in \mathbb{B}^*$ is defined as

$$K(x) = \min\{\ell(p) \mid V(p) = x\}\,,$$

where V is a universal prefix machine. The universal enumerable semimeasure on $\mathbb{B}^*$ as a discrete space is

$$m(x) = \lambda\{pq \mid V(p) = x, p \in \mathbb{B}^*, q \in \mathbb{B}^\infty\}$$

(see [3] for details). For any other enumerable semimeasure μ there is a constant c such that $m(x) \ge c\mu(x)$ for all x. By Levin's theorem [2], $K(x) \overset{\pm}{=} -\log_2 m(x)$. Relativizing w.r.t. the oracle $0'$, we get the oracle complexity $K^{0'}(x) = \min\{\ell(p) \mid V^{0'}(p) = x\}$ and the maximal $0'$-enumerable semimeasure $m^{0'}(x)$, and $K^{0'}(x) \overset{\pm}{=} -\log_2 m^{0'}(x)$.

The following two theorems provide a complete description of relations between GTM- and $0'$-complexities for finite sequences.

Theorem 1. *For $x \in \mathbb{B}^*$,*

$$-\log_2 m^{0'}(x) \overset{\pm}{=} -\log_2 P^G(x) \overset{\pm}{=} K^G_{\rightarrowtail}(x) \overset{\pm}{=} K^G_{\twoheadrightarrow}(x) \overset{\pm}{=} K^{0'}(x)\,.$$

Proof. Lemma 1 applied to universal machines for $n = \ell(x) + 1$ shows that $K^{0'}(x) \overset{\pm}{=} K^G_{\twoheadrightarrow}(x)$.

Since $K^{0'}(x) \overset{\pm}{=} -\log_2 m^{0'}(x)$, the inequality (1) will give the statement if we prove that $-\log m^{0'}(x) \overset{+}{\leq} -\log_2 P^G(x)$. By Lemma 2 we have $P^G(x) = R(x, \ell(x) + 1)$ (generally speaking, $R(x, m)$ includes the measure of all computations that generate $x_{1:m}$ and its prolongations and also of some nonconverging computations; but if $x_{1:m}$ ends with "blank", all further symbols on the output tape also have to be "blanks", and thus the computation that stabilizes to $x_{1:m}$ in the first m output bits must converge, and moreover, converge to $x_{1:m-1}$). Thus, $P^G(x)$ is a $0'$-enumerable semimeasure and therefore it is less (up to a multiplicative constant) than $m^{0'}(x)$. □

Theorem 2. *For $x \in \mathbb{B}^*$,*

$$K^{0'}(x) \overset{+}{\leq} K^G_{\rightrightarrows}(x) \overset{+}{\leq} K^{0'}(x) + K(K^{0'}(x)) \,.$$

Both bounds are almost tight; namely, there is a constant C such that for infinitely many x it holds $K^{0'}(x) \geq K^G_{\rightrightarrows}(x) - 2\log_2\log_2 K^{0'}(x) - C$; and for infinitely many x it holds $K^G_{\rightrightarrows}(x) \geq K^{0'}(x) + K(K^{0'}(x)) - 2\log_2 K(K^{0'}(x)) - C$.

Remark. One can prove better tightness statements, where $\log_2\log_2 K^{0'}$ and $\log_2 K(K^{0'}(x))$ are replaced by expressions with any number of logarithms.

Proof. First, $K^{0'}(x) \overset{+}{\leq} K^G_{\rightrightarrows}(x)$ since $K^G_{\twoheadrightarrow}(x) \overset{+}{\leq} K^G_{\rightrightarrows}(x)$. On the other hand, having the length of uniform weak description, GTM can stop reading the input tape and assume the further input bit are zeros (this does not change the answer). Thus, $K^G_{\rightrightarrows}(x) \leq K^G_{\twoheadrightarrow}(x) + K(K^G_{\twoheadrightarrow}(x)) + O(1)$.

The tightness of the bound $K^{0'}(x) \overset{+}{\leq} K^G_{\rightrightarrows}(x)$ follows from the fact that $K^G_{\rightrightarrows}(x) \overset{+}{\leq} K(x)$ and $K^{0'}(x)$ is almost equal to $K(x)$ for $0'$-random x (i. e. x of given length that have maximal $0'$-complexity). The tightness of $K^G_{\rightrightarrows}(x) \overset{+}{\leq} K^{0'}(x) + K(K^{0'}(x))$ is proved very similarly to Theorem 5 below (for infinite sequences). □

5 Infinite Sequences

One can define the complexity of infinite sequences using halting machines, in the same manner as in Lemma 1, with the help of a machine that has two inputs, a sequence and a number, and generates the initial segment (specified by the number) of the infinite output sequence. It is easy to see that this approach leads to the same complexity measure as monotone complexity (defined below) restricted to infinite sequences.

Monotone machines are non-halting machines that print their output (finite or infinite) bit by bit (see [3] for details). Let W be a universal monotone machine. For $p \in \mathbb{B}^\infty$, by $W(p)$ denote the (complete) output of W on the input sequence p; for $p \in \mathbb{B}^*$ by $W(p)$ denote the output of W printed after reading p and not more. In [3], the most popular monograph on Kolmogorov complexity, the monotone complexity of $x \in \mathbb{B}^* \cup \mathbb{B}^\infty$ is defined as

$$Km(x) = \min\{\ell(p) \mid p \in \mathbb{B}^*, W(p) = xy, y \in \mathbb{B}^* \cup \mathbb{B}^\infty\} \,.$$

Gács [1] uses another definition,

$$Km_{\mathrm{I}}(x) = \min\{\ell(p) \mid p \in \mathbb{B}^*, \forall q \in \mathbb{B}^\infty W(pq) = xy, y \in \mathbb{B}^* \cup \mathbb{B}^\infty\} \,.$$

(Obviously, Km and Km_{I} correspond to the prefix and weakly prefix modes of description in our terms).

The universal a priori probability of $x \in \mathbb{B}^* \cup \mathbb{B}^\infty$ is

$$M(x) = \lambda\{p \in \mathbb{B}^\infty \mid W(p) = xy, y \in \mathbb{B}^* \cup \mathbb{B}^\infty\}$$

(note that M is defined on continuous tree-like space, in contrast to m). It is known that $-\log_2 M(x) \overset{+}{\le} Km_{\mathrm{I}}(x) \overset{+}{\le} Km(x)$ and $Km(x) \overset{+}{\le} -\log_2 M(x) + \log_2 Km(x)$. Gács [1] showed that the difference between $-\log_2 M(x)$ and $Km_{\mathrm{I}}(x)$ is unbounded for $x \in \mathbb{B}^*$, in contrast to the prefix complexity case. The proof does not imply any relation between $-\log_2 M(x)$ and $Km_{\mathrm{I}}(x)$ for infinite sequences, and the question about coincidence of $-\log_2 M(x)$, $Km_{\mathrm{I}}(x)$, and $Km(x)$ for $x \in \mathbb{B}^\infty$ is open.

After relativization w.r.t. the oracle $\mathsf{0}'$, we get $Km^{\mathsf{0}'}$ and $M^{\mathsf{0}'}$. Note that for $x \in \mathbb{B}^\infty$, $Km^{\mathsf{0}'}(x)$ and $-\log_2 M^{\mathsf{0}'}(x)$ are finite iff x is $\mathsf{0}'$-computable.

Theorem 3. *For $x \in \mathbb{B}^\infty$,*

$$-\log_2 P^G(x) \overset{\pm}{=} -\log_2 M^{\mathsf{0}'}(x)\,.$$

Proof. By the second part of Lemma 1, a GTM can emulate a $\mathsf{0}'$-machine, therefore $M^{\mathsf{0}'}(x) = \lambda\{p \in \mathbb{B}^\infty \mid W^{\mathsf{0}'}(p) = x\} \le C \cdot P^G(x)$.

To prove the other direction, let us define an auxiliary semimeasure on $x \in \mathbb{B}^* \cup \mathbb{B}^\infty$: $M^G(x) = \lambda\{p \in \mathbb{B}^\infty \mid \forall n \le \ell(x)\, \exists t_n\, \forall t > t_n\, [U_t(p)]_n = x_n\}$. Clearly, for $x \in \mathbb{B}^\infty$ we have $P^G(x) = M^G(x) \le \lim_{n\to\infty} M^G(x_{1:n})$ and $M^{\mathsf{0}'}(x) = \lim_{n\to\infty} M^{\mathsf{0}'}(x_{1:n})$. By Lemma 2, $M^G(x) = R(x, \ell(x))$ is $\mathsf{0}'$-enumerable on $x \in \mathbb{B}^*$. Trivially, M^G is a semimeasure on continuous space ($M^G(x) \ge M^G(x0) + M^G(x1)$), therefore, $M^G(x) \le C \cdot M^{\mathsf{0}'}(x)$ for $x \in \mathbb{B}^*$. □

Theorem 4. *For $x \in \mathbb{B}^\infty$,*

$$K^G_{\twoheadrightarrow}(x) \overset{\pm}{=} Km^{\mathsf{0}'}_{\mathrm{I}}(x)\,.$$

Proof. Follows immediately from Lemma 1. □

Theorem 5. *For $x \in \mathbb{B}^\infty$,*

$$Km^{\mathsf{0}'}(x) \overset{+}{\le} K^G_{\rightrightarrows}(x) \overset{+}{\le} Km^{\mathsf{0}'}(x) + K(Km^{\mathsf{0}'}(x))\,.$$

The upper bound is almost tight: for some constant C and for infinitely many x it holds $K^G_{\rightrightarrows}(x) \ge Km^{\mathsf{0}'}(x) + K(Km^{\mathsf{0}'}(x)) - 2\log_2 K(Km^{\mathsf{0}'}(x)) - C$.

Proof. Both bounds hold for the same reason as in Theorem 2. To prove the tightness of the upper bound, consider the set

$$A = \{p \in \mathbb{B}^* \mid U \text{ reads } p \text{ and not more}\}\,.$$

Evidently, A is $\mathsf{0}'$-decidable and prefix-free.

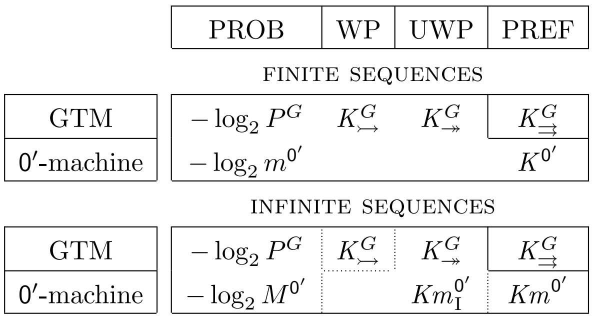
Put $A_n = \{p \in A \mid \ell(p) \le n\}$. Since A is prefix-free, we have $|A_n| \le 2^n/(n \log_2 n)$ for infinitely many n. Since A is $\mathsf{0}'$-decidable, there is a $\mathsf{0}'$-computable sequence of naturals n_k such that $|A_{n_k}| \le 2^{n_k}/(n_k \log_2 n_k)$ and $n_k \ge 2^{k^2}$.

Now, using the diagonal process, we can construct $x \in \mathbb{B}^\infty$ such that $K^G_{\rightrightarrows}(x) > n_k$ and $Km^{\mathsf{0}'}(x) \le n_k - \log_2 n_k + C$, where the constant C does not depend on k. Put $B_k = \{p \in \mathbb{B}^* \mid \ell(p) \le n_k,\ \exists y \in \mathbb{B}^* \cup \mathbb{B}^\infty\, U(p) \rightrightarrows y\}$, and let us provide that $U(p) \not\rightrightarrows x$ for all $p \in B_k$. To find the n-th bit of x, we enumerate $p \in A_{n_k}$ such that the first n bits of $U(p)$ stabilize (maybe, to "blank"), until $|B_k|$ such p's are enumerated (obviously, for n large enough, we enumerate all p from B_k and only them). Then we take the j-th of these p (in lexicographical order), where $j \equiv n \pmod{|B_k|}$, take the n-th output bit of $U(p)$, and set x_n to a different value.

Trivially, $K^G_{\rightrightarrows}(x) > n_k$. Let us estimate $Km^{0'}(x)$. The construction above gives an algorithm for computing x with the help of the oracle $0'$ (to determine when bits stabilize) if n_k and $|B_k|$ are given, thus $Km^{0'}(x) \le K^{0'}(\langle n_k, |B_k|\rangle) + O(1)$. The following code is a prefix $0'$-description of the pair $\langle n_k, |B_k|\rangle$: optimal prefix code of k and then exactly l_k digits that represent $|B_k|$, where $l_k = n_k - \log_2 n_k - \log_2\log_2 n_k$. Since $|B_k| \le |A_{n_k}| \le 2^{n_k}/(n_k \log_2 n_k)$, we have enough codes for $|B_k|$. On the other hand, we can find n_k given k using the oracle $0'$, and then compute l_k, therefore the encoding is prefix. Hence $K^{0'}(\langle k, |B_k|\rangle) \overset{+}{\le} K(k) + l_k \overset{+}{\le} 2\log_2 k + n_k - \log_2 n_k - \log_2\log_2 n_k \overset{+}{\le} n_k - \log_2 n_k$. Since $n_k - \log_2 n_k$ is a monotonically increasing function of n_k, we have $Km^{0'}(x) \overset{+}{\le} K^G_{\rightrightarrows}(x) - \log_2 K^G_{\rightrightarrows}(x)$. Finally, $K^G_{\rightrightarrows}(x) \overset{+}{\ge} Km^{0'}(x) + \log_2 K^G_{\rightrightarrows}(x) \overset{+}{\ge} Km^{0'}(x) + \log_2 Km^{0'}(x) \overset{+}{\ge} Km^{0'}(x) + K(Km^{0'}(x)) - 2\log_2 K(Km^{0'}(x))$. $\square$

6 Conclusion

In this paper several GTM-complexities were introduced. The known relations between them, and also between them and $0'$-complexities, are summarized in the following diagram.



FINITE SEQUENCES

	PROB	WP	UWP	PREF
GTM	$-\log_2 P^G$	$K^G_{\rightarrowtail}$	$K^G_{\twoheadrightarrow}$	$K^G_{\rightrightarrows}$
$0'$-machine	$-\log_2 m^{0'}$			$K^{0'}$

INFINITE SEQUENCES

	PROB	WP	UWP	PREF
GTM	$-\log_2 P^G$	$K^G_{\rightarrowtail}$	$K^G_{\twoheadrightarrow}$	$K^G_{\rightrightarrows}$
$0'$-machine	$-\log_2 M^{0'}$		$Km^{0'}_{\mathrm{I}}$	$Km^{0'}$

The columns correspond to probabilistic, weakly prefix, uniform weakly prefix, and prefix descriptions, respectively. The borders between cells describe relation between the corresponding values: no border means that the values are equal up to a constant, the solid line separates values that differ by a logarithmic term, the dashed line shows that the relation is unknown. Note that the values in the cells grow from left to right in every row.

The main **open question** is if weak GTM-complexities ($K^G_{\twoheadrightarrow}$, $K^G_{\rightarrowtail}$, and $-\log_2 P^G$) coincide on infinite sequences. A closely related question is if $Km^{0'}(x)$ and $Km^{0'}_{\mathrm{I}}(x)$ coincide with $-\log_2 M^{0'}(x)$ for $x \in \mathbb{B}^\infty$, and if this holds for non-relativized $Km(x)$, $Km_{\mathrm{I}}(x)$, and $-\log_2 M(x)$. If they do coincide, this would make a surprising contrast to Gács' result on the monotone complexity of finite sequences.

Acknowledgments. The authors are grateful to Marcus Hutter, Jan Poland, and Shane Legg for useful discussions and proofreading.

References

[1] P. Gács. On the relation between descriptional complexity and algorithmic probability. *Theoretical Computer Science*, 22:71–93, 1983.

[2] L. A. Levin. Laws of information (nongrowth) and aspects of the foundation of probability theory. *Problems of Information Transmission*, 10(3):206–210, 1974.

[3] M. Li and P. M. B. Vitányi. *An Introduction to Kolmogorov Complexity and its Applications (2nd edition).* Springer, 1997.

[4] J. Poland. A Coding Theorem for Enumerating Output Machines. *Information Processing Letters*, 91(4):157–161, 2004.

[5] H. Rogers, Jr. *Theory of Recursive Functions and Effective Computability.* McGraw-Hill, New York, 1967.

[6] J. Schmidhuber. Algorithmic theories of everything. Technical Report IDSIA-20-00, quant-ph/0011122, IDSIA, Manno (Lugano), Switzerland, 2000.

[7] J. Schmidhuber. Hierarchies of generalized Kolmogorov complexities and nonenumerable universal measures computable in the limit. *International Journal of Foundations of Computer Science*, 13(4):587–612, 2002.

[8] N. K. Vereshchagin, A. Shen, and V. A. Uspensky. Lecture Notes on Kolmogorov Complexity. Unpublished, http://lpcs.math.msu.su/∼ver/kolmbook.