ELSEVIER
Theoretical Computer Science 271 (2002) 47–58
Theoretical Computer Science
www.elsevier.com/locate/tcs

# Descriptive complexity of computable sequences

Bruno Durand[a,*], Alexander Shen[b], Nikolai Vereshchagin[c,1]

[a] *CMI-LIM, Technopôle de Château-Gombert, Universite de Provence 39 rue Joliot-Curie, F-13453 Marseille Cedex 13, France*

[b] *Institute of Problems of Information Transmission, Moscow, Russia*

[c] *Department of Mathematical Logic and Theory of Algorithms, Moscow State University, Vorobjevy Gory, Moscow 119899, Russia*

**Abstract**

Our goal is to study the complexity of infinite binary recursive sequences. We introduce several measures of the quantity of information they contain. Some measures are based on size of programs that generate the sequence, the others are based on the Kolmogorov complexity of its finite prefixes. The relations between these complexity measures are established. The most surprising among them are obtained using a specific two-players game[2]. 

## 1. Introduction

The notion of Kolmogorov entropy (= complexity) for finite binary strings was introduced in the 1960s independently by Solomonoff [8], Kolmogorov [5] and Chaitin [1]. There are different versions (plain Kolmogorov entropy, prefix entropy, 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 entropy $K(x|y)$ as defined in [5], but similar results can be obtained for prefix complexity.

When an infinite 0–1-sequence is given, we may study the entropy (= complexity) of its finite prefixes. If prefixes have high complexity, the sequence is random (see [6] for details and references); if prefixes have low complexity, the sequence is computable. In the sequel, we study the latter type.

Let $K(x)$, $K(x|y)$ denote the plain Kolmogorov entropy (complexity) of a binary string $x$ and the conditional Kolmogorov entropy (complexity) of $x$ when $y$ (some other

* Corresponding author.

*E-mail addresses:* bruno.durand@gyptis.univ-mrs.fr, bdurand@cmi.univ-mrs.fr (B. Durand), shen@mccme.ru (A. Shen), ver@mech.math.msu.su (N. Vereshchagin).

[1] The work was done while visiting LIP, Ecole Normale Supérieure of Lyon.

[2] This paper is the extended version of [4].

0304-3975/02/$ - see front matter 
PII: S0304-3975(01)00030-5

binary string) is known. That is, $K(x)$ is the length of the shortest program $p$ that prints $x$; $K(x|y)$ is the length of the shortest program that prints $x$ given $y$ as input. (For details see [6] or [10].)

Let $\omega_{1:n}$ denote first $n$ bits (= $n$-prefix) of the sequence $\omega$.

Let us recall the following criteria of computability of $\omega$ in terms of entropy of its finite prefixes:

(a) $\omega$ is computable if and only if $K(\omega_{1:n}|n) = \mathcal{O}(1)$. This result is attributed in [7] to A.R. Meyer (see also [10, 6]).

(b) $\omega$ is computable if and only if $K(\omega_{1:n}) \leqslant K(n) + \mathcal{O}(1)$ [2].

(c) $\omega$ is computable if and only if $K(\omega_{1:n}) \leqslant \log_2 n + \mathcal{O}(1)$ [2].

These results provide criteria of the computability of infinite sequences. For example, (a) can be reformulated as follows: sequence $\omega$ is computable if and only if $M(\omega)$ is finite, where

$$M(\omega) = \max_n K(\omega_{1:n}|n) = \max_n \min_p \{l(p) \mid p(n) = \omega_{1:n}\}$$

($l(p)$ stands for the length of program $p$; $p(n)$ denotes its output on $n$).

Therefore, $M(\omega)$ can be considered as a complexity measure for $\omega$: $M(\omega)$ is finite iff $\omega$ is computable.

Another straightforward approach is to define entropy (complexity) of a sequence $\omega$ as the length of the shortest program computing $\omega$:

$$K(\omega) = \min\{l(p) \mid \forall n \;\; p(n) = \omega_{1:n}\}$$

(and by definition $K(\omega) = \infty$ if $\omega$ is not computable).

The difference between $K(\omega)$ and $M(\omega)$ can be explained as follows: $M(\omega) \leqslant m$ means that for every $n$ there is a program $p_n$ of size at most $m$ that computes $\omega_{1:n}$ given $n$; this program may depend on $n$. On the other hand, $K(\omega) \leqslant m$ means that there is one such program that works for all $n$. Thus, $M(\omega) \leqslant K(\omega)$ for all $\omega$, and one can expect that $M(\omega)$ may be significantly less than $K(\omega)$. (Note that the known proofs of (a) give no bounds of $K(\omega)$ in terms of $M(\omega)$.)

Indeed, Theorem 3 shows that there is no computable bound for $K(\omega)$ in terms of $M(\omega)$: for any computable function $\alpha(m)$ there exist computable infinite sequences $\omega^0, \omega^1, \omega^2, \ldots$ such that $M(\omega^m) \leqslant m + \mathcal{O}(1)$ and $K(\omega^m) \geqslant \alpha(m) - \mathcal{O}(1)$.

The situation changes surprisingly when we compare "almost all" versions of $K(\omega)$ and $M(\omega)$ defined in the following way:

$$K_\infty(\omega) = \min\{l(p) \mid \forall^\infty n \;\; p(n) = \omega_{1:n}\},$$

$$M_\infty(\omega) = \limsup_n K(\omega_{1:n}|n) = \min\{m \mid \forall^\infty n \;\exists p \;\; (l(p) \leqslant m \text{ and } \; p(n) = \omega_{1:n})\}$$

($\forall^\infty n$ stands for "for all but finitely many $n$"). It is easy to see that $M_\infty(\omega)$ is finite only for computable sequences. Indeed, if $M_\infty(\omega)$ is finite, then $M(\omega)$ is also finite, and the computability of $\omega$ is implied by Meyer's theorem.

Surprisingly, it turns out that $K_\infty(\omega) \leqslant 2M_\infty(\omega) + \mathcal{O}(1)$ (Theorem 5) so the difference between $K_\infty$ and $M_\infty$ is not so large as between $K$ and $M$. We stress that this result is

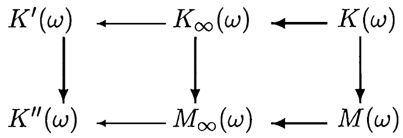


Fig. 1. Relations between different complexity measures for infinite sequences.

rather strange because a multiplicative constant 2 appears, and has no intuitive meaning taking into account that all the six complexity measures ("entropies") mentioned above are "well calibrated" in the following sense: there are $\Theta(2^m)$ sequences whose entropy does not exceed $m$. In the general theory of Kolmogorov complexity, additive constants often appear, but not multiplicative ones. As Theorem 6 shows, *this bound is tight*.

It is interesting also to compare $K_\infty$ and $M_\infty$ with $K$ and $M$, as well as with relativized versions of $K$. For any oracle $A$ one may consider a relativized Kolmogorov complexity $K^A$ allowing programs to access the oracle. Then $K^A(\omega)$ is defined in a natural way. By $K'(\omega)$ [or $K''(\omega)$] we mean $K^A(\omega)$ where $A = \mathbf{0}'$ [or $\mathbf{0}''$]. The results of this comparison are shown in the diagram of Fig. 1.

Arrows go from the bigger quantity to the smaller one (up to $\mathcal{O}(1)$-term, as usual). Bold arrows indicate inequalities that are immediate consequences of the definitions. Other arrows are provided by Theorem 1 ($K'(\omega) \leqslant K_\infty(\omega) + \mathcal{O}(1)$) and Theorem 4 ($K''(\omega) \leqslant M_\infty(\omega) + \mathcal{O}(1)$).

As we have said, $K_\infty(\omega) \leqslant 2M_\infty(\omega) + \mathcal{O}(1)$, so $K_\infty$ and $M_\infty$ differ only by a bounded factor. If we ignore such a difference, we get a simplified diagram

$$K''(\omega) \leftarrow K'(\omega) \leftarrow K_\infty(\omega), \quad M_\infty(\omega) \leftarrow M(\omega) \leftarrow K(\omega),$$

where $X \leftarrow Y$ means that $X = \mathcal{O}(Y)$.

In the latter diagram no arrow could be inverted. Indeed, $K''(\omega)$ is finite while $K'(\omega)$ is infinite for a sequence $\omega$ that is $\mathbf{0}''$-computable but not $\mathbf{0}'$-computable. Therefore the first arrow cannot be inverted. The second one cannot be inverted for similar reasons: $K'(\omega)$ is finite while $K_\infty(\omega)$ and $M_\infty(\omega)$ are infinite for a sequence that is $\mathbf{0}'$-computable but not computable. Theorem 2 shows that $K_\infty(\omega)$ and $M_\infty(\omega)$ could be small while $M(\omega)$ is large. Finally, Theorem 3 shows that $M(\omega)$ could be small while $K(\omega)$ is large.

These diagrams and the statements we made about them do not tell us whether the inequalities $K_\infty(\omega) \leqslant M(\omega) + \mathcal{O}(1)$ and $K'(\omega) \leqslant M_\infty(\omega) + \mathcal{O}(1)$ are true. The first one is not true, as Theorem 6 implies. We do not know whether the second one is true.

Other open questions: (1) is it possible to reverse the second arrow ($K'(\omega) \leftarrow K_\infty(f), M_\infty(f)$) for *computable* sequences? (2) what can be said about similar notions for finite strings? in particular, is $\limsup_n K(x|n)$ equal to $K'(x) + \mathcal{O}(1)$ or not? [3]

[3] It was shown recently by the third author that $\limsup_n K(x|n) = K'(x) + \mathcal{O}(1)$. See the paper "Kolmogorov Complexity Conditional to Large Integers" in this volume.

## 2. Theorems and proofs

**Theorem 1.** $K'(\omega) \leqslant K_\infty(\omega) + \mathcal{O}(1)$.

**Proof.** Let $p(n) = \omega_{1:n}$ for almost all $n$. The following program $q$ (with access to $0'$) computes $\omega_{1:n}$ given $n$: For $k = n, n+1, \ldots$ find out (using $0'$) whether (a) $p(k)$ is defined and is a binary string of length $k$; (b) $p(m)$ is consistent with $p(k)$ for all $m > k$; consistency means that either [$p(m)$ has length $m$ and has prefix $p(k)$] or [$p(m)$ is undefined]. As soon as $k$ satisfying both (a) and (b) is found, print the first $n$ bits of $p(k)$.

Obviously, $q(n) = \omega_{1:n}$ for all $n$ and the bit length of $q$ is $\mathcal{O}(1)$ longer than that of $p$. $\square$

**Theorem 2.** *For any computable function* $\alpha(m)$ *there exist infinite sequences* $\omega^0, \omega^1, \ldots$ *such that* $M(\omega^m) \geqslant \alpha(m)$ *while* $K_\infty(\omega^m) \leqslant m + \mathcal{O}(1)$.

**Proof.** Let $x_m$ be the lexicographically first string $x$ of length $\alpha(m)$ such that $K(x|\alpha(m)) \geqslant \alpha(m)$. (Such a string exists since the number of programs of length less than $k$ is less than $2^k$.)

Now let $\omega^m = x_m 0000\ldots$ . By definition, $M(\omega^m) \geqslant K(x_m|\alpha(m)) \geqslant \alpha(m)$. On the other hand, $K_\infty(\omega^m) \leqslant m + \mathcal{O}(1)$. Indeed, the set $\{x \mid K(x|l(x)) < l(x)\}$ is enumerable. Consider the program $p_m$ that having input $n$ performs $n$ steps of enumeration of this set. Then the program $p_m$ finds the first string $x_m^n$ of length $m$ that was not encountered, and outputs first $n$ bits of the sequence $x_m^n 0000\ldots$ . If $n$ is large enough then $x_m^n = x_m$ and $p$ outputs $\omega^m_{1:n}$. It remains to note that the length of $p_m$ is $\log m + \mathcal{O}(1)$. $\square$

**Theorem 3.** *For any computable function* $\alpha(m)$ *there exist infinite sequences* $\omega^0, \omega^1, \ldots$ *such that* $K(\omega^m) \geqslant \alpha(m)$ *while* $M(\omega^m) \leqslant m + \mathcal{O}(1)$.

**Proof.** Let $c$ be a constant (to be specified later). The set $E = \{\langle x, k\rangle \mid K(x) < \alpha(k) + c\}$ is enumerable. Consider the process of its enumeration. Let $s(m)$ be the time (step number) when all pairs of type $\langle x, m\rangle$ with a given $m$ have been appeared in $E$. Now let $\omega^m = 0^{s(m)}1111\ldots$ .

Let us prove that $K(\omega^m) > \alpha(m) - \mathcal{O}(1)$. Assume that $p(n) = \omega^m_{1:n}$ for all $n$. Given $p$ we can find the first 1 in $\omega^m$ and hence $s(m)$. Thus $K(s(m)) \leqslant K(\omega^m) + \mathcal{O}(1)$. On the other hand, given $s(m)$ we can find the (lexicographically) first string $x_m$ of entropy $\alpha(m)$ or more, therefore, $\alpha(m) \leqslant K(x_m) \leqslant K(s(m)) + \mathcal{O}(1)$. Hence $\alpha(m) \leqslant K(\omega^m) + \mathcal{O}(1)$.

Let us prove now that $M(\omega^m) \leqslant m + \mathcal{O}(1)$. Let the program $q$ on input $n$ output $n$ zeros. Then $q(n) = \omega^m_{1:n}$ for all $n \leqslant s(m)$.

Consider the program $p_m$ that on input $n$ does $n$ steps of enumeration of the set $E$, finds the number $s(m,n)$ of the last step among them when a new pair of type $\langle x, m\rangle$ with a given $m$ has been appeared, and then outputs the first $n$ bits of the sequence $0^{s(m,n)}1111111\ldots$ . If $n \geqslant s(m)$, then $p_m$ outputs the correct prefix of $\omega^m$.

Thus, for any $n$, either $p_m$ or $q$ (given $n$) outputs $\omega^m_{1:n}$. It remains to note that the length of $p_m$ is $\log m + \mathcal{O}(1)$. □

Theorems 2 and 3 can be reinforced using a technique presented in [3]: they are true for any computable infinite family of distinct sequences $\omega^0, \omega^1, \ldots$ (the family itself should be computable). Anyway these *pathological* cases are rare: the difference between $K(x)$ and $K''(x)$ can be huge but this concerns only an exponentially small portion of strings $x$ of a given size.

**Theorem 4.**

$$K''(\omega) \leqslant M_\infty(\omega) + \mathcal{O}(1).$$

**Proof.** Let $m = M_\infty(\omega) + 1$. Consider the set $T = \{x \mid K(x|l(x)) < m\}$. By definition, all sufficiently long prefixes of $\omega$ belong to $T$. The set $T$ is enumerable. For each $n$ there are at most $2^m$ strings of length $n$ in $T$. A string $x \in T$ is called "good" if there is a sequence $\xi$ such that $x$ is a prefix of $\xi$ and all prefixes of $\xi$ longer than $x$ belong to $T$ (in other words, if $x$ lies on the infinite path in $T$). It is easy to see that König's lemma allows to express the statement "$x$ is good" as $\forall\exists$-statement. Therefore, the set $\bar{T}$ of all good strings is $\mathbf{0}''$-decidable.

Now assign to every string in $\bar{T}$ a number as follows. Consider all the strings in $\bar{T}$ in order of increasing length, and the strings of the same length in the lexicographical order. Assume that all the strings preceeding the current string $u$ have been already assigned a number. If the father of $u$ is not in $\bar{T}$ then assign to $u$ the first unused number. Assume that the father $v$ of $u$ is in $\bar{T}$ and hence is already assigned a number. If both sons of $v$ are in $\bar{T}$ then the left son of $v$ is assigned the same number as $v$ and the right son is assigned the first unused number. Otherwise $u$ is assigned the same number as its father. The number of used numbers does not exceed $2^m$. Obviously, all but finitely nodes in any infinite path in $\bar{T}$ have been assigned the same number. This number is considered as the number of the path. There is an $\mathbf{0}''$-algorithm that gives $k$-bit prefix of path number $i$ for given $k$ and $i$. Appending $i$ (considered as $m$-bit string) to that algorithm, we get a $\mathbf{0}''$-program that gives $k$-bit prefixes of $i$th path for all $k$ (this program needs also $m$ to construct $T$ and $\bar{T}$, but $m$ is given implicitly as the length of $i$). Since one of the paths goes along $\omega$, we conclude that $K''(\omega) \leqslant m + \mathcal{O}(1) = M_\infty(\omega) + \mathcal{O}(1)$. □

The next two theorems provide the connection between $K_\infty$ and $M_\infty$.

**Theorem 5.** $K_\infty(\omega) \leqslant 2M_\infty(\omega) + \mathcal{O}(1)$.

**Theorem 6.** *There is a sequence $\omega^m$ of infinite strings such that $M(\omega^m) \leqslant m + \mathcal{O}(1)$ and $K_\infty(\omega^m) \geqslant 2m$ (hence $M_\infty(\omega^m)$, $M(\omega^m) = m + \mathcal{O}(1)$, $K_\infty(\omega^m) = 2m + \mathcal{O}(1)$).*

**Proof.** (The original proof of Theorem 5 was simplified significantly by An.A. Muchnik.) First, let us define a game that is relevant to both Theorems 5 and 6 and may be interesting in its own right.

Let $k, l$ be integer parameters. The $(k, l)$-*game* is played by two players called the Man (M) and the Nature (N). On its moves, N builds a binary rooted tree. More specifically, during its move N adds a binary string to a finite set $T$ (initally empty). Without loss of generality, we may assume that N is allowed to add any finite number of strings to $T$ at any move. On his moves, M may color certain binary strings using colors from the set $\{1, 2, \ldots, l\}$ (several colors may be attached to the same string; attached colors cannot be removed later).

The game stops after a finite number of moves if

(1) $T$ is not a tree (that is, there are $x \in T$ and $y \notin T$ such that $y$ is a prefix of $x$); in this case M wins, or

(2) for some $n$, the number of strings of length $n$ in $T$ (the number of nodes having height $n$) exceeds $k$; in this case M also wins, or

(3) there are two different strings of the same length colored by the same color; in this case N wins.

Otherwise the game lasts indefinitely long, and the winner is determined as follows. Let $T$ be the ultimate tree (formed by all strings included in $T$ at all steps). An infinite 0–1-sequence is called an *infinite branch* of $T$ if $\omega_{1:n} \in T$ for all $n$.

M wins if for any infinite branch $\beta$ there exists a color $c$ such that all but finitely many nodes of $\beta$ are colored by $c$ (and, may be, by other colors). Otherwise N wins.

(One may give the following interpretation to this game. The tree built by Nature is the tree of all breeds of animals, and nodes at height $n$ are breeds existing at time $n$. The coloring is giving names to breeds. Thus Man is required to give stable names to all eternal breeds.)

We will use also a modified version of this game where rule (1) is omitted and the definition of an infinite branch is changed as follows: sequence $\omega$ is an infinite branch if all but finitely many prefixes of $\omega$ are in $T$. (Obviously, the modified game is more difficult for M than the original one.)

The following two lemmas play a key role in the proof of Theorems 5 and 6.

**Lemma 1.** *For any k, there is a computable winning strategy for M in the modified* $(k, k^2)$-*game* (*the winning algorithm has k as an input*).

**Lemma 2.** *N has a computable winning strategy in the* $(k, l)$-*game if* $l < k^2/4$.

Before proving these lemmas, let us finish the proof of Theorems 5 and 6 using them.

Theorem 5 requires us to prove that $K_\infty(\omega) < 2M_\infty(\omega) + \mathcal{O}(1)$.

Fix $\omega$. Let $T = \{x \mid K(x|l(x)) \leqslant M_\infty(\omega)\}$. Then for any $n$ the set $T$ has no more than $k = 2^{M_\infty(\omega)+1}$ strings of length $n$. According to our assumption, $\omega_{1:n} \in T$ for all but finitely many $n$. Thus, $\omega$ is an infinite branch in $T$. Consider now the following strategy for N in modified $(k, k^2)$-game: N just enumerates $T$ (ignoring M's replies). M can defeat this strategy using his computable strategy that exists according to Lemma 1.

Since both M and N are using computable strategies, the set $C=\{\langle x,p\rangle \mid \text{node } x \text{ gets color } p \text{ at some stage}\}$ is enumerable. As M wins, there is a color $p$ that is attached to $\omega_{1:n}$ for all sufficiently large $n$. Each color can be considered as binary string of length $2(M_\infty(\omega)+1)$, since there are at most $k^2$ colors.

The following algorithm computes $\omega_{1:n}$ given $n$ and $p$. First find the value $k=2^{M_\infty(\omega)+1}=2^{l(p)/2}$. Second, enumerate $C$ until a pair $\langle x,p\rangle$ appears with $l(x)=n$, i.e., until some node $x$ having depth $n$ gets color $p$. Then return $x$. For all sufficiently large $n$ this algorithm will return $\omega_{1:n}$ (since the infinite branch $\omega$ has color $p$ assigned).

The program $q$ to compute $\omega_{1:n}$ given $n$ for almost all $n$ consists of the above algorithm with the string $p$ appended. Thus, the length of $q$ is $2M_\infty(\omega)+\mathcal{O}(1)$, and the Theorem 5 (modulo Lemma 1) is proved.

Now let us derive Theorem 6 from Lemma 2. We need to prove that there exist infinite sequences $\omega^0,\omega^1,\ldots$ such that $M(\omega^m)\leqslant m+\mathcal{O}(1)$ and $K_\infty(\omega^m)\geqslant 2m$.

For any fixed $m$ consider the following strategy for M. He enumerates all triples $\langle p,n,x\rangle$ such that $p(n)=x$; if it turns out that $l(x)=n$ and $l(p)<2m$, he assigns color $p$ to string $x$. This strategy may be performed by an algorithm having $m$ as an input.

Let $k=2^{m+1}$, $l=2^{2m}-1$. Since $l<k^2/4$, Lemma 2 guarantees that N could defeat this strategy using its own computable strategy. Therefore, there exists an algorithm $A$ that given $m$ generates a tree $T^m$ which has an infinite branch $\omega$ that is not properly colored, i.e., there is no $p$ of length less than $2m$ such that $p(n)=\omega_{1:n}$ for almost all $n$. In other words, $K_\infty(\omega)\geqslant 2m$.

On the other hand, $M(\omega)\leqslant m+\mathcal{O}(1)$. Indeed, let $n$ be a natural number. Let us describe a program of size $m+\mathcal{O}(1)$ that computes $\omega_{1:n}$. Consider an algorithm $B$ that for a given string $q$ of length $m+1$ and for any $n$ uses $A$ to generate $T^m$ and waits until $q$ nodes (here $q$ is identified with its ordinal number among all strings of length $m+1$) at height $n$ appear. Then $B$ outputs the node that appeared last. Since $\omega_{1:n}\in T^m$, for some $q$ the output will be equal to $\omega_{1:n}$. The string $q$ appended to $B$ constitutes a program to compute $\omega_{1:n}$ given $n$. This program has size $m+\mathcal{O}(1)$.

Theorem 6 is proved (modulo Lemma 2). □

Now we have to prove Lemmas 1 and 2.

Recall that Lemma 1 says that for any $k$, there is a computable winning strategy for M in the modified $(k,k^2)$-game (the winning algorithm has $k$ as an input).

**Proof** (*Using An. Muchnik's argument*). Let M use $k^2$ colors indexed by pairs $(a,b)$, where $a$ and $b$ are natural numbers in range $1..k$. Let us explain how the color $(a,b)$ is assigned. (Different colors are assigned independently.) Observing the growing set $T$, M looks for all pairs of strings $u$ and $v$ such that

(a) $u$ has number $a$ if we count all the (already appeared) strings in $T$ of the same length as $u$ in the lexicographic order;

(b) $v$ has number $b$ if we count all the (already appeared) strings in $T$ of the same length as $u$ in the reverse lexicographic order;

(c) $u$ is a prefix of $v$.

After such a pair of strings is found, any prefix of $u$ gets color $(a,b)$ unless some other string of the same length already has this color (and M is prohibited to use $(a,b)$ again on that level). Then M looks for another pair of strings $u$ and $v$ with the same properties, etc.

We need to prove that this strategy guarantees that any infinite branch will be colored uniformly starting at some point. Let $T$ be the set of all strings that N gives (at all steps). Let $\omega$ be an infinite branch, so $\omega_{1..n} \in T$ for all sufficiently large $n$. For these $n$ let $a_n$ denote the lexicographic number of $\omega_{1..n}$ in the set $T_n$ of all strings of length $n$ that are in $T$, and let $b_n$ denote the inverse lexicographic number of $\omega_{1..n}$ in $T_n$. Let $a = \limsup a_n$ and $b = \limsup b_n$. We claim that for sufficiently large $n$, the string $\omega_{1..n}$ will have color $(a,b)$.

Indeed, consider a pair $(u,v)$ that satisfies the conditions listed above. Let us prove first that for sufficiently long sequences only prefixes of $\omega$ have chance to get colored with color $(a,b)$. Indeed, for large enough $n$ we have $a_n \leqslant a$, so sufficiently long strings $u$ are "on the right of $\omega$" or are prefixes of $\omega$. ("On the right of $\omega$" means that $u$ follows the prefix of $\omega$ having the same length, in the lexicographic order.) For the same reasons all sufficiently long strings $v$ are on the left of $\omega$ or are prefixes of $\omega$. Therefore, the only chance for $u$ to be a prefix of $v$ (if both are long enough) is when both $u$ and $v$ are prefixes of $\omega$. Therefore, no other long strings (except prefixes of $\omega$) could get color $(a,b)$.

According to the definition of $a$ and $b$ there are infinitely many $n$ such that $a_n = a$ and infinitely many $m$ such that $b_m = b$. Choose a pair of such $n$ and $m$; assume that $n \leqslant m$. The strings $u = \omega_{1..n}$ and $v = \omega_{1..m}$ will be discovered after all strings of length $n$ and $m$ appear in the enumeration of $T$ since they will have correct ordinal numbers. And all prefixes of $u$ will get color $(a,b)$ unless some other vertex of the same length already has this color. (And this is possibly only for short strings, as we have seen). Since $u$ may be arbitrarily long, all sufficiently long prefixes of $\omega$ will get color $(a,b)$. Lemma 1 is proved. □

Lemma 2 says that N has a computable winning strategy in $(k,l)$-game if $l < k^2/4$.

**Proof.** Let $m = k/2$. First we introduce some terminology. We consider finite trees $T$ with $m$ distinguished leaves at the height equal to height of the tree. These distinguished leaves are called *tops* of the tree. The $m$ paths from the root to $m$ tops are called *trunks* of the tree. All the nodes that belong to the trunks are called *trunk nodes*; others are called *side nodes*.

We call a tree $T'$ an *extension of a tree* $T$ if (a) $T \subset T'$; (b) $T'$ does not contain new vertices on the levels that exist in $T$ (i.e., any string is $T' - T$ is longer than any string in $T$); (c) all trunks of $T'$ continue those of $T$ (that is, $j$th trunk of $T'$ continues $j$th trunk of $T$ for all $j \leqslant m$).

First N builds any tree $T_0$ of width $m$ that has $m$ trunks. Denote its height by $h_0$. Then N continues all the $m$ trunks of $T_0$ (for example, by adding, for any top $v$, nodes $v0$, $v00$, and so on) and waits until M starts to color nodes on the trunks (otherwise he looses).

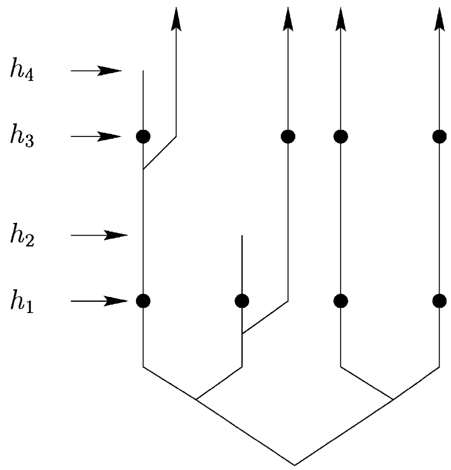


Fig. 2. Getting side nodes colored ($m = 4$).

More specifically, N waits until there exists $h_1 > h_0$ such that the nodes at height $h_1$ on all $m$ trunks are colored. We call these nodes *special* ones. The colors of special nodes are be pairwise different, as the special nodes are at the same height (otherwise M looses). Let $h_2$ be the height of trunks when M colors the last special node ($h_2 > h_1$).

N has just forced M to use $m$ different colors and has constructed a finite tree of width $m$. However, we wish (for the next iteration) that the nodes colored in $m$ different colors do not belong to trunks at the expense of increasing the width of the tree by 1. This is done as follows. Once N has forced M to color $m$ special nodes at the same height $h_1$, it chooses one trunk and cuts it (this means that N will not continue that trunk). Then N takes the father of the special node on that trunk and starts from the father another trunk instead of the cut trunk. The nodes lying on the cut trunk from the height $h_1$ to $h_2$ become side nodes. Thus at least one side node is colored. Call this node a *distinguished* node (see Fig. 2). After that, N still grows $m$ trunks in parallel (continuing $m - 1$ non-cut trunks and the trunk having a branch with the distinguished node) until M colors $m$ nodes on $m$ trunks at a new height $h_3 > h_2$.

Call these nodes as the *new special nodes*. Now N chooses a trunk whose new special node is colored in a color different from the color of the distinguished node, cuts it and starts a new trunk from its node at height $h_3 - 1$. We thus obtain the second side node colored in a color different from the color of the distinguished node. Call this side node also as a distinguished node. Thus we have two distinguished side nodes having different colors.

This process is repeated $m$ times. Each time N cuts a trunk whose special node is colored in a color different from the colors of the existing distiguished nodes (such a special node exists while the number of distinguished nodes is less than $m$). After $m$ repetitions we have a tree of width $m + 1$ that has $m$ distinguished side nodes colored in $m$ different colors (Fig. 3).

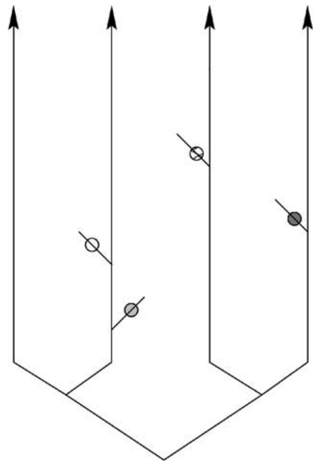

Fig. 3. A tree constructed by N after the strategy $S_1$ terminates. Here $m = 4$, and the width of the tree is $m + 1 = 5$. The circles represent $m$ side nodes colored in pairwise different colors. Some other nodes may be colored also.

The described strategy will be denoted by $S_1$. Its starting point may be any tree $T$ with $m$ trunks. It either terminates and constructs an extension $T'$ of $T$ such that $T' - T$ is colored in $m$ different colors, or wins. The set $T' - T$ has width $m + 1$.

Now let us describe the induction step. Assume $X$ is a subset of a tree $T$. Let $\mathsf{colors}(X)$ [$\mathsf{sidecolors}(X)$] denote the set of colors of all nodes [all side nodes] in $X$.

Assume we have a strategy $S_i$ ($i < m$) for N with the following properties. Starting from any tree $T$ with $m$ trunks it constructs a finite extension $T'$ of $T$ such that the difference $T' - T$ has width at most $m + i$ and $|\mathsf{sidecolors}(T' - T)| \geqslant im$.

Our goal is to define a strategy $S_{i+1}$ satisfying the same conditions (for increased value of $i$). We define first an auxilliary strategy $\tilde{S}_{i+1}$ that, starting from any tree $T$ with $m$ trunks, constructs a finite extension $T'$ of $T$ such that the difference $T' - T$ has width at most $m + i$, $|\mathsf{colors}(T' - T)| \geqslant (i + 1)m$, and $|\mathsf{sidecolors}(T' - T)| \geqslant im$ (or $\tilde{S}_{i+1}$ wins).

The strategy $\tilde{S}_{i+1}$ given a tree $T$ works as follows. Apply $S_i$ starting from $T$. Wait until $S_i$ terminates. Let $T_1$ be the continuation of $T$ constructed by $S_i$. Then $|\mathsf{sidecolors}(T_1 - T)| \geqslant im$. Apply $S_i$ starting from $T_1$. Wait until $S_i$ constructs a continuation $T_2$ of $T_1$ with $|\mathsf{sidecolors}(T_2 - T_1)| \geqslant im$. Applying $S_i$ many times, we get $T_1, T_2, T_3, \ldots$ . Wait until there exist $j$ and $s$ such that $j \leqslant s$ and all the nodes along all the trunks inside $T_j - T_{j-1}$ at step $s$ are colored and each trunk has its own color (if no such $j$ and $s$ exist, the strategy $\tilde{S}_{i+1}$ never terminates and wins). Let $T' = T_s$. The tree $T_s$ has $im$ different colors on side nodes in $T_j - T_{j-1}$ and $m$ new colors on nodes on $m$ trunks.

Now we are able to define the strategy $S_{i+1}$. Starting from a tree $T$ it works as follows. Apply $\tilde{S}_{i+1}$ starting from $T$. Wait until it terminates. Let $T_1$ denote the resulting tree. The set $\mathsf{colors}(T_1 - T)$ has at least $(i + 1)m$ colors. The problem, however, is

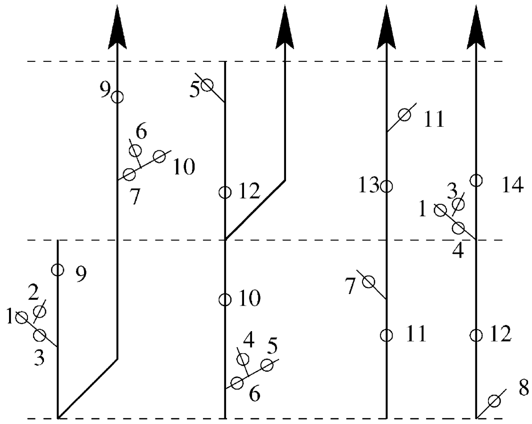


Fig. 4. A possible final position when $S_3$ is applied ($m = 4$). The lines with arrows are trunks. Circles represent nodes that M was forced to color by $\tilde{S}_3$; numbers represent their colors.

that some of them may be used for trunk nodes only. In this case choose a trunk of $T_1$ that has a node colored in a color $c \in \mathsf{colors}(T_1 - T) - \mathsf{sidecolors}(T_1 - T)$. Let $j$ be the number of that trunk. We add to $T_1$ a new branch starting from the $j$th top of $T$ and declare this branch a new trunk of $T_1$; the old $j$th trunk is not a trunk anymore. This operation increases the width of $T_1 - T$ to $m + i + 1$. The gain is that the set $\mathsf{sidecolors}(T_1 - T)$ has got a new color $c$. So $|\mathsf{sidecolors}(T_1 - T)| \geqslant im + 1$ now. If it happens that the set $\mathsf{sidecolors}(T_1 - T)$ already has at least $(i + 1)m$ colors, we stop. Otherwise, we apply once more the strategy $\tilde{S}_{i+1}$ starting from $T_1$. We get $T_2$ such that $|\mathsf{colors}(T_2 - T_1)| \geqslant (i + 1)m$. As $|\mathsf{sidecolors}(T_1 - T)| < (i + 1)m$, the set $\mathsf{colors}(T_2 - T_1)$ has at least one color that does not belong to $\mathsf{sidecolors}(T_1 - T)$. We choose again a color $c$ from $\mathsf{colors}(T_2 - T_1) - \mathsf{sidecolors}(T_1 - T)$, choose a trunk node in $T_2 - T_1$ colored by $c$, make a new trunk from the top of $T_1$ lying on that trunk and thus get $\mathsf{sidecolors}(T_2 - T) \geqslant \mathsf{sidecolors}(T_1 - T) + 1 \geqslant im + 2$. Repeating this trick at most $m$ times, we obtain an extension $T'$ such that $\mathsf{sidecolors}(T' - T) \geqslant (i+1)m$ and the width of $T' - T$ is at most $m + i + 1$ (Fig. 4).

The induction step is described. Note that the strategy $\tilde{S}_m$ wins in the $2m, (m^2 - 1)$-game. □

## References

[1] G.J. Chaitin, On the length of programs for computing finite binary sequences: statistical considerations, J. ACM 16 (1969) 145–159.

[2] G.J. Chaitin, Information-theoretic characterizations of recursive infinite strings, Theoret. Comput. Sci. 2 (1976) 45–48.

[3] B. Durand, S. Porrot, Comparison between the complexity of a function and the complexity of its graph, Theoret. Comput. Sci. 271 (this Vol.) (2002) 37–46.

[4] B. Durand, A. Shen, N. Vereshchagin, in: Descriptive complexity of computable sequences, STACS'99, Lecture Notes in Computer Sciences, vol. 1563, Springer, Berlin, March 1999.

[5] A.N. Kolmogorov, Three approaches to the quantitative definition of information, Problems Inform. Transmission 1 (1) (1965) 1–7.
[6] M. Li, P. Vitányi, An Introduction to Kolmogorov Complexity and its Applications, 2nd ed., Springer, Berlin, 1997.
[7] D.W. Loveland, A variant of Kolmogorov concept of complexity, Inform. and Control 15 (1969) 510–526.
[8] R.J. Solomonoff, A formal theory of inductive inference, part 1 and part 2, Inform. and Control 7 (1964) 1–22, 224–254.
[9] V.A. Uspensky, A.Kh. Shen, Relations between varieties of Kolmogorov complexities, Math. Systems Theory 29 (1996) 271–292.
[10] A.K. Zvonkin, L.A. Levin, The complexity of finite objects and the development of the concepts of information and randomness by means of theory of algorithms, Russian Math. Surveys 25 (6) (1970) 83–124.