Electronic Colloquium on Computational Complexity, Report No. 86 (2001)

Kolmogorov Complexity Conditional to Large Integers

Nikolai K. Vereshchagin*

1 Introduction

Kolmogorov complexity of a string x is defined as the minimal length of a program that prints x:

$$K(x) = \min\{l(p) \mid p(\Lambda) = x\},$$

where Λ stands for the empty string. We assume that some optimal programming language is fixed; formal definitions are presented in the next section. In a more general framework, any "algorithmic problem" has its Kolmogorov complexity, which is defined as the minimal length of a program solving the problem. We will not discuss in this paper the general notion of an algorithmic problem (see [7] for such discussion), as our paper is devoted to very specific problems. The plain Kolmogorov complexity, $K(x)$, is the Kolmogorov complexity of the problem "print x". Likewise the conditional Kolmogorov complexity, defined as

$$K(x|y) = \min\{l(p) \mid p(y) = x\},$$

is the complexity of the problem "given y print x".

The subject of this paper is the problem "print x given any sufficiently large integer". Its Kolmogorov complexity is denoted by $K_{\lim}(x)$:

$$K_{\lim}(x) \quad = \quad \min\{l(p) \mid p(n) = x \text{ for all but finitely many } n\}.$$

*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 L.I.P., Ecole Normale Supérieure of Lyon.

ISSN 1433-8092

We consider also a related problem "given any number greater than n print x" whose complexity is denoted by $K(x|\mathtt{gr}\,n)$. The problem which stands in the condition of this formula, namely, the problem "print a number greater than n" is interesting in its own right. Its complexity, $\min\{K(i) \mid i \geqslant n\}$, which might be denoted by $K(\mathtt{gr}\,n)$, was denoted by $m(n)$ in [8] and by $\alpha(n)$ in [2]; Li and Vitányi in [4] use the notation $m(n)$ too. We will also use the notation $m(n)$. The function $m(n)$ has a natural inverse function

$$B(k) = \max\{n \mid K(n) \leqslant k\},$$

the maximum number with complexity k or fewer. More precisely, the following holds:

$$m(n) \leqslant k \Leftrightarrow n \leqslant B(k)$$
$$m(B(k)) = k + O(1).$$

The first assertion is a direct corollary of the definitions: both inequalities are equivalent to the existence of $i \geqslant n$ with $K(i) \leqslant k$. To prove the second one note that $K(B(k)) \leqslant k$. Since $K(i) > k$ for any $i > B(k)$, we have $m(B(k)) = K(B(k)) \leqslant k$. On the other hand, $K(n+1) = K(n) + O(1)$ for any n, hence $k < K(B(k)+1) = K(B(k)) + O(1) = m(B(k)) + O(1)$.

Both functions $m(n)$ and $B(k)$ increase. The function $B(k)$ is a version of the well known *Busy Beaver* function due to Rado [6], the maximal number of 1's in the output of any Turing machine with k states and purely binary tape alphabet (no blanks) when it is started on the empty input. Other versions of inverse function to $m(n)$ are presented in [2].

It is easy to see that $K_{\mathrm{lim}}(x)$ is equal to the limit of the increasing sequence $\{K(x|\mathtt{gr}\,n)\}_{n=0,1,2,\ldots}$. It was noted by An. Muchnik and S. Positselskij (personal communication) that $K_{\mathrm{lim}}(x)$ coincides with the plain Kolmogorov complexity of x relativized with $0'$, the set of programs that halt on the empty input:

$$K_{\mathrm{lim}}(x) = K^{0'}(x) + O(1);$$

this equality will be proven later.

In this paper we show that $K_{\mathrm{lim}}(x)$ coincides with its non-uniform version $K_{\mathrm{limsup}}(x)$ defined as follows. In the definition of $K_{\mathrm{lim}}(x)$ we require that program p prints x given any sufficiently large n. Let us allow now the program p to depend on n:

$$K_{\mathrm{limsup}}(x) = \min\{m \mid \text{ for all but finitely many } n \text{ there is } p \text{ such that } l(p) \leqslant m,\ p(n) = x\}.$$

In other words, $K_{\mathrm{limsup}}(x) = \limsup_n K(x|n)$. The main result of this paper states that $K_{\mathrm{limsup}}(x) = K_{\mathrm{lim}}(x) + O(1)$ (thus we answer a question left open in [1]). To prove this statement we define a two-players game which is interesting in its own right and design a winning strategy in this game.

Acknowledgments. The author thanks B. Durand and A. Shen for bringing his attention to the problem, An. Muchnik and S. Positselskij for theorem 1, and M. Ushakow for finding an error in the previous proof of theorem 2.

2 Preliminaries

Let $\{0,1\}^*$ stand for the set of all strings over the alphabet $\{0,1\}$. The length of a string x is denoted by $l(x)$, Λ stands for the empty string.

A *programming language* is a partial computable function F from $\{0,1\}^* \times \{0,1\}^*$ to $\{0,1\}^*$. The first argument of F is called a program, the second argument is called the input, and $F(p,x)$ is called the output of program p on input x. A programming language F is called *universal* if for any other programming language G there exists a string c_G such that $F(c_G p, x) = G(p,x)$ for any p, x. By Solomonoff – Kolmogorov theorem (see e.g. [4]) universal programming languages exist. We fix any universal programming language F and write $p(x)$ instead of $F(p,x)$.

The set of natural numbers is denoted by $\mathbb{N}$. If a natural number is considered as input to a program, it is represented in binary notation.

We shall use the following versions of Kolmogorov complexity:

$$\begin{aligned}
K(x) &= \min\{l(p) \mid p(\Lambda) = x\},\\
K(x|y) &= \min\{l(p) \mid p(y) = x\},\\
K(x|\mathtt{gr}\, n) &= \min\{l(p) \mid p(k) = x \text{ for all } k \geqslant n\},\\
K_{\mathrm{lim}}(x) &= \lim K(x|\mathtt{gr}\, n)\\
&= \min\{l(p) \mid (\forall^\infty n \in \mathbb{N})\ p(n) = x\}\\
K_{\mathrm{limsup}}(x) &= \limsup K(x|n)\\
&= \min\{m \mid (\forall^\infty n \in \mathbb{N})\ \exists p\ l(p) \leqslant m,\ p(n) = x\},
\end{aligned}$$

where $(\forall^\infty n \in \mathbb{N})$ stands for "for all but finitely many $n \in \mathbb{N}$".

We shall use the following well known facts (see [4]):

- $K(x) \leqslant l(x) + O(1)$;

- for any l there is a string x of length l with $K(x) \geqslant l$;
- for any computable function $f(x)$ there is a constant c such that $K(f(x)) \leqslant K(x) + c$ for all x in the domain of f;
- for any computable function $f(x, y)$ there is a constant c such that $K(f(x, y)) \leqslant K(x) + 2K(y) + c$ for all x, y in the domain of f.

Let $0'$ stand for the set $\{p \mid p \text{ halts on the empty input}\}$. Instead of computable functions consider functions that are computable by machines having an oracle answering any questions of the type "$x \in 0'$?". The notion of a programming language as well as other notions of this section obtained by this replacement are called *relativized by* $0'$. The relativized Kolmogorov complexity is denoted by $K^{0'}(x)$.

3 $K_{\lim}(x)$ and the plain Kolmogorov complexity

The difference between $K_{\lim}(x)$ and $K(x)$ may be illustrated by the following example due to Kamae [3]. Let x_k stand for the first string of length k and of complexity at least k, with respect to the lexicographical order:

$$K(x_k) \geqslant k, \qquad l(x_k) = k.$$

There is a constant c such that for all but finitely many n we have $K(x_k|n) \leqslant \log_2 k + c$. Indeed, let n be greater than the running time of any program of length k or fewer, when it is started on Λ. Given k and n we can find x_k by simulating n steps of the run of all the programs of length at most k. Hence

$$K_{\lim}(x_k) \leqslant \log_2 k + c$$

for all k. Actually we have $K(x_k|\mathtt{gr}\, B(k + c')) \leqslant \log_2 k + c$ for some c, c' and all k. Indeed, let $t(p)$ denote the running time of program p, and $t(p)$ is undefined if p does not halt. Then $K(t(p)) \leqslant l(p) + c'$ for some c' and all p. Hence $t(p) \leqslant B(l(p) + c')$.

The following theorem clarifies the notion of $K_{\lim}(x)$.

Theorem 1 (An. Muchnik, S. Positselskij). $K_{\lim}(x) = K^{0'}(x) + O(1)$.

Proof. Let q be a shortest program that prints x, with respect to the universal relativized programming language. In the run of this program only finitely many questions were made to the oracle, let those questions be "$p_1 \in 0'$?", ..., "$p_k \in 0'$?". Let n be the maximum running time of those programs among $p_1, \ldots, p_k$ that halt. Then given q and any number t greater than n we can find x without querying oracle, as we can answer ourselves all the questions by running the programs $p_1, \ldots, p_k$ within t steps. Hence, $K_{\mathrm{lim}}(x) \leqslant K^{0'}(x) + O(1)$.

Conversely, let $p(n) = x$ for all sufficiently large n. Given p, for $k = 0, 1, 2, \ldots$ we ask the oracle whether there are $n_1, n_2 \geqslant k$ such that $p(n_1)$ and $p(n_2)$ are defined and are different. This question may be reformulated as a question whether a certain program halts. Once a k for which there are no such n_1, n_2 is found, we ask the oracle for $n = k, k+1, k+2, \ldots$ whether $p(n)$ is defined. When such n is found we print $p(n)$. Thus we have defined a function $f(p)$ that is computable relative to $0'$ and such that $f(p) = x$ provided $p(n) = x$ for all sufficiently large n. Hence, $K^{0'}(x) \leqslant K_{\mathrm{lim}}(x) + O(1)$. □

Remark. Similar theorem holds for prefix complexity. It easily follows from the result of [5].

4 $K_{\mathrm{lim}}(x)$ and $K_{\mathrm{limsup}}(x)$

Theorem 2. $K_{\mathrm{limsup}}(x) = K_{\mathrm{lim}}(x) + O(1)$.

Proof. The inequality $K_{\mathrm{limsup}}(x) \leq K_{\mathrm{lim}}(x)$ is straightforward so we need to prove that $K_{\mathrm{lim}}(x) \leq K_{\mathrm{limsup}}(x) + O(1)$.

We want to show that

$$K_{\mathrm{limsup}}(x) < m \Rightarrow K_{\mathrm{lim}}(x) < m + c$$

for some constant c not depending on m and x. In other words,

$$\forall^\infty n \ K(x|n) < m \Rightarrow \exists p \ \forall^\infty n \ (l(p) < m + c, \ p(n) = x).$$

Obviously, it suffices to construct a computable function $F(p, n)$ such that

$$\forall^\infty n \ K(x|n) < m \Rightarrow \exists p \ \forall^\infty n (l(p) = m, G(p, n) = x). \qquad (1)$$

Fix m. We shall view pairs (n, x), where x is a string and n a positive integer, as cells of an infinite table (x indicates row, n column), strings of length m

as colors and we shall say that a cell (n, x) has color p if $G(p, n) = x$. So to define G we have to design a computable strategy of coloring certain cells of the table. To this end let us start a program, call it Nature, that enumerates cells (n, x) with $K(x|n) < m$ by putting a certain mark on each enumerated cell. Receiving marked cells we color some cells. Our goal is that after infinite number of steps, when all cells with $K(x|n) < m$ are marked, the following holds for all x:

(W) if all but finitely cells in xth row, (n, x), $(n + 1, x)$, ..., are marked then there is a color p such that all but finitely many cells in that row, (s, x), $(s + 1, x)$, ..., have color p; s may be greater than n.

So we obtain a game, called m-game, between two players, Nature and Mathematician. On its moves Nature marks certain cells, without loss of generality we may assume that it marks one cell per move. On her moves Mathematician may color some cells, she has a set of $l = 2^m$ colors. In every column, Nature is allowed to mark at most $k = 2^m$ cells. Mathematician is not allowed to use any color twice in the same column; otherwise G will not be a function. The game is played infinite number of moves and Mathematician wins if the above assertion (W) is true at the end of the game.

In the sequel we call any infinite continuous sequence of marked cells, (n, x), $(n + 1, x)$, $(n + 2, x)$, ..., a *path*. Thus Mathematician wins if all but finitely many cells of each path have the same color.

So we have to construct a computable winning strategy for Mathematician in the m-game. Now we forget that we know Nature's strategy—anyway we are not able to use this knowledge. We shall design a strategy that wins against any strategy of Nature. Also we may forget the specific values of k, the maximum number of marked cells in each column, an l, the number of colors, and ask for which k, l Nature wins and for which Mathematician wins.

Note that if $l < k$ then Nature wins. Indeed, in this case Nature just marks all the cells in the first k rows thus making k different paths. As $l < k$, for any n there is a path whose nth cell is not colored. Hence there is a path having infinite number of uncolored cells.

Lemma 1. *Mathematician has a winning strategy if $l = k$. This strategy is computable uniformly on k.*

Assume the lemma is true. Then for $k = l = 2^m$ let Mathematician use the computable winning strategy which exists by the above lemma. As Nature's strategy—recall that it marks the cell (n, x) when it discovers that

$K(x|n) < m$—is also computable, we obtain a computable process of coloring cells of the table, uniformly on m. As Mathematician wins, the function

$$G(p,n) = \text{ the cell in } n\text{th column which gets color } p$$

satisfies condition (1). And it is computable: given p, n we find m as the length of p and observe the play in the m-game until a cell in nth column gets color p. □

Proof of the lemma. The main problem is that we do not know at any moment whether Nature has finished or not to mark cells of a given column.

Assume for a moment that for every n we get know on some move that no more cells will be marked in nth column. Then our strategy could be as follows. Find the greatest N such that no more cells will be marked in columns $1, 2, \ldots, N$. For all $n \leqslant N$ define on marked cells of nth column a linear order as follows. To compare (n, x_1) and (n, x_2) find the least $t_1 \leq n$ such that all the cells $(t_1, x_1), (t_1 + 1, x_1), \ldots, (n, x_1)$ are marked, and find t_2 defined in the similar way using (n, x_2) instead of (n, x_1). Let $(n, x_1) < (n, x_2)$ if $t_1 < t_2$ or $t_1 = t_2$ and x_1 precedes x_2 in the lexicographical order. Color the ith marked cell in nth column with respect to this order in color i for all $i \leqslant k$. Let us prove that this strategy wins. Assume that after infinite number of moves, for all $n \geqslant n_0$ the cell (n, x) has been marked. Then, starting from $n = n_0$ the ordinal number of (n, x) among the marked cells of nth column does not increase. Therefore, it does not change starting from some $n = n_1$.

The winning strategy for Mathematician is as follows. Suppose we have to make the sth move. Call a sequence of cells $C_1 = (n_1, x_1)$, ..., $C_j = (n_j, x_j)$ *sound* if they stay in different rows and the following holds. If, in addition to cells marked so far, we mark all the cells in rows $x_1, \ldots, x_j$ to the right of $(n_1, x_1), \ldots, (n_j, x_j)$ respectively, where we assume that columns are numbered from the left to the right, then all columns will have no more than k marked cells. The maximum length of a sound sequence is equal to k and, moreover, for any $i < k$ any sound sequence C_1, ..., C_i of length i can be extended to a sound sequence of length k. The latter is proved as follows: take as C_{i+1} any cell that stands in a row different from those occupied by C_1, ..., C_i to the right of all marked cells, then take as C_{i+2} any cell that stands in a row different from those occupied by $C_1, \ldots, C_{i+1}$ to the right of all marked cells, and so on. The resulting sequence is sound.

Fix any one-to-one correspondence between cells and natural numbers. This correspondence induces a well order on cells. Then define a sequence of k cells, $C_1^s, \ldots, C_k^s$ by induction: C_i^s is the least cell C such that the sequence $C_1^s, \ldots, C_{i-1}^s, C$ is sound. In other words, $C_1^s, \ldots, C_k^s$ is the lexicographically least sound sequence of length k. This follows immediately from the fact that any sound sequence can be extended to a sound sequence of length k. We shall call C_i^s the *ith distinguished cell on move s* and the row where it stands the *ith distinguished row on move s*.

Figure 1: The cells number 1, 9 and 26 are distinguished; we assume that $k = 3$.

✓ 1	✓ 3	✓ 6	✓ 10	15	✓ 21	28
✓ 2	5	9	✓ 14	✓ 20	✓ 27	✓
✓ 4	✓ 8	✓ 13	19	✓ 26		
7	12	18	✓ 25			
11	✓ 17	24				
16	23					

Thus, we find distinguished rows and color the cell standing in ith distinguished row and in sth column in color i; recall that s is the number of the current move.

Mathematician's strategy is described. Let us prove that it wins. First note that if the first $i-1$ distinguished cells do not change on the move $s+1$, compared with sth move, then the ith distinguished cell does not decrease on the move $s+1$, compared with sth move: $C_i^s \leqslant C_i^{s+1}$. Indeed, the sequence of distinguished cells on move $s+1$ is sound also on move s, and therefore is lexicographically greater than the distinguished sequence no move s. As these sequences have the same prefix of length s, we get $C_i^s \leqslant C_i^{s+1}$.

Assume that all but finitely many cells in xth row, $C = (n, x)$, $(n+1, x)$, ..., are marked at the end of the game. To show that the strategy wins it suffices to prove that there is $i \leqslant k$ such that on all but finitely many moves the xth row is the ith distinguished row. We know that the sequence

consisting of the first distinguished cells $C_1^1, C_1^2, C_1^3, \ldots$ does not decrease. We shall prove that it is bounded from above by the cell $C = (n, x)$. Hence this sequence has a limit and it reaches its limit on some move t. If the limit cell C_1^t stands in xth row, we are done. Otherwise, we will show that starting from tth move the sequence of the second distinguished cells is also bounded by C. As it does not decrease starting from move t it also has a limit and it reaches its limit on some step $r \geqslant t$. Repeating this argument at most k times we will find $i \leqslant k$ such that xth row is the ith distinguished row on all but finitely many moves.

Thus we need to prove the following facts.

1) Assume that starting from some move t the following two assertions hold: (1) the jth distinguished cell does not change for all $j < i$ and (2) xth row is not among $i-1$ first distinguished rows. Then starting from move t the ith distinguished cell is less than or equal to C.

Proof. Assume that $s \geqslant t$. It suffices to prove that the cell C is an eligible candidate for the ith distinguished cell on move s. To prove this we need to verify that if we append C to the sequence of the first $i-1$ distinguished cells we get a sound sequence. By assumption C does not stand in any of the of the first $i-1$ distinguished rows. Let us prove that if we mark all cells in $i-1$ first distinguished rows to the right of distinguished cells and all cells in xth row to the right of C then no column will have $k+1$ marked cells. Suppose the contrary: a column number n' has $k+1$ marked cells after we have marked cells as described. One of those $k+1$ cells must stand in xth row; otherwise the sequence of the first $i-1$ distinguished cells is not sound on move s. Assume that that cell, (n', x), is marked on step r. Then on the move number $\max\{r, s\}$ it is already marked, hence on that move the sequence of the first $i-1$ distinguished cells is not sound. The contradiction shows that C is an eligible candidate for the ith distinguished cell on move s, hence $C_i^s \leqslant C$.

2) Assume that starting from some move t all the distinguished cells do not change. Then xth row is distinguished starting from some move.

Proof. Let n' be the number of any column to the right of all columns in which distinguished cells stand on tth move (and later) and to the right of nth column. Let the cell (n', x) be marked on step r. As the sequence of distinguished cells is sound on move number $\max\{t, r\}$, we conclude that the cell (n', x) stands in one of the distinguished rows on this move. $\square$

5 Conclusion

We have not considered an interesting question, how large is the least n for which $K(x|\mathtt{gr}\, n) = K_{\lim}(x)$? In the example from the beginning of section 3 we have shown that $K(x_k|\mathtt{gr}\, B(k+c)) \leqslant \log_2 k + c$. However, even in this case it is not clear whether $n = B(k+c)$ is enough to reach the lower limit of $K(x|\mathtt{gr}\, n)$ (note that $K_{\lim}(x_k) = K_{\lim}(k) + O(1)$). Is it true that $K(x_k|\mathtt{gr}\, B(k+c)) \leqslant K_{\lim}(k) + c$? What we are able to prove is the lower bound $n \geqslant B(K(x) - K_{\lim}(x))$. Indeed, if $K_{\lim}(x) = K(x|\mathtt{gr}\, n)$, then

$$K(x) \leqslant m(n) + K(x|\mathtt{gr}\, n) = m(n) + K_{\lim}(x).$$

Hence $K(x) - K_{\lim}(x) \leqslant m(n)$ and $B(K(x) - K_{\lim}(x)) \leqslant n$, as $B(k)$ is an inverse function to $m(n)$. Of course this argument is not rigorous, as we omitted some logarithmic terms.

References

[1] B. Durand, A. Shen, and N. Vereshchagin. "Descriptive Complexity of Computable Sequences". In C. Meinel and S. Tison (Eds.): Proc. of 16th Ann. Symp. on Theoretical Aspects of Computer Science, Trier, Germany, March 1999, LNCS 1563, pp. 153–162.

[2] P. Gács. "On the relation between descriptional complexity and algorithmic probability," *Theor. Comp. Scie.* **22** (1983) 71–93.

[3] T. Kamae. "On Kolmogorov's complexity and information," *Osaka J. Math.* **10** (1973) 305–307.

[4] M. Li, P. Vitányi. An Introduction to Kolmogorov Complexity and its Applications. Springer Verlag, 1997.

[5] An.A. Muchnik. "Lower limits on frequencies in computable sequences and relativized a priori probability," *SIAM Theory Probab. Appl.,* **32** (1987) 513–514.

[6] T. Rado. "On a simple source for non-computable functions," *Proc. Symposium Mathematical Theory of Automata,* MRI Symposium Series **XII** (Polytechnic Press, Brooklyn, 1962) 75–81.

[7] V. Uspensky and A. Semenov. Algorithms: Main Ideas and Applications. Kluwer Acad. Publ., 1993, 269 p.

[8] A.K. Zvonkin and L.A. Levin. "The complexity of finite objects and the development of the concept of information and randomness by means of the theory of algorithms," *Russian Math. Surveys,* **25(6)** (1970), 83–124.

ECCC ISSN 1433-8092
http://www.eccc.uni-trier.de/eccc
ftp://ftp.eccc.uni-trier.de/pub/eccc
ftpmail@ftp.eccc.uni-trier.de, subject 'help eccc'