Theoretical Computer Science 809 (2020) 531–537

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Descriptive complexity of computable sequences revisited

Nikolay Vereshchagin [1]

Moscow State University and HSE University, Russian Federation

ARTICLE INFO

Article history:
Received 9 September 2019
Received in revised form 4 January 2020
Accepted 13 January 2020
Available online 16 January 2020
Communicated by R. Giancarlo

Keywords:
Kolmogorov complexity
Limit complexity
Computable sequences

ABSTRACT

The purpose of this paper is to answer two questions left open in Durand et al. (2001) [2]. Namely, we consider the following two complexities of an infinite computable 0-1-sequence α: $C^{0'}(\alpha)$, defined as the minimal length of a program with oracle $0'$ that prints α, and $M_\infty(\alpha)$, defined as $\limsup C(\alpha_{1:n}|n)$, where $\alpha_{1:n}$ denotes the length-n prefix of α and $C(x|y)$ stands for conditional Kolmogorov complexity. We show that $C^{0'}(\alpha) \leqslant M_\infty(\alpha) + O(1)$ and $M_\infty(\alpha)$ is not bounded by any computable function of $C^{0'}(\alpha)$, even on the domain of computable sequences.

1. Introduction

The notion of Kolmogorov complexity for finite binary strings was introduced in the 60ies independently by Solomonoff, Kolmogorov and Chaitin [8,3,1]. There are different versions (plain Kolmogorov complexity, prefix complexity, etc. see [9] for the details) that differ from each other not more than by an additive term logarithmic in the length of the argument. In the sequel we are using plain Kolmogorov complexity $C(x)$ and conditional complexity $C(x|y)$, as defined in [3].

For infinite 0-1-sequences α, it is natural to define Kolmogorov complexity of a sequence α as the minimal length of a program to compute α_n from n:

$$C(\alpha) = \min\{l(p) \mid \forall n \in \mathbb{N}\ p(n) = \alpha_n\},$$

(and by definition $C(\alpha) = \infty$ if α is not computable). Here $l(p)$ denotes the length of the binary string p and $p(n)$ denotes the result of applying the program p to input n. As usual in Kolmogorov complexity theory, we assume that some optimal programming language U is fixed. That is, $(p, n) \mapsto U(p, n)$ is a computable function such that for any other computable function $V(p, n)$ there is a constant c such that for all p there is p' with $l(p') \leqslant l(p) + c$ and $U(p', n) = V(p, n)$ for all n. By $p(n)$ we then denote $U(p, n)$; conditional Kolmogorov complexity is defined as $C(x|n) = \min\{l(p) \mid p(n) = x\}$ and unconditional Kolmogorov complexity is defined as $C(x) = C(x|0)$. (For more details see [4,7].)

By definition $C(\alpha)$ is finite only if the sequence α is computable. Recall the following criteria of computability of α in terms of complexity of its finite prefixes: α is computable if and only if $C(\alpha_{1:n}|n) = O(1)$. Here $\alpha_{1:n}$ denotes the first n bits (= length-n prefix) of the sequence α. This result is attributed in [5] to A.R. Meyer (see also [4,7]). These criteria suggest another natural complexity measure of computable 0-1-sequences:

E-mail address: nikolay.vereshchagin@gmail.com.

[1] The article was prepared within the framework of the HSE University Basic Research Program. The author was in part funded by RFBR according to the research project N0 19-01-00563.

https://doi.org/10.1016/j.tcs.2020.01.013
0304-3975/© 2020 Elsevier B.V. All rights reserved.

$$\begin{tikzcd}
C^{0'}(\alpha) \arrow[d, thick] & C_\infty(\alpha) \arrow[l] \arrow[d, thick] & C(\alpha) \arrow[l, thick] \arrow[d, thick] \\
C^{0''}(\alpha) & M_\infty(\alpha) \arrow[l] & M(\alpha) \arrow[l, thick]
\end{tikzcd}$$

Fig. 1. Relations between different complexity measures for infinite sequences. Arrows go from the bigger quantity to the smaller one (up to $O(1)$-term, as usual). Bold arrows indicate inequalities that are immediate consequences of the definitions. Other arrows are provided by [2, Theorems 1 and 4].

$$M(\alpha) = \max_n C(\alpha_{1:n}|n) = \max_n \min_p \{l(p) \mid p(n) = \alpha_{1:n}\}.$$

What is the difference between $M(\alpha)$ and $C(\alpha)$? Up to an additive constant the latter is equal to

$$\min_p \max_n \{l(p) \mid p(n) = \alpha_{1:n}\}.$$

Thus the difference is in the order of operators max and min: $M(\alpha) \leqslant m$ means that for every n there is a program p_n of size at most m that computes $\alpha_{1:n}$ from n; this program may depend on n. On the other hand, $C(\alpha) \leqslant m$ means that there is a one such program that works for all n. Thus, $M(\alpha) \leqslant C(\alpha) + O(1)$ for all α.

One can consider also “almost all” versions of $C(\alpha)$ and $M(\alpha)$ defined in the following way:

$$C_\infty(\alpha) = \min\{l(p) \mid \forall^\infty n \; p(n) = \alpha_{1:n}\}$$
$$M_\infty(\alpha) = \limsup_n C(\alpha_{1:n}|n) = \min\{m \mid \forall^\infty n \exists p \; (l(p) \leqslant m \text{ and } p(n) = \alpha_{1:n})\}.$$

Here $\forall^\infty n$ stands for “for all but finitely many n”. Both these measures are finite only for computable sequences. Indeed, if $C_\infty(\alpha)$ or $M_\infty(\alpha)$ is finite, then $M(\alpha)$ is also finite, and the computability of α is implied by Meyer's theorem. All the four complexity measures mentioned above are “well calibrated” in the following sense: there are $\Theta(2^m)$ sequences whose complexity does not exceed m.

The measures $C(\alpha), M(\alpha), C_\infty(\alpha), M_\infty(\alpha)$ were introduced in [2], where the exact relations between them were established. The relations are shown on Fig. 1. On this picture $C^A(\alpha)$ denotes the relativized version of $C(\alpha)$ (programs have access to the oracle A).

The paper [2] shows that on this diagram no arrow could be inverted. More specifically, the following are true:

- $C(\alpha)$ cannot be bounded by any computable function of $M(\alpha)$ [2, Theorem 3];
- $M(\alpha)$ (and hence of $C(\alpha)$) cannot be bounded by any computable function of $C_\infty(\alpha)$ (and hence of $M_\infty(\alpha)$) [2, Theorem 2];
- $C_\infty(\alpha) \leqslant 2M_\infty(\alpha) + O(1)$ [2, Theorem 5], which is tight: for every m there is a sequence α with $C_\infty(\alpha) \geqslant 2m$ and $M(\alpha) \leqslant m + O(1)$ (and hence $M_\infty(\alpha) \leqslant m + O(1)$) [2, Theorem 6].
- $C^{0''}(\alpha)$ is finite while $C^{0'}(\alpha)$ is infinite for a sequence α that is $0''$-computable but not $0'$-computable.
- $C^{0'}(\alpha)$ and $C^{0''}(\alpha)$ are finite while $C_\infty(\alpha)$ and $M_\infty(\alpha)$ are infinite for a sequence that is $0'$-computable but not computable.

These results leave open the question about the relation between $C^{0'}(\alpha)$ and $M_\infty(\alpha)$: is it true that $C^{0'}(\alpha) \leqslant M_\infty(\alpha) + O(1)$. Another question left open in [2] is the following: are the inequalities

$$C_\infty(\alpha) \leqslant C^{0'}(\alpha) + O(1), \quad M_\infty(\alpha) \leqslant C^{0''}(\alpha) + O(1)$$

true on the domain of *computable* sequences? In this paper we answer the first question in the positive and the remaining two questions in the negative (Theorems 2 and 3 below). Thus we get the following diagram for complexities of computable sequences:

$$\begin{tikzcd}
 & C_\infty(\alpha) \arrow[d, "2"] & C(\alpha) \arrow[l, "\infty"'] \arrow[d, "\infty"] \\
C^{0'}(\alpha) & M_\infty(\alpha) \arrow[l, "\infty"'] & M(\alpha) \arrow[l, "\infty"']
\end{tikzcd}$$

The sign 2 near the arrow means that the larger quantity is at most 2 times the smaller quantity (up to an additive constant), and the sign ∞ means that the larger quantity cannot be bounded by any computable function of the smaller quantity even for computable sequences.

It is instructive to compare these results with similar results for finite sequences (i.e. strings). For $x \in \{0,1\}^*$ let $M_\infty(x) = \limsup C(x|n)$ and $C_\infty(x) = \min\{l(p) \mid p(n) = x \text{ for almost all } n\}$. From definitions it is straightforward that $M_\infty(x) \leqslant C_\infty(x) \leqslant C^{0'}(x)$ for all x (up to an additive constant). And by [10] we have

Theorem 1 *([10]).* $C^{0'}(x) \leqslant M_\infty(x) + O(1)$.

Hence all the three quantities coincide up to an additive constant. Similar inequality holds for infinite sequences as well (Theorem 2 from the present paper). However, the analog of the straightforward inequality $M_\infty(x) \leqslant C^{0'}(x) + O(1)$ is not true for infinite sequences, even on the domain of computable sequences.

2. Theorems and proofs

Theorem 2. $C^{0'}(\alpha) \leqslant M_\infty(\alpha) + O(1)$.

Proof. Fix k and consider the set S of all binary strings x with $C(x|l(x)) \leqslant k$. This set is computably enumerable uniformly on k. The *width* of S is less than 2^{k+1} (this means that for all n the set contains less than 2^{k+1} strings of length n).

We will view the set $\{0,1\}^*$ of all binary strings as a rooted tree. Its root is the empty string Λ and each edge connects a vertex x with its *children* $x0$ and $x1$.

An infinite path in S is an infinite set of vertices $\{x_0, x_1, x_2, \dots\}$ from S such that x_i is a child of x_{i-1} for all $i > 0$. Let us stress that we do not require infinite paths start in the root, that is, x_0 may be non-empty.

If $M_\infty(\alpha) \leqslant k$, then for some n prefixes of α of length at least n form an infinite path in S.[2] We have to show that in this case $C^{0'}(\alpha) \leqslant k + O(1)$.

The proof will follow from two lemmas. To state the lemmas we need yet another definition. A set T of strings is called *leafless*, if for all $x \in T$ at least one child of x is in T.

Lemma 1 *(on trimming leaves). For every computably enumerable set $S \subset \{0,1\}^*$ of width at most w there is a computably $0'$-decidable set $T \subset \{0,1\}^*$ such that*

(1) T is leafless,
(2) the width of T is at most w,
(3) T includes all infinite paths in S.

The program of the algorithm that $0'$-recognizes T can be found from w and the program enumerating S.

Lemma 2. *Let T be a leafless set of width at most w. Then for any infinite 0-1-sequence α whose sufficiently long prefixes form and infinite path in T we have $C^T(\alpha) \leqslant \log w + O(1)$. The constant $O(1)$ does not depend on T, α, w. Moreover, the program witnessing the inequality $C^T(\alpha) \leqslant \log w + O(1)$ needs only an oracle enumerating the set T (in any order).*

We first finish the proof of the theorem assuming the lemmas. By applying Lemma 1 to the set $S = \{x \mid C(x|l(x)) \leqslant k\}$ we obtain a leafless set T of width less than 2^{k+1} that includes all infinite paths in S and is $0'$-decidable uniformly on k. If $M_\infty(\alpha) \leqslant k$, then by the second lemma $C^T(\alpha) \leqslant k + O(1)$. Since T is $0'$-decidable uniformly on k, we have also $C^{0'}(\alpha|k) \leqslant k + O(1)$. Let p be a program of length at most $k+c$ (for some constant c) witnessing this inequality. Then given the string $p10^{k+c-l(p)}$ of length $k+c+1$ we can find both p and k (the latter can be found from its length $k+c+1$). Hence we can conclude that $C^{0'}(\alpha) \leqslant k + O(1)$.

It remains to prove the lemmas. We start with the proof of the simpler Lemma 2.

Proof of Lemma 2. Basically we have to number infinite paths in T in such a way that given the number of a path we can find all its vertices. We will imagine that we have *tokens* with numbers from 1 to w, and move those tokens along infinite paths in T. The number of an infinite path in T will be the number of the token that moves along that path.

More specifically, we start an enumeration of the set T. Observing strings enumerated in T we will place tokens on some of them; vertices baring tokens will be called *distinguished*. We will do that so that the following be true:

(1) distinguished vertices are pairwise inconsistent (neither of them is a prefix of another one),
(2) every string enumerated so far in T is a prefix of some distinguished vertex,
(3) tokens move only from a vertex to its descendant (=extension).

At the start no strings are enumerated so far and all tokens are not used. When a new string x is enumerated into T, we first look whether it is a prefix of a distinguished vertex. In that case we do nothing, since property (2) remains true.

[2] If, moreover, $M(\alpha) \leqslant k$, then that path starts in the root.

Otherwise property (2) has been violated. If x is an extension of a distinguished string y (such a vertex y is unique by property (1)), then we move the token from y to x keeping (1) and (3) true and restoring (2).

Finally, if x is inconsistent with all distinguished nodes, we take a new token and place it on x restoring (2) and keeping (1).

Since T is leafless and its width is at most w, the set T cannot have more than w pairwise inconsistent strings (for all large enough n each of those strings has a length-n extension in T and those extensions are pairwise different). Therefore we do not need more than w tokens.

By construction for every infinite path in T a token is at a certain time placed on a vertex of the path and moves along the path infinitely often.

To every natural number i from 1 to w we assign a program p_i that for input n waits until the ith token is placed on a string x of length at least n, then it prints the first n bits of x. □

It remains to prove the first lemma. We present two its proofs: the direct original proof and the proof suggested by an anonymous referee, which is based on the low basis theorem. Both proofs use Cantor topology on the set of subsets of $\{0,1\}^*$. Its base consists of sets of the form:

$$\Omega_{A,B} = \{X \subset \{0,1\}^* \mid A \subset X,\ B \cap X = \emptyset\}, \tag{1}$$

where A, B are any *finite* subsets of $\{0,1\}^*$. By definition, open sets in Cantor topology are arbitrary unions of these sets. It is well known that this topological space is compact. We start with the direct proof.

A direct proof of Lemma 1. We will consider leafless sets T such that the width of the set $T \cup S$ does not exceed w. Such sets will be called *acceptable*. For instance, the empty set is acceptable. The key observation is the following: *the family of acceptable sets is closed in Cantor topology*.

The set T is defined as the largest acceptable set with respect to some linear order. More specifically, consider the lexicographical order on binary strings (for strings of different length, the shorter string is less than the longer one). Then we define $X < Y$ for different sets $X, Y \subset \{0,1\}^*$ if the lex first string in the symmetric difference of X, Y belongs to $Y \setminus X$ (in other words, we compare sets according to the lexicographical order on their characteristic sequences). Not every non-empty family of subsets of $\{0,1\}^*$ has the largest set with respect to this order. However, this holds for *closed* families. Hence there exists the largest acceptable set T.

In other words, one can define T recursively: enumerate all binary strings $x_1, x_2, \dots$ according to the lexicographical order, then put x_i in T if there is an acceptable set R which includes the set $T \cap \{x_1, \dots, x_{i-1}\}$, or, equivalently,

$$R \cap \{x_1, \dots, x_{i-1}\} = T \cap \{x_1, \dots, x_{i-1}\}.$$

This definition guarantees that for all i there is an acceptable set R with $R \cap \{x_1, \dots, x_{i-1}\} = T \cap \{x_1, \dots, x_{i-1}\}$. Since the family of acceptable sets is closed, this implies acceptability of T. And by construction this T is larger than or equal to every acceptable set.

Properties (1) and (2) hold automatically for T. Let us verify the property (3). Let α be an infinite path in S. Consider the set $T' = T \cup \alpha$. It is leafless (since every vertex from α has a child in α). Besides, $T' \cup S = T \cup S$ (as $\alpha \subset S$), hence T' is acceptable. The definition of T implies that it is a maximal acceptable set (w.r.t. inclusion). Therefore $T' = T$, or, in other words, $\alpha \subset T$.

It remains to show that T is $0'$-decidable. Assume that we already know for every string among $x_1, \dots, x_{i-1}$ whether it belongs to T or not. We have to decide whether $x_i \in T$. By construction x_i is in T if and only if there is an acceptable set including the set $T \cap \{x_1, \dots, x_{i-1}\}$ and x_i. Thus it suffices to prove that for any finite $E \subset \{0,1\}^*$ we can decide with the help of $0'$ whether there is an acceptable set including E or not. To this end we reformulate this property of E. Fix a computable enumeration of S and denote by S_j the subset of S consisting of all strings enumerated in j steps.

Call a set R *acceptable at time* j if it is leafless and the width of $R \cup S_j$ is at most w. We claim that

there is an acceptable set including E if and only if for all j there is a set R_j including E that is acceptable at time j.

Since acceptability implies acceptability at time j for all j, one direction is straightforward. In the other direction: assume that for every j there is a set $R_j \supset E$ which is acceptable at time j. We have to construct an acceptable set $R \supset E$.

By compactness arguments, the sequence $R_1, R_2, \dots$ has an accumulation point R. Since both properties "to include E" and "be leafless" are closed, the set R possesses these properties. It remains to show that the width of the set $R \cup S$ is at most w. For the sake of contradiction assume that there are $w+1$ strings of the same length n that belong to $R \cup S$. Then consider the (open) family that consists of all sets R' such that the set $R' \cup S$ includes all those strings. Since R is in this family, for infinitely many j the set R_j is in this family, that is, $R_j \cup S$ includes all those strings. Among those j's, for almost all j the set $R_j \cup S_j$ includes all those strings. We obtain a contradiction, as the width of all the sets $R_j \cup S_j$ is at most w.

It remains to show decidability of the following property of the pair (E, j): *there is a set R including E that is acceptable at time j* (indeed, in this case the oracle $0'$ is able to decide whether this property holds for all j). Indeed, the sets S_j and

E are finite. Let n be the maximal length of strings from these sets. Without loss of generality we may assume that each string $x \in R$ of length n or larger has exactly one child in R, namely, $x0$, and all strings from R of length larger than n are obtained from strings of length n from R by appending zeros. Such sets R are essentially finite objects and there are finitely many of them. For any such set we can decide whether it includes E and is acceptable at time j. The lemma is proved. □

Remark 1. The set T constructed in the proof of Lemma 2 can be defined in three equivalent ways. (1) T is the largest acceptable set according to the lexicographical order on characteristic sequences (this definition was used in the proof). (2) A recursive definition: include x_i in T if for all j there is a set that is acceptable at time j and includes $T \cap \{x_1, \dots, x_{i-1}\}$ and x_i (this definition was suggested by B. Bauwens). In the proof of Lemma 1, we showed that definitions (1) and (2) are equivalent. (3) T is the limit of the sequence $R_1, R_2, \dots$ where R_j is the largest set that is acceptable at time j (one can show that this sequence converges indeed). This definition was used in the original proof of Lemma 1.

Both from definitions (2) and (3) it easily follows that T is $0'$-decidable. On the other hand, from definition (1) it easily follows that T includes all infinite paths in S. We have chosen definition (1) as the main definition, since it is independent on the enumeration of the set S.

Proof of Lemma 1 using the low basis theorem. It seems natural to let T be the union of all infinite paths in S. In this case the conditions (1)–(3) hold automatically. However, this set is only Π_2, since

$$T = \{x \mid \forall i \text{ there is an extension of } x \text{ of length } l(x) + i \text{ in } S\}$$

(the König's lemma). It is not hard to find an example of a c.e. set S for which this set T is Π_2 complete (and hence is not $0'$-decidable).

However we can overcome this obstacle using the low basis theorem (see, e.g., [6]). To state the theorem we need the notion of an effectively closed family. A family U of subsets of $\{0, 1\}^*$ is called *effectively open* if U can be represented in the form

$$\bigcup_{(A,B)\in C} \Omega_{A,B} \tag{2}$$

where C is a computably enumerable set of pairs of finite subsets of $\{0, 1\}^*$, and $\Omega_{A,B}$ is defined by Equation (1). Complements of effectively open families are called *effectively closed.*

The low basis theorem: Let V be a non-empty effectively closed family of subsets of $\{0, 1\}^*$. Then there exists a set $D \in V$ which is low, i.e., $D' = 0'$. The program with oracle $0'$ that recognizes D' can be found from any program that enumerates any set C such that the complement of V equals (2).

We apply this theorem to the family of all sets $D \subset \{0, 1\}^*$ with the following properties: (1) $S \subset D$ and (2) the width of D is at most w. Since S is enumerable, this family is effectively closed. The set S is in this family, hence the family is non-empty. By the low basis theorem we conclude that there is a low set D in the family.

Consider now instead of the above set T the following set

$$\tilde{T} = \{x \mid \forall i \text{ there is an extension } y \text{ of } x \text{ of length } l(x) + i \text{ in } D\}.$$

The property (1) of D guarantees that $T \subset \tilde{T}$ and hence $\tilde{T}$ includes all infinite paths in S. The property (2) of D guarantees that the width of $\tilde{T}$ is at most w. And the set $\tilde{T}$ is obviously D'-decidable, which implies $0'$-decidability since D is low. □

Now we know that $C^{0'}(\alpha) \leqslant M_\infty(\alpha) + O(1)$. How large can the gap between $C^{0'}(\alpha)$ and $M_\infty(\alpha)$ be? For α equal to the characteristic sequence of $0'$ the gap is infinite, since $C^{0'}(0')$ is finite while $M_\infty(0')$ is infinite. However, we are mostly interested in computable sequences, thus we refine the question: How large can the gap between $C^{0'}(\alpha)$ and $M_\infty(\alpha)$ be for *computable* sequences α?

It turns out that this such gap can be arbitrary large: $M_\infty(\alpha)$ cannot be bounded by any computable function of $C^{0'}(\alpha)$. More specifically, the following holds:

Theorem 3. *For any computable function $f : \mathbb{N} \to \mathbb{N}$ for all m there is a* computable *sequence α for which $C^{0'}(\alpha) \leqslant m + O(1)$ while $M_\infty(\alpha) \geqslant f(m)$. The constant $O(1)$ depends on the function f.*

We present two proofs of Theorem 3. The first one was suggested by an anonymous referee and uses Theorem 1. The second one is the original one, it is direct.

The first proof. Let $\alpha = u1000\dots$ where the string u will be chosen later. Notice that we can find u from all sufficiently long prefixes of α (trim the suffix $1000\dots0$) and the other way around: from u we can find all prefixes of α. It follows that $M_\infty(\alpha)$ equals $M_\infty(u)$ (up to and additive constant term). And the latter equals $C^{0'}(u)$ by Theorem 1.

Thus we have to find for each m a string u such that $C^{0'}(u) \geqslant f(m)$ while $C^{0'}(u1000\dots) \leqslant m + O(1)$ (for every m). To this end consider all programs of length at most $f(m)$ that have access to the oracle $0'$. Let

$$u = 0^{t_1}10^{t_2-t_1}1\dots10^{t_k-t_{k-1}}$$

where $t_1 \leqslant t_2 \leqslant \dots \leqslant t_k$ is the sorted sequence of running times of all those programs that halt (thus k is the number of halting programs). The key point is that the sequence $u1000\dots$ is $0'$-computable from k by the following program:

```
for $t = 1, 2, \dots$:
  print 0;
  for all strings $p$ of length at most $f(m)$:
    run $p$ within $t$ steps,
    if $p$ halts exactly in $t$ steps,
      then print 1;
    endif;
  endfor;
endfor.
```

Hence $C^{0'}(u1000\dots) \leqslant \log m + O(1) \leqslant m + O(1)$. On the other hand, from u and m we can find the list of all halting programs (with oracle $0'$) of length at most $f(m)$. This list has $0'$-relativized complexity at least $f(m) - O(\log m)$ (from m and that list we can find the first string x with $C^{0'}(x) > f(m)$). Hence $C^{0'}(u) \geqslant f(m) - O(\log m)$.

We obtained a weaker lower bound for $C^{0'}(\alpha)$ than the required one. However, we can apply the above arguments to a larger function, say to $f'(m) = f(m) + m$. Thus for each m there is a sequence α with $C^{0'}(\alpha) \leqslant m + O(1)$ and $M_\infty(\alpha) \geqslant f(m) + m - O(\log m)$. If m is large enough, then the right hand side of this inequality is at least $f(m)$. And for remaining finite number of exceptions we can construct α by applying the above arguments for $f'(m) = m$ and a single large enough m. □

The second proof. We will use the Game Approach. Assume that a natural parameter w is fixed. Consider the following game between two players, Alice and Bob. Players turn to move alternate. On each move each player can color any string or do nothing. We will imagine that Alice uses green color and Bob uses red color (each string can be colored in both colors). For every n Alice may color at most w strings of length n. The players make infinitely many moves and then the game ends. Alice wins if (1) for some n there are w strings of length n who all have been colored by both players (w red-green strings of the same length), or (2) there is an infinite 0-1-sequence α such that $\alpha_{1:n}$ is the lex first green string of length n for all n and $\alpha_{1:n}$ has not been colored by Bob (is not red) for infinitely many n.

From the rules of the game it is clear that it does not matter who starts the game (postponing a move does not hurt).

Lemma 3. *For every w Alice has a winning strategy in this game and this strategy is computable uniformly on w.*

Proof. Alice's strategy is recursive. If $w = 1$, then Alice just colors the strings $\Lambda, 0, 00, 000, \dots$ (on her ith move she colors the string 0^i).

Assume now that we have already defined Alice's winning strategy in the w-game, we will call it the *w-strategy*. Then Alice can win $(w+1)$-game as follows: she colors first the empty string and then runs the w-strategy in the subtree with the root 1.[3] If w-strategy wins in the first way (that is, for some n there are w red-green strings of length n that start with 1), then Alice stops w-strategy. She then colors the strings $0, 00, 000, \dots, 0^m$ where m is larger than the length of all strings colored by w-strategy. Then Alice runs w-strategy for the second time, but this time in the subtree with the root 0^m1. Again, if the second run of w-strategy wins in the first way, then Alice stops it and colors the strings $0^{m+1}, 0^{m+2}, \dots, 0^l$ where l is larger than the length of all strings colored green so far. And so on.

The $w+1$-strategy is constructed. Let us show that it obeys the rules, that is, for all n it colors at most $w+1$ strings of length n. Indeed, for each n at most w strings of length n were colored by a run of w-strategy (different runs of w-strategy color strings of different lengths) and besides the string 0^n might be colored.

Let us show that $w+1$-strategy wins the game. We will distinguish two cases.

Case 1. We have run w-strategy infinitely many times. Then each its run has won in the first way. Hence for infinitely many n there exist w red-green strings of length n and all those strings have a 1 (indeed, we have run w-strategies only in subtrees with roots of the form $00\dots01$). Consider now the strings 0^n for those n's. If at least one of them has been colored by Bob, then we have won in the first way. Otherwise the nodes $\Lambda, 0, 00, \dots$ are lex first green strings of lengths $0, 1, 2, \dots$ and infinitely many of them are not red. This means that we have won in the second way.

Case 2. A run of w-strategy, say in the subtree with root 0^l1, has not been stopped and hence it won in the second way. Then the string 0^l1 has been extended by an infinite green path P (including the string 0^l1 itself) that contains infinitely many non-red nodes. Since, all the nodes $0, 00, 000, \dots, 0^l$ are also green, the path 0^l1P consists entirely of green nodes, starts in the root, contains infinitely many non-red nodes and all its nodes are lex first green nodes (recall that all strings with prefix 0^l0 have not been colored by Alice). □

[3] Formally, that means that Alice adds prefix 1 to every move made by the w-strategy, postpones Bob's moves that do not start with 1, and for every Bob's move of the form $1x$ tells the w-strategy that Bob has made the move x.

To prove the theorem we apply $2^{f(m)}$-strategy against the following “blind” Bob’s strategy: Bob colors a string x of length n when he finds a program p of length less than $f(m)$ with $p(n) = x$ (he runs all programs of length less than $f(m)$ on all inputs in a dovetailing style). This strategy is computable and for all n it colors less than $2^{f(m)}$ strings of length n. Hence Alice wins in the second way: there is an infinite green path whose infinitely many nodes are not red. Call this path α. By construction we have $M_\infty(\alpha) \geqslant f(m)$.

On the other hand, the set of all green nodes is computably enumerable and its width is at most $f(m)$. Hence $M(\alpha) < f(m) + O(1)$ and by Meyer’s theorem α is computable.

Finally, the sequence α can be computed from m with oracle $0'$: for every n we can find the lex first green string of length n. Hence $C^{0'}(\alpha) < \log m + O(1) < m + O(1)$. The theorem is proved. $\square$

Remark 2. Notice that in Theorem 3 we can weaken the assumption about the function f by assuming that f is only $0'$-computable. Indeed, in the first proof the program computing the sequence $u1000\ldots$ from m has access to oracle $0'$ thus it may compute $u1000\ldots$ even if f is only $0'$-computable. In the second proof, to compute α from m we first compute $f(m)$ from m using $0'$ and then again use $0'$ to find the lex first green string of every length n.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The author is sincerely grateful to Bruno Bauwens and to anonymous referees for providing alternative proofs of Theorems 2 and 3 and for Remark 2.

References

[1] G.J. Chaitin, On the length of programs for computing finite binary sequences: statistical considerations, J. ACM 16 (1969) 145–159.
[2] B. Durand, A. Shen, N. Vereshchagin, Descriptive complexity of computable sequences, Theor. Comput. Sci. 171 (2001) 47–58.
[3] A.N. Kolmogorov, Three approaches to the quantitative definition of information, Probl. Inf. Transm. 1 (1) (1965) 1–7.
[4] M. Li, P. Vitányi, An Introduction to Kolmogorov Complexity and Its Applications, second edition, Springer Verlag, 1997.
[5] D.W. Loveland, A variant of Kolmogorov concept of complexity, Inf. Control 15 (1969) 510–526.
[6] P. Odifreddi, Classical Recursion Theory, North-Holland, 1989.
[7] A. Shen, V.A. Uspensky, N. Vereshchagin, Kolmogorov Complexity and Algorithmic Randomness, American Mathematical Society, 2017.
[8] R.J. Solomonoff, A formal theory of inductive inference, part 1 and part 2, Inf. Control 7 (1–22) (1964) 224–254.
[9] V.A. Uspensky, A.Kh. Shen, Relations between varieties of Kolmogorov complexities, Math. Syst. Theory 29 (1996) 271–292.
[10] N. Vereshchagin, Kolmogorov complexity conditional to large integers, Theor. Comput. Sci. 271 (2002) 59–67.