Descriptive complexity of computable sequences

Bruno Durand*, Alexander Shen**, and Nikolai Vereshagin***

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.

1 Introduction

The notion of Kolmogorov entropy (=complexity) for finite binary strings was introduced in the 60ies independently by Solomonoff, Kolmogorov and Chaitin [7, 4, 1]. There are different versions (plain Kolmogorov entropy, prefix entropy, etc. see [8] 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 $\mathrm{K}(x|y)$ as defined in [4], 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 [5] for details and references); if prefixes have low complexity, the sequence is computable. In the sequel, we study the latter type.

Let $\mathrm{K}(x)$, $\mathrm{K}(x|y)$ denote the plain Kolmogorov entropy (complexity) of a binary string x and the conditional Kolmogorov entropy (complexity) of x when y (some other binary string) is known. That is, $\mathrm{K}(x)$ is the length of the shortest program p that prints x; $\mathrm{K}(x|y)$ is the length of the shortest program that prints x given y as input. (For details see [5] or [9].)

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

Let us recall the following criteria of computability of ω in terms of entropy of its finite prefixes.

(a) ω is computable if and only if $\mathrm{K}(\omega_{1:n}|n) = \mathcal{O}(1)$. This result is attributed in [6] to A.R. Meyer (see also [9, 5]).
(b) ω is computable if and only if $\mathrm{K}(\omega_{1:n}) < \mathrm{K}(n) + \mathcal{O}(1)$ [2].

* LIP, Ecole Normale Supérieure de Lyon, 46 Allée d'Italie, 69364 Lyon Cedex 07, France. E-mail: `Bruno.Durand@ens-lyon.fr`

** Institute of Problems of Information Transmission, Moscow, Russia. E-mail: `shen@mccme.ru`

*** Dept. of Mathematical Logic and Theory of Algorithms, Moscow State University, Vorobjevy Gory, Moscow 119899, Russia. E-mail: `ver@mech.math.msu.su`. The work was done while visiting LIP, Ecole Normale Supérieure of Lyon.

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

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

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

($l(p)$ 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 ω: $M(\omega)$ is finite iff ω is computable.

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

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

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

The difference between $K(\omega)$ and $M(\omega)$ can be explained as follows: $M(\omega) \leq m$ 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, $K(\omega) \leq m$ means that there is a one such program that works for all n. Thus, $M(\omega) \leq K(\omega)$ for all ω, 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 $M(\omega^m) \leq m + \mathcal{O}(1)$ and $K(\omega^m) \geq \alpha(m) - \mathcal{O}(1)$.

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

$$\begin{aligned} K_\infty(\omega) &= \min\{l(p) \mid \forall^\infty n \; p(n) = \omega_{1:n}\} \\ M_\infty(\omega) &= \limsup_n \mathrm{K}(\omega_{1:n}|n) = \min\{m \mid \forall^\infty n \exists p \; (l(p) \leq m \text{ and } p(n) = \omega_{1:n})\}, \end{aligned}$$

($\forall^\infty n$ 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 ω is implied by Meyer's theorem.

Surprisingly, it turns out that $K_\infty(\omega) \leq 2M_\infty(\omega) + \mathcal{O}(1)$ (theorem 5) so the difference between K_∞ and M_∞ is not so large as between K and M. We stress that this result is 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_∞ and M_∞ 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 by a diagram (Fig. 1).

$$\begin{tikzcd}
K'(\omega) \arrow[d] & K_\infty(\omega) \arrow[l] \arrow[d] & K(\omega) \arrow[l] \arrow[d] \\
K''(\omega) & M_\infty(\omega) \arrow[l] & M(\omega) \arrow[l]
\end{tikzcd}$$

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

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 ($K'(\omega) \leq K_\infty(\omega) + \mathcal{O}(1)$) and Theorem 4 ($K''(\omega) \leq M_\infty(\omega) + \mathcal{O}(1)$).

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

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

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

On the last diagram no arrow could be inverted. Indeed, $K''(\omega)$ is finite while $K'(\omega)$ is infinite for a sequence ω 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 $K_\infty(\omega) \leq M(\omega) + \mathcal{O}(1)$ and $K'(\omega) \leq M_\infty(\omega) + \mathcal{O}(1)$ are true. The first one is not true, as Theorem 6 implies. We don't 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 $\limsup_n \mathrm{K}(x|n)$ equal to $\mathrm{K}'(x) + \mathcal{O}(1)$ or not?[1]

2 Theorems and proofs

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

[1] It was shown recently by the third author that $\limsup_n \mathrm{K}(x|n) = \mathrm{K}'(x) + \mathcal{O}(1)$.

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. □

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

Proof. Let x_m be the lexicographically first string x of length $\alpha(m)$ such that $\mathrm{K}(x|\alpha(m)) \geq \alpha(m)$. (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, $M(\omega^m) \geq \mathrm{K}(x_m|\alpha(m)) \geq \alpha(m)$. On the other hand, $K_\infty(\omega^m) \leq m + \mathcal{O}(1)$. Indeed, the set $\{x \mid \mathrm{K}(x|l(x)) < l(x)\}$ 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 $\log m + \mathcal{O}(1)$. □

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

Proof. Let c be a constant (to be specified later). The set $E = \{\langle x, k\rangle \mid \mathrm{K}(x) < \alpha(k) + c\}$ 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 ω^m and hence $s(m)$. Thus $\mathrm{K}(s(m)) \leq K(\omega^m) + \mathcal{O}(1)$. On the other hand, given $s(m)$ we can find the (lexicographically) first string x_m of entropy $\alpha(m)$ or more, therefore, $\alpha(m) \leq \mathrm{K}(x_m) \leq \mathrm{K}(s(m)) + \mathcal{O}(1)$. Hence $\alpha(m) \leq K(\omega^m) + \mathcal{O}(1)$.

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

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 $0^{s(m,n)}1111111....$ If $n \geq s(m)$, then p_m outputs the correct prefix of ω^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 $\log m + \mathcal{O}(1)$. □

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

Proof. Let $m = M_\infty(\omega) + 1$. Consider the set $T = \{x \mid \mathrm{K}(x|l(x)) < m\}$. By definition, all sufficiently long prefixes of ω 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 ξ such that x is a prefix of ξ and all prefixes of ξ 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 $\overline{T}$ of all good strings is $\mathbf{0}''$-decidable.

This set can be represented as an union of non-overlapping infinite paths: consider all the strings in order of increasing length; if a string in $\overline{T}$ is found that is not already included in one of the paths, take a path that starts with it (if there are many of them, choose the lexicographically first, i.e., turn to the left when possible). The number of different paths does not exceed 2^m. This decomposition process is $\mathbf{0}''$-effective, i.e., 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 $\overline{T}$, but m is given implicitly as the length of i). Since one of the paths goes along ω, we conlude that $K''(f) \leq m + \mathcal{O}(1) = M_\infty(\omega) + \mathcal{O}(1)$. □

The next two theorems provide the connection between K_∞ and M_∞.

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

Theorem 6. *There is a sequence ω^m of infinite strings such that $M(\omega^m) \leq m+\mathcal{O}(1)$ and $K_\infty(\omega^m) \geq 2m$ (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). On his moves, M may color certain binary strings using colors from the set $\{1, 2, \dots, l\}$ (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 depth 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 β there exists a color c such that all but finitely many nodes of β 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 the rule (1) is omitted and the definition of an infinite branch is changed as follows: sequence ω is an infinite branch if all but finitely many prefixes of ω 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 ω. Let $T = \{x \mid \mathrm{K}(x|l(x)) \le M_\infty(\omega)\}$. 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 ω 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 $C = \{\langle x, p\rangle \mid$ node x gets color p 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 ω 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 $2M(\omega) + \mathcal{O}(1)$, 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 $M(\omega^m) \leq m + \mathcal{O}(1)$ and $K_\infty(\omega^m) \geq 2m$.

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 $l < k^2/4$, the 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 ω 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, $K_\infty(\omega) \geq 2m$.

On the other hand, $M(\omega) \leq m + \mathcal{O}(1)$. 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 in the lexicographic order;
(b) v has number b if we count all the (already appeared) strings in T 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 ω 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 ω have chance to get colored with color (a, b). Indeed, for large enough n we have $a_n \leq a$, so sufficiently long strings u are "on the right of ω" or are prefixes of ω. ("On the right of ω means that u follows the prefix of ω having the same length, in the lexicographic order.) For the same reasons all sufficiently long strings v are on the left of ω or are prefixes of ω. 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 ω. Therefore, no other long strings (except prefixes of ω) 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 $n \leq m$. 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 ω will get color (a, b). Lemma 1 is proved.

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

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. Those 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*; other 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, jth trunk of T' continues jth trunk of T for all $j \leq m$).

First N builds any tree T_0 of width m that has m trunks. 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). More specifically, N waits until there exists h_1 such that the nodes at height h_1 on all m trunks are colored. We call those 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 ($h_2 \geq h_1$).

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 the trunks 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 that node a *distinguished* node. 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 those nodes 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 distiguished node. Call that side node also a distiguished node. Thus we have two distinduished 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.

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 $m + i$ and $|\,\mathsf{sidecolors}(T' - T)| \geq im$.

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 $m+i$, $|\,\mathsf{colors}(T'-T)| \geq (i+1)m$, and $|\,\mathsf{sidecolors}(T'-T)| \geq im$ (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 $|\,\mathsf{sidecolors}(T_1 - T)| \geq im$. Apply S_i starting from T_1. Wait until S_i constructs a continuation T_2 of T_1 with $|\,\mathsf{sidecolors}(T_2 - T_1)| \geq im$. Applying S_i many times, we get $T_1, T_2, T_3, \ldots$. Wait until there exist j and s such that $j \leq s$ 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 startegy $\tilde{S}_{i+1}$ 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 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 jth top of T and declare this branch a new trunk of T_1; the old jth 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 $|\,\mathsf{sidecolors}(T_1 - T)| \geq im + 1$ 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 $|\,\mathsf{colors}(T_2 - T_1)| \geq (i+1)m$. 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 $\mathsf{sidecolors}(T_2 - T) \geq \mathsf{sidecolors}(T_1 - T) + 1 \geq im + 2$. Repeating this trick at most m times, we obtain an extension T' such that $\mathsf{sidecolors}(T' - T) \geq (i+1)m$ and the width of $T' - T$ is at most $m + i$.

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

References

1. G.J. Chaitin. "On the length of programs for computing finite binary sequences: statistical considerations," *J. of ACM*, 16:145–159, 1969.
2. G.J. Chaitin. "Information-theoretic characterizations of recursive infinite strings," *Theor. Comp. Sci.,* 2:45–48, 1976.
3. B. Durand, S. Porrot. Comparison between the complexity of a function and the complexity of its graph. In MFCS'98. *Lecture Notes in Computer Science,* 1998.
4. A.N. Kolmogorov. "Three approaches to the quantitative definition of information." *Problems of Information Transmission*, 1(1):1–7, 1965.
5. M. Li, P. Vitányi. An Introduction to Kolmogorov Complexity and its Applications. Second edition. Springer Verlag, 1997.
6. D.W. Loveland. "A variant of Kolmogorov concept of Complexity", *Information and Control,* 15:510–526, 1969.
7. R.J. Solomonoff. "A formal theory of inductive inference, part 1 and part 2," *Information and Control,* 7:1–22, 224-254, 1964.
8. V.A. Uspensky, A.Kh. Shen'. "Relations between varieties of Kolmogorov complexities," *Math. Systems Theory,* 29:271–292, 1996.
9. 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):83–124, 1970.