Prefix-like Complexities and Computability in the Limit[⋆]

Alexey Chernov[1] and Jürgen Schmidhuber[1,2]

[1] IDSIA, Galleria 2, 6928 Manno, Switzerland
[2] TU Munich, Boltzmannstr. 3, 85748 Garching, München, Germany

Abstract. Computability in the limit represents the non-plus-ultra of constructive describability. It is well known that the limit computable functions on naturals are exactly those computable with the oracle for the halting problem. However, prefix (Kolmogorov) complexities defined with respect to these two models may differ. We introduce and compare several natural variations of prefix complexity definitions based on generalized Turing machines embodying the idea of limit computability, as well as complexities based on oracle machines, for both finite and infinite sequences.
Keywords: Kolmogorov complexity, limit computability, generalized Turing machine, non-halting computation.

1 Introduction

Limit computable functions are functions representable as a limit of a computable function over an extra argument. They are a well-known extension of the standard notion of computability, and appear in many contexts, e. g. [1, 13, 6]. It was argued that many human activities (such as program debugging) produce the final result only in the limit, and that limit computability is the non-plus-ultra of constructive describability—even more powerful models of computation cannot be classified as constructive any more, e. g. [10, 12].

Several authors considered variants of Kolmogorov complexity based on limit computations and computations with the oracle for the halting problem $\mathsf{0}'$, e. g. [2, 3, 8]. Limit computable functions are exactly functions computable with the oracle $\mathsf{0}'$ by the well-known Shoenfield limit lemma [19]. In algorithmic information theory, however, we cannot simply apply the Shoenfield lemma to replace limit computability by $\mathsf{0}'$-computability. The reason is that the lemma is proven for functions on naturals, whereas definitions of prefix and monotone complexity require functions on sequences satisfying some kind of prefix property—see [7, 11, 15, 16].

In the present paper, we prove equalities and inequalities for prefix complexity with the oracle $\mathsf{0}'$ and several natural variations of prefix complexity based on generalized Turing machines (GTMs), one of the natural models for limit computability [18] (also compare [4]). GTMs never halt and are allowed to rewrite their previous output, with the only requirement being that each output bit

⋆ This work was sponsored by SNF grant 200020-107590/1.

eventually stabilizes forever. We prove that depending on the subtleties of the definition, the corresponding complexities may differ up to a logarithmic term.

Originally, Kolmogorov [14] 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. a mapping from the set of descriptions to the set of objects. The specifying methods may be implemented on various computing devices (such as ordinary TMs, possibly non-halting TMs, possibly supplied with an oracle, etc.). All yield different complexity variants. 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).

Even for a given machine, however, we obtain different complexity variants defining the machine input and the input size in various ways. Let us consider a generic machine with a single one-way infinite input tape containing only zeros and ones, reading the input tape bit by bit, and generating some output object. Researchers used (sometimes implicitly) at least three variants of "input mode":

Prefix mode. Informally, the machine has to separate the description ("a significant part of the input") from the rest of the input to generate the object. Formally, the description is the initial part of the input string actually read during production of the object. The size of the description is its length; the set of possible descriptions is prefix-free: no description is prefix of another.

Weak prefix mode. Informally, the machine does not have to separate the informative part of the description from the uninformative rest. 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 the description is its length. The set of descriptions is not prefix-free, but if one description is prefix of another, they will describe the same object. Every prefix mode description is also a weak prefix one, but the converse does not hold. In the weak prefix case, the set of "minimal descriptions" (those that are not prolongations of other descriptions) is prefix-free, but in general this set cannot be enumerated in an effective way, unlike in the prefix case.

For machines with halting computations, the weak prefix mode can be interpreted with the help of an "interactive" model of the input. Instead of reading the input off an input tape, the machine obtains its finite or infinite input sequence bit by bit from the user (or some physical or computational process) deciding when the next bit is provided. The result of any computation may not depend on the timing of the input bits, but on the input sequence only. Clearly, if the machine generates some object on input x, the machine will generate the same object on all prolongations of x (since the user may provide x at the beginning, and the rest once the machine has halted). On the other hand, one may assume the following property: if the machine generates some object on all prolongations of x, then it will generate the same object also on x. (Proof idea: consider the set of y such that the machine halts on xy, but does not halt on any prefix of xy; this set contains a prefix of any infinite prolongation of x and is finite, since otherwise the machine does not halt on some infinite prolongation of x; hence one can enumerate the set of all x with the required property.) Clearly, the input sequences of this new machine are exactly the weak prefix descriptions.

Probabilistic mode. Informally, the input tape is interpreted as a source of random bits, and the probability of generating some object serves to measure its complexity (complex objects are unlikely). Formally, a description is any set of infinite sequences such that the machine generates the object when the input tape contains an element of this set. The size of the description is the negative logarithm of its uniform measure. If x is a weak 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 there is a prefix-free set of sequences (such as prefix descriptions), but in general one cannot find it effectively.

For any machine model, one may consider these three input modes and get three complexity types. The prefix mode complexity is the largest, the probabilistic mode complexity the smallest. In fact, two important results of algorithmic complexity theory can be interpreted as comparing these input modes for specific machine models. These results concern prefix and monotone complexity, and provide examples of machine models where the three kinds of complexity coincide and where they are different. In both cases the standard TM is used, and the difference is in the definition of the computational result (the computed object) only.

For prefix complexity, the machine is said to generate an object if the machine prints the object and halts (thus, objects are identifiable with finite sequences). Levin's remarkable Coding Theorem [15] (see also [7]) implies that in this case all three input modes lead to the same complexity (up to an additive constant). Informally speaking, the theorem says that the probability of guessing a program for the given data is essentially equivalent to the one of guessing its shortest program. From a technical point of view, the Levin theorem allows us to simplify many proofs, every time using the most suitable of the three definitions (see [22] for an extensive discussion of prefix and weak prefix modes for prefix complexity).

The monotone complexity provides an opposite example. Now the objects are finite or infinite sequences, the TM prints its output bit by bit, and may or may not halt. We say the machine generates a finite sequence if this sequence appears on the output tape at some point (subsequently 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 the a priori semimeasure on binary sequences. The weak prefix mode is used for the main definition of monotone complexity by Gács in [11] (which is referred to as type 2 monotone complexity in [16, pp. 312–313]). The prefix mode is used for definition of monotone complexity in the main text of [16] (where it also referred to as type 3 monotone complexity, pp. 312–313). All three values coincide up to a logarithmic additive term. Gács [11] proved that the difference between the monotone complexity (under his definition) and the logarithm of the a priori semimeasure (between the probabilistic and weak prefix modes in our terms) is unbounded on finite sequences. It is unknown whether the two monotone complexities (the weak prefix and prefix modes) coincide; all known theorems hold for both.

Here the three modes are studied for both finite and infinite output sequences computed on GTMs. Informally speaking, it turns out that the prefix-mode complexity differs from the weak prefix-mode complexity by a logarithmic term, for both finite and infinite sequences. For finite sequences, the weak prefix-mode complexity coincides with the probabilistic-mode complexity up to a constant. For infinite sequences, they coincide up to a logarithm. It remains an open question whether this bound is tight. A diagram in Sect. 6 displays the results including relations to complexities with the oracle $\mathsf{0}'$ for the halting problem. The rest of the paper is organized as follows: Sect. 2 contains definitions of GTM and complexities; Sect. 3 provides technical lemmas connecting GTMs and oracle machines; the main results are proven in Sect. 4 for finite sequences and in Sect. 5 for infinite sequences.

2 Definition of GTMs 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[3] 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.

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 reflect the prefix and weak prefix modes. Clearly, if $T(p) \rightrightarrows x$, then $T(p) \rightarrowtail x$.

Recall that for the weak prefix mode we had two equivalent models in the case of halting computations. For non-halting computations, there are several (non-equivalent) ways of defining some analogue of the "interactive" machine (where the user sometimes provides a new bit). The following variant is chosen for conveniently 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 stabilization of a certain output bit is determined by some

[3] 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 the requirement is violated; clearly, the results of converging computations are not affected.

finite part of the input. It is easy to see that uniform weak convergence can be implemented by some kind of "interactive" machine. In the non-uniform case, however, for any initial part of the input sequence there may be a prolongation where this bit will change. As is shown below, strong, weak, and uniform weak convergence yield different classes of computable functions.

By the standard argument, there is a universal GTM U. For $x \in \mathbb{B}^* \cup \mathbb{B}^\infty$, we define complexities corresponding to the prefix and weak 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 probabilistic mode is reflected by the 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 $x \in \mathbb{B}^* \cup \mathbb{B}^\infty$,

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

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 [18]. Poland [17] suggested a definition of complexity for enumerable output machines similar to $K^G_{\rightarrowtail}(x)$ in our case, and proved that for enumerable output machines, his complexity is equal to the logarithm of the a priori measure up to a constant (for GTMs such an equality was not known even with logarithmic accuracy [18]).

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: the machine can work with different oracles but the result depends on the oracle used.

Denote by $\mathsf{0}'$ the oracle for the halting problem (see [20]). By the Shoenfield limit lemma [19], $\mathsf{0}'$-computable functions are exactly the limit computable functions. The GTM also embodies the idea of computability in the limit: it tries various answers and eventually (in the limit) gives the correct answer. But if the input is provided without delimiters, a difference arises. In the prefix mode case, an oracle machine can use the oracle to detect the input end (and to stop reading in time), while a GTM does not differ from an ordinary TM in this respect. In the probabilistic mode case, a GTM has an advantage since it does not halt and may use the entire (infinite) input sequence.

It turns out that uniform weak convergence for GTMs is equivalent to weak convergence for machines with the oracle for the halting problem[4].

Lemma 1. *1. For any GTM T there exists an oracle machine $\tilde{T}$ with two input tapes*[5] *such that: For any $p \in \mathbb{B}^*$, if $T(p) \twoheadrightarrow x$, then $\forall q \in \mathbb{B}^\infty\ \forall n\ \tilde{T}^{0'}(pq, n)$ 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 $T^{0'}(p, n)$ halts and prints $x_{1:n}$ for all n, then $\tilde{T}(p) \rightsquigarrow x$. If $T^{0'}(pq, n)$ halts and prints $x_{1:n}$ for some $p \in \mathbb{B}^*$, for all $q \in \mathbb{B}^\infty$ and for all n, then $\tilde{T}(p) \twoheadrightarrow x$.

Note that one cannot replace uniform weak convergence by weak convergence in the first statement of Lemma 1, because the behavior of the GTM may always depend on the *unread* part of the input, which is unacceptable for halting machines. Actually, there are functions computable 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 let $f(n)$ be undefined otherwise (compare [10]).

The next lemma relates probabilistic GTM-descriptions to $0'$-machines. For any halting machine (such as traditional 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 [16]. In contrast, Example 7 in [17] 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 $m \in \mathbb{N}$.

4 Finite Sequences

The prefix complexity of $x \in \mathbb{B}^*$ is $K(x) = \min\{\ell(p) \mid V(p) = x\}$, where V is a universal prefix machine. The maximal (universal) enumerable semimeasure on naturals (we fix some encoding of naturals by finite binary sequences) is $m(x) =$

[4] It was mentioned in [18] that oracle complexity $K^{0'}$ equals GTM-complexity $K^G (\overset{\pm}{=} K^G_{\rightrightarrows})$, but without proof and without specifying accuracy. Surprisingly, our present refinement of the connection between oracle machines and GTMs is enough to refute (for $x \in \mathbb{B}^*$) Conjecture 5.3 from [18], namely, that $P^G(x) = O(2^{-K^G_{\rightrightarrows}(x)})$.

[5] The first tape provides the GTM input, the second the required length of the oracle machine output (in any self-delimiting form). This second tape is added to formulate statements about halting machines uniformly for finite and infinite outputs.

$\lambda\{pq \mid V(p) = x, p \in \mathbb{B}^*, q \in \mathbb{B}^\infty\}$ (see [16] for details). For any other enumerable semimeasure on naturals μ there is a constant c such that $m(x) \geq c\mu(x)$ for all x. By the Levin theorem [15], $K(x) \stackrel{\pm}{=} -\log_2 m(x)$. Relativizing w.r.t. the oracle $\mathsf{0}'$, we get the oracle complexity $K^{\mathsf{0}'}(x) = \min\{\ell(p) \mid V^{\mathsf{0}'}(p) = x\}$, the maximal $\mathsf{0}'$-enumerable semimeasure $m^{\mathsf{0}'}(x)$, and $K^{\mathsf{0}'}(x) \stackrel{\pm}{=} -\log_2 m^{\mathsf{0}'}(x)$.

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

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

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

Theorem 2. *For $x \in \mathbb{B}^*$, it holds $K^{\mathsf{0}'}(x) \stackrel{+}{\leq} K^G_{\rightrightarrows}(x) \stackrel{+}{\leq} K^{\mathsf{0}'}(x) + K(K^{\mathsf{0}'}(x))$. Both bounds are almost tight; namely, for some constant C and for infinitely many x it holds that $K^{\mathsf{0}'}(x) \geq K^G_{\rightrightarrows}(x) - 2\log_2\log_2 K^{\mathsf{0}'}(x) - C$; and for infinitely many x it holds that $K^G_{\rightrightarrows}(x) \geq K^{\mathsf{0}'}(x) + K(K^{\mathsf{0}'}(x)) - 2\log_2 K(K^{\mathsf{0}'}(x)) - C$.*

Note. (One can get stronger tightness statements where $\log_2 K(K^{\mathsf{0}'}(x))$ and $\log_2\log_2 K^{\mathsf{0}'}$ are replaced by expressions with any number of logarithms.)

5 Infinite Sequences

The complexity of infinite sequences can be defined with the help of halting machines having two inputs and generating the initial segment of the infinite output sequence (as in Lemma 1). It follows easily, however, that this approach will lead to the usual monotone complexity but restricted to infinite sequences.

Monotone machines are non-halting machines that print their output (finite or infinite) bit by bit (see [16] 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 just p and nothing else. In the book [16], the monotone complexity of $x \in \mathbb{B}^* \cup \mathbb{B}^\infty$ is defined as $Km(x) = \min\{\ell(p) \mid p \in \mathbb{B}^*, W(p) = xy, y \in \mathbb{B}^* \cup \mathbb{B}^\infty\}$ (corresponding to the prefix mode of description in our terms). Gács [11] used another definition, $Km_{\mathrm{I}}(x) = \min\{\ell(p) \mid p \in \mathbb{B}^*, \forall q \in \mathbb{B}^\infty W(pq) = xy, y \in \mathbb{B}^* \cup \mathbb{B}^\infty\}$ (corresponding to the weak prefix mode of description). The universal (a priori) probability of $x \in \mathbb{B}^* \cup \mathbb{B}^\infty$ is $M(x) = \lambda\{p \in \mathbb{B}^\infty \mid W(p) = xy, y \in \mathbb{B}^* \cup \mathbb{B}^\infty\}$. It is known that $-\log_2 M(x) \stackrel{+}{\leq} Km_{\mathrm{I}}(x) \stackrel{+}{\leq} Km(x) \stackrel{+}{\leq} -\log_2 M(x) + \log_2 Km(x)$. Gács [11] showed that the difference between $-\log_2 M(x)$ and $Km_{\mathrm{I}}(x)$ is unbounded for $x \in \mathbb{B}^*$ (unlike the prefix complexity case). The proof does not imply any relation between $-\log_2 M(x)$ and $Km_{\mathrm{I}}(x)$ for $x \in \mathbb{B}^\infty$, and the question about coincidence of $-\log_2 M(x)$, $Km_{\mathrm{I}}(x)$, and $Km(x)$ for $x \in \mathbb{B}^\infty$ remains 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 $x \in \mathbb{B}^\infty$, it holds $-\log_2 P^G(x) \stackrel{\pm}{=} -\log_2 M^{\mathsf{0}'}(x)$.*

Theorem 4. *For $x \in \mathbb{B}^\infty$, it holds $K^G_{\twoheadrightarrow}(x) \stackrel{\pm}{=} Km^{\mathsf{0}'}_{\mathrm{I}}(x)$.*

Theorem 5. *For $x \in \mathbb{B}^\infty$, it holds $Km^{0'}(x) \overset{+}{\leq} K^G_{\rightrightarrows}(x) \overset{+}{\leq} Km^{0'}(x) + K(Km^{0'}(x))$. The upper bound is almost tight: for some constant C and for infinitely many x it holds $K^G_{\rightrightarrows}(x) \geq Km^{0'}(x) + K(Km^{0'}(x)) - 2\log_2 K(Km^{0'}(x)) - C$.*

6 Conclusion

Generalized Turing machines (GTMs) are a natural model for computability in the limit, and hence are closely related to machines with the oracle $0'$. It turns out, however, that there is no single obvious way of formally specifying a prefix complexity based on GTMs. Instead there are several closely related but slightly different ways that all seem natural. This paper introduced and studied them, exhibiting several relations between them, and also between them and $0'$-complexities, as summarized by the following diagram:

	PROB	WP	UWP	PREF
	FINITE SEQUENCES			
GTM	$-\log_2 P^G$	$K^G_{\rightarrowtail}$	$K^G_{\twoheadrightarrow}$	$K^G_{\rightrightarrows}$
$0'$-machine	$-\log_2 m^{0'}$			$K^{0'}$
	INFINITE SEQUENCES			
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, weak prefix, uniform weak 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. The values in the cells grow from left to right in every row.

The diagram shows a significant difference between GTMs and machines with the oracle $0'$ for the prefix mode of description, and their similarity for the other modes. (It is worth noting that the prefix mode is not contained in the general scheme of defining complexities [21], but the prefix complexity based on the prefix mode is very convenient in certain proofs).

The main **open question** is whether weak GTM-complexities ($K^G_{\twoheadrightarrow}$, $K^G_{\rightarrowtail}$, and $-\log_2 P^G$) coincide on infinite sequences. A closely related (and probably, difficult) 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 form a surprising contrast to the famous result of Gács [11] 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. Eugene Asarin and Pieter Collins. Noisy Turing Machines. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *Proceedings of 32nd International Colloquium on Automata, Languages and Programming*, volume 3580 of *Lecture Notes in Computer Science*, pages 1031–1042. Springer, 2005.
2. Verónica Becher and Santiago Figueira. Kolmogorov Complexity for Possibly Infinite Computations. *J. of Logic, Lang. and Inf.*, 14(2):133–148, 2005.
3. Verónica Becher, Santiago Figueira, André Nies, and Silvana Picchi. Program Size Complexity for Possibly Infinite Computations. *Notre Dame J. Formal Logic*, 46(1):51–64, 2005.
4. M. S. Burgin. Inductive Turing Machines. *Notices of the Academy of Sciences of the USSR (translated from Russian)*, 270(6):1289–1293, 1991.
5. Cristian S. Calude and Boris Pavlov. Coins, Quantum Measurements, and Turing's Barrier. *Quantum Information Processing*, 1(1-2):107–127, 2002.
6. John Case, Sanjay Jain, and Arun Sharma. On Learning Limiting Programs. In *COLT '92: Proceedings of the fifth annual workshop on Computational Learning Theory*, pages 193–202, New York, NY, USA, 1992. ACM Press.
7. G. J. Chaitin. A Theory of Program Size Formally Identical to Information Theory. *Journal of the ACM*, 22:329–340, 1975.
8. Bruno Durand, Alexander Shen, and Nikolai Vereshchagin. Descriptive Complexity of Computable Sequences. *Theor. Comput. Sci.*, 271(1-2):47–58, 2002.
9. Gabor Etesi and Istvan Nemeti. Non-Turing Computations via Malament-Hogarth Space-Times. *International Journal of Theoretical Physics*, 41:341, 2002.
10. R. V. Freivald. Functions Computable in the Limit by Probabilistic Machines. In *Proceedings of the 3rd Symposium on Mathematical Foundations of Computer Science*, pages 77–87, London, UK, 1975. Springer-Verlag.
11. P. Gács. On the Relation between Descriptional Complexity and Algorithmic Probability. *Theoretical Computer Science*, 22:71–93, 1983.
12. E. M. Gold. Limiting Recursion. *Journal of Symbolic Logic*, 30(1):28–46, 1965.
13. Susumu Hayashi and Masahiro Nakata. Towards Limit Computable Mathematics. In *TYPES '00: Selected papers from the International Workshop on Types for Proofs and Programs*, pages 125–144, London, UK, 2002. Springer-Verlag.
14. A. N. Kolmogorov. Three Approaches to the Quantitative Definition of Information. *Problems of Information Transmission*, 1:1–11, 1965.
15. L. A. Levin. Laws of Information (Nongrowth) and Aspects of the Foundation of Probability Theory. *Problems of Information Transmission*, 10(3):206–210, 1974.
16. M. Li and P. M. B. Vitányi. *An Introduction to Kolmogorov Complexity and its Applications (2nd edition)*. Springer, 1997.
17. J. Poland. A Coding Theorem for Enumerating Output Machines. *Information Processing Letters*, 91(4):157–161, 2004.
18. 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.
19. Joseph R. Shoenfield. On Degrees of Unsolvability. *Annals of Math.*, 69:644–653, 1959.
20. S. G. Simpson. Degrees of Unsolvability: A Survey of Results. In J. Barwise, editor, *Handbook of Mathematical Logic*, pages 631–652. North-Holland, Amsterdam, 1977.
21. V. A. Uspensky and A. Shen. Relations between Varieties of Kolmogorov Complexities. *Math. Systems Theory*, 29:271–292, 1996.
22. V. A. Uspensky, N. K. Vereshchagin, and A. Shen. Lecture Notes on Kolmogorov Complexity. unpublished, http://lpcs.math.msu.su/∼ver/kolm-book.

Appendix

Proof of Lemma 1.
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 $\mathsf{0}'$-decidable, 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 the next t_0.
2. The machine $\tilde{T}$ is constructed almost trivially, by repeating essentially the proof of the Shoenfield limit lemma. First observe that each bit of the oracle $\mathsf{0}'$ is limit computable, and hence computable by a GTM. To compute the n-th output bit, $\tilde{T}$ emulates the running of $T^{\mathsf{0}'}(p, n)$, and in parallel computes all $\mathsf{0}'$ bits submitting their current values to the emulated computation of T. The emulation restarts every time when the value of a previously used oracle bit changes. If $T^{\mathsf{0}'}(p, n)$ halts, then the computation uses only a finite number of the oracle bits, thus eventually the emulation will be 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 ensures that $\tilde{T}(p) \rightarrowtail x$. To prove $\tilde{T}(p) \twoheadrightarrow x$, note that there exists an m such that on any input beginning with p, $T^{\mathsf{0}'}$ reads no 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 "shifted" through the quantifier over t_n. □

Proof of Lemma 2. First we eliminate one 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 $\mathsf{0}'$-computable, and thus $R(x, m)$ is $\mathsf{0}'$-enumerable from below. □

Proof of Theorem 1. By Lemma 1, $K^{\mathsf{0}'}(x) \stackrel{+}{=} K^G_{\twoheadrightarrow}(x)$.

Since $K^{\mathsf{0}'}(x) \stackrel{+}{=} -\log_2 m^{\mathsf{0}'}(x)$, the inequality (1) yields the statement if we prove $-\log m^{\mathsf{0}'}(x) \stackrel{+}{\le} -\log_2 P^G(x)$. 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 a "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 $\mathsf{0}'$-enumerable semimeasure on naturals and therefore it is less (up to a multiplicative constant) than $m^{\mathsf{0}'}(x)$. □

Proof of Theorem 2. First, $K^{\mathsf{0}'}(x) \overset{+}{\leq} K^G_{\rightrightarrows}(x)$ since $K^G_{\twoheadrightarrow}(x) \overset{+}{\leq} K^G_{\rightrightarrows}(x)$. On the other hand, knowing the length of the uniform weak description, GTM can stop reading the input tape and assume the remaining input bits 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 $K^{\mathsf{0}'}(x) \overset{+}{\leq} K^G_{\rightrightarrows}(x)$ follows from the fact that $K^G_{\rightrightarrows}(x) \overset{+}{\leq} K(x)$ and $K^{\mathsf{0}'}(x)$ is almost equal to $K(x)$ for $\mathsf{0}'$-random x (i. e. x of given length that have maximal $\mathsf{0}'$-complexity). The tightness of $K^G_{\rightrightarrows}(x) \overset{+}{\leq} K^{\mathsf{0}'}(x) + K(K^{\mathsf{0}'}(x))$ is proven as in Theorem 5 below. □

Proof of Theorem 3. By the second part of Lemma 1, a GTM can emulate a $\mathsf{0}'$-machine, therefore $M^{\mathsf{0}'}(x) = \lambda\{p \in \mathbb{B}^\infty \mid W^{\mathsf{0}'}(p) = x\} \leq C \cdot P^G(x)$.

To prove the other direction, let us define an auxiliary semimeasure on $x \in \mathbb{B}^* \cup \mathbb{B}^\infty$: $M^G(x) = \lambda\{p \in \mathbb{B}^\infty \mid \forall n \leq \ell(x)\, \exists t_n\, \forall t > t_n\, [U_t(p)]_n = x_n\}$. Clearly, for $x \in \mathbb{B}^\infty$ we have $P^G(x) = M^G(x) \leq \lim_{n\to\infty} M^G(x_{1:n})$ 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 binary sequences ($M^G(x) \geq M^G(x0) + M^G(x1)$), therefore, $M^G(x) \leq C \cdot M^{\mathsf{0}'}(x)$ for $x \in \mathbb{B}^*$. □

Proof of Theorem 5. Both bounds hold for the same reason as in Theorem 2. To prove the tightness of the upper bound, consider the set $A = \{p \in \mathbb{B}^* \mid U$ reads p and no more$\}$. Evidently, A is $\mathsf{0}'$-decidable and prefix-free. Put $A_n = \{p \in A \mid \ell(p) \leq n\}$. Since A is prefix-free, we have $|A_n| \leq 2^n/(n \log_2 n)$ for infinitely many n. Since A is $\mathsf{0}'$-decidable, there is a $\mathsf{0}'$-computable sequence of naturals n_k such that $|A_{n_k}| \leq 2^{n_k}/(n_k \log_2 n_k)$ and $n_k \geq 2^{k^2}$.

Now, using the diagonal construction, we can get $x \in \mathbb{B}^\infty$ such that $K^G_{\rightrightarrows}(x) > n_k$ and $Km^{\mathsf{0}'}(x) \leq n_k - \log_2 n_k + C$, where the constant C does not depend on k. Set $B_k = \{p \in \mathbb{B}^* \mid \ell(p) \leq n_k,\ \exists y \in \mathbb{B}^* \cup \mathbb{B}^\infty\, U(p) \rightrightarrows y\}$ and choose x such that $U(p) \not\rightrightarrows x$ for all $p \in B_k$. To define the n-th bit of x, we enumerate $p \in A_{n_k}$ such that the first n bits of $U(p)$ stabilize (possibly 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 lexicographically j-th of these p 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 $Km^{\mathsf{0}'}(x)$. The construction above gives an algorithm for computing x with the help of the oracle $\mathsf{0}'$ (to determine when bits stabilize) if n_k and $|B_k|$ are given, thus $Km^{\mathsf{0}'}(x) \leq K^{\mathsf{0}'}(\langle n_k, |B_k|\rangle) + O(1)$. The following is a prefix $\mathsf{0}'$-description of the pair $\langle n_k, |B_k|\rangle$: optimal prefix code of k followed by exactly l_k digits representing $|B_k|$ where $l_k = n_k - \log_2 n_k - \log_2 \log_2 n_k$. Since $|B_k| \leq |A_{n_k}| \leq 2^{n_k}/(n_k \log_2 n_k)$, we have enough codes for $|B_k|$. On the other hand, we can find n_k given k using the oracle $\mathsf{0}'$, and then compute l_k, therefore the encoding is prefix. Hence $K^{\mathsf{0}'}(\langle k, |B_k|\rangle) \overset{+}{\leq} K(k) + l_k \overset{+}{\leq} 2\log_2 k + n_k - \log_2 n_k - \log_2 \log_2 n_k \overset{+}{\leq} n_k - \log_2 n_k$. Since $n_k - \log_2 n_k$ is a monotonically increasing function of n_k, we have $Km^{\mathsf{0}'}(x) \overset{+}{\leq} K^G_{\rightrightarrows}(x) - \log_2 K^G_{\rightrightarrows}(x)$. Finally, $K^G_{\rightrightarrows}(x) \overset{+}{\geq} Km^{\mathsf{0}'}(x) + \log_2 K^G_{\rightrightarrows}(x) \overset{+}{\geq} Km^{\mathsf{0}'}(x) + \log_2 Km^{\mathsf{0}'}(x) \overset{+}{\geq} Km^{\mathsf{0}'}(x) + K(Km^{\mathsf{0}'}(x)) - 2\log_2 K(Km^{\mathsf{0}'}(x))$. □