Electronic Colloquium on Computational Complexity, Report No. 35 (2000)

Independent minimum length programs to translate between given strings

Nikolai K. Vereshchagin
Moscow State University
ver@mccme.ru*

Michael V. Vyugin
Moscow State University
misha@vyugin.mccme.ru†

Abstract

A string p is called a program to compute y given x if $U(p,x) = y$, where U denotes universal programming language. Kolmogorov complexity $K(y|x)$ of y relative to x is defined as minimum length of a program to compute y given x. Let $K(x)$ denote $K(x|\text{empty string})$ (Kolmogorov complexity of x) and let $I(x:y) = K(x) + K(y) - K(\langle x,y\rangle)$ (the amount of mutual information in x,y). In the present paper we answer in negative the following question posed in [1]: Is it true that for any strings x,y there are independent minimum length programs p,q to translate between x,y, that is, is it true that for any x,y there are p,q such that $U(p,x) = y$, $U(q,y) = x$, the length of p is $K(y|x)$, the length of q is $K(x|y)$, and $I(p:q) = 0$ (where the last three equalities hold up to an additive $O(\log(K(x|y) + K(y|x)))$ term)?

1 Introduction

We write *string* to denote a finite binary string. Other finite objects, such as pairs of strings, may be encoded into strings in natural ways. The set of all strings is denoted by $\{0,1\}^*$ and the length of a string x is denoted by $l(x)$,

The Kolmogorov complexity, $K(x)$ of a string x is the length of a shortest program to compute x. We assume that some universal programming language U is fixed and we write $U(p,x) = y$ if the program p on input x computes y. Intuitively, $K(x)$ represents the minimal amount of information to generate x by an effective process. The conditional Kolmogorov complexity, $K(x|y)$ of x relative to y is the length of the shortest program that computes x given y as an input. The *mutual information* between x and y is the quantity

$$I(x:y) = K(x) + K(y) - K(\langle x,y\rangle).$$

Following [1] we define information distance between x,y as $d(x,y) = \max\{K(x|y), K(y|x)\}$. Another natural notion of information distance between x,y is the length of the shortest program that given x computes y and, conversely, given y computes x. Let $d_1(x,y)$ denote this value. Obviously,

$$d(x,y) \leqslant d_1(x,y) \leqslant K(x|y) + K(y|x).$$

One may expect that in the worst case we have $d_1(x,y) = K(x|y) + K(y|x)$. However, one of the results of [1] states that for any x,y it holds $d_1(x,y) = d(x,y) + O(\log d(x,y))$. This assertion is a direct corollary of the following theorem.

Theorem 1 (Conversion theorem [1]). *Let $K(y|x) \geqslant K(x|y)$. There is a string d of length $K(y|x) - K(x|y)$ and a program r of length $K(x|y) + 2K(\langle K(x|y), K(y|x)\rangle) + O(1)$ that given xd computes y and conversely, given y computes xd.*

*Dept. of Mathematical Logic and Theory of Algorithms, Moscow State University, Vorobjevy Gory, Moscow 119899, Russia. The work was done while visiting L.I.P., Ecole Normale Supérieure of Lyon.

†Dept. of Mathematical Logic and Theory of Algorithms, Moscow State University, Vorobjevy Gory, Moscow 119899, Russia.

ISSN 1433-8092

This theorem implies that there are minimum length programs p, q to translate between x, y that overlap as much as possible. Indeed, given $p = \langle r, l(d)\rangle$ and y we can compute x and given $q = \langle r, d\rangle$ and x we can compute y.

The opposite question is whether p and q can always be made completely *independent*, that is, $I(p : q) = 0$. That is, is it true that for every x, y there are p, q such that $U(p, x) = y$, $U(q, y) = x$ and $l(p) = K(y|x)$, $l(q) = K(x|y)$, $I(p : q) = 0$ (where the last three equalities hold up to an additive $O(\log d(x, y))$ term)? This question was posed in [1]. In the present paper we answer it in negative. Moreover, we show that there are x, y for which any minimum length p, q to translate between x and y have maximum overlap, that is $K(p, q) = d(x, y)$ (Corollary 8 in Section 5).

However, if we require the equalities $l(p) = K(y|x)$, $l(q) = K(x|y)$, $I(p : q) = 0$ to hold up to an additive $O(\log K(\langle x, y\rangle))$ term (and not up to an additive $O(\log d(x, y))$ term), the situation changes drastically. In this case, there exist always independent p, q (Section 3).

We study also two related questions: 1) For which pairs (m, l) $(m \geqslant l)$ for any x of complexity m there is a random string z of length l such that $K(z|x) = O(\log l)$? We prove that this is true if m is either small or very large compared to l. For intermediate values of m such z may not exist (Section 4). 2) Is it true that for any x, y there is minimum length p to translate from y to x that is simple relative to x? We obtain a negative answer to this question (Section 5).

Acknowledgements. We are sincerely grateful to An. Muchnik for suggesting a new proof of Theorem 7, which is used in the present paper, and to A. Shen for writing Section 4.

2 Preliminaries

The length of a string x is denoted by $l(x)$, Λ stands for the empty string. Let $\bar{x}$ denote the string

$$\underbrace{000\ldots0}_{l(x)\text{ times}}1x = 0^{l(x)}1x.$$

We shall use the string $\bar{x}y$ to encode the pair (x, y); the notation $\langle x, y\rangle$ will mean the same as $\bar{x}y$. Let $\log n$ denote the binary logarithm of n.

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 U is called *universal* if for any other programming language F there exists a string t_F such that $U(t_F p, x) = F(p, x)$ for any p, x. By Solomonoff – Kolmogorov theorem (see e.g. [2]) universal programming languages exist. We fix some universal programming language U and define

- $K(x|y) = \min\{l(p) \mid U(p, y) = x\}$ (conditional Kolmogorov complexity of x relative to y),
- $K(x) = K(x|\Lambda)$ (Kolmogorov complexity of x),
- $I(x : y) = K(x) + K(y) - K(\langle x, y\rangle)$ (mutual information between x and y).

If $U(p, y) = x$, we say that p computes x given y. If $U(p, \Lambda) = x$, we say that p computes x.

Instead of $K(\langle x, y\rangle)$ and $K(z|\langle x, y\rangle)$ we shall write $K(x, y)$ and $K(z|x, y)$ respectively. We use the following well known facts (see [2]):

- $K(x) \leqslant l(x) + O(1)$;
- $K(x|y) \leqslant K(x) + O(1)$;
- 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;
- there is a constant c such that $K(x, y) \leqslant K(x) + K(y|x) + 2\log K(y|x) + c$, $K(x, y) \leqslant K(x) + K(y|x) + 2\log K(x) + c$ for all x, y (note that the terms $2\log K(y|x)$, $2\log K(x)$ appear since we have to encode a pair of programs by one program);
- $K(x, y) \geqslant K(x) + K(y|x) - O(\log K(x, y))$.
- $I(x : y) \geqslant -O(\log K(x|y))$, $I(x : y) \geqslant -O(\log K(y|x))$.

- Strings x and y are called *independent* if $I(x:y)$ is close to 0. A string x is called *random*, or *incompressible*, if $K(x)$ is close to $l(x)$.
- If x is incompressible then so is any prefix x' of x $(l(x') - K(x') \leqslant l(x) - K(x) + 2\log l(x') + O(1))$.
- If p is a minimum length program to compute y given x, then p is incompressible $(K(p) = l(p) + O(1) = K(y|x) + O(1))$.

3 Minimum overlap up to an additive $O(\log K(x,y))$ term

Theorem 2. *For any strings x, y there are p, q such that $U(p,x) = y$, $U(q,y) = x$, $l(p) = K(y|x)$, $l(q) = K(x|y)$, and $I(p:q) = 0$ (where the last three equalities hold up to an additive $O(\log K(x,y))$ term).*

Proof. To demonstrate the idea let us present the sketch of the proof assuming $K(y|x) = K(x|y) = k$. By Conversion theorem there is a program r of length about k to translate from x to y and vice versa (that is, $U(r,x) = y, U(r,y) = x$). Assume first that x is incompressible, that is, $K(x)$ is close to $l(x)$. Let x' consist of the first k bits of x (note that the length of x is greater than k, since $k = K(x|y) \leqslant K(x) \leqslant l(x)$; we omit additive $O(1)$ terms and even $O(\log K(x,y))$ terms). Let $p = r \oplus x'$ (bitwise sum modulo 2) and $q = r$. Then p may be used to find y given x as well as r: find r as $p \oplus x'$ and use r to find y given x. It remains to show that strings $r \oplus x'$ and r are independent, that is $K(r \oplus x', r)$ is close to $K(r \oplus x') + K(r) \leqslant 2k$. We have $K(r \oplus x', r) = K(x', r)$. But x' and r are independent, as so are x and r (recall that r is a minimum length program to compute y given x). Therefore $K(x', r) = K(x') + K(r) = k + k = 2k$ (since x is incompressible, so is x', thus $K(x') = l(x') = k$). Note that p is not a program to compute y given x in formal sense. We know only that $K(y|x,p) = O(1)$. We will fix this later.

Recall that we assumed that x is incompressible. If x is compressible, let x'' be the minimum length program to compute x that halts in fewest number of steps (if there are several programs that halt in fewest number of steps then take the first one in the lexicographical order). Then x'' is incompressible, and equivalent to x in the following sense: $K(x|x'') = O(1)$, $K(x''|x) = \log K(x) + O(1)$ (given $K(x)$ and x we can find x'' by running all programs of length $K(x)$). As our equalities hold up to additive $O(\log K(x,y))$ term, we can replace x by x'' in the above arguments.

Consider now the general case. Without loss of generality assume that $K(x|y) \leqslant K(y|x)$. Then the above argument is modified as follows. Let r, d be strings which exist by Conversion theorem. That is, $l(d) = K(y|x) - K(x|y)$, $l(r) = K(x|y) + O(\log K(y|x))$, $U(r, xd) = y$, and $U(r,y) = xd$. Let x'' be defined as earlier and let x' be the prefix of x'' of length $K(x|y)$. (It may happen that the length of x'' is less than $K(x|y)$. In this case $K(x|y) - l(x'') = O(1)$, as $l(x'') = K(x) \geqslant K(x|y) - O(1)$ and we let $x' = x''$.) Let $p = (x' \oplus r)d$ and $q = r$. Obviosly p, q have proper lengths. And p, q are independent, as $K(p,q) = K(x', r, d)$ and x', r, d are independent, hence $K(p,q) = K(x') + K(r) + K(d) = K(x|y) + K(x|y) + (K(y|x) - K(x|y)) = K(x|y) + K(y|x) = K(q) + K(p)$.

Now we have to convert p, q in real programs to translate between x and y. Given $x' \oplus r$, d, x, and $K(x)$ we can find y (first run in parallel all programs of length $K(x)$ to find x'', then compute x' and r and apply U to r, xd). On the other hand, given r, $l(d)$, and y we can find x. Therefore, there are computable functions f, g such that

$$f(\overline{l(d)}r, y) = x, \qquad g(\overline{K(x)l(d)}(x' \oplus r)d, x) = y.$$

By universality of U there are strings t_f, t_g such that

$$U(t_f\overline{l(d)}r, y) = x,$$
$$U(t_g\overline{K(x)l(d)}(x' \oplus r)d, x) = y.$$

Thus let

$$q = t_f\overline{l(d)}r, \qquad p = t_g\overline{K(x)l(d)}(x' \oplus r)d.$$

We have

$$\begin{aligned} l(q) &= l(r) = K(x|y), \\ l(p) &= l(x' \oplus r) + l(d) \\ &= K(x|y) + (K(y|x) - K(x|y)) = K(y|x) \end{aligned}$$

(recall that we omit additive $O(\log K(x,y))$ terms). So it remains to prove that p and q are independent. We have

$$K(p,q) = K(x' \oplus r, r, d) = K(x', r, d).$$

and

$$\begin{aligned} & K(x) + K(y|x) = K(x,y) \\ \leqslant & K(x', r, d) + K(x|x') \\ \leqslant & K(x', r, d) + K(x) - l(x') \\ \Rightarrow & K(x', r, d) \geqslant K(y|x) + l(x') = K(y|x) + K(x|y). \end{aligned}$$

Therefore we have

$$\begin{aligned} I(p:q) &= K(p) + K(q) - K(p,q) \\ &\leqslant l(p) + l(q) - (K(y|x) + K(x|y)) = 0. \end{aligned}$$ □

We shall show that the statement of the theorem becomes false if we require the equalities $l(p) = K(y|x)$, $l(q) = K(x|y)$, and $I(p:q) = 0$ hold up to an additive $O(\log d(x,y))$ term. To do this let us look why the above argument does not prove this. There are two reasons for that. First we have $l(p) \geqslant K(y|x) + 2\log K(x)$. Second we used the inequality that does not hold up to an additive $O(\log d(x,y))$ term (at least we do not know it to hold up to an additive $O(\log d(x,y))$ term): $K(x)+K(y|x) \leqslant K(x,y)$.

To overcome the first obstacle we have to show that there is a string x' of length $K(x|y) + O(\log d(x,y))$ such that $K(x|x') \leqslant K(x) - K(x|y) + O(\log d(x,y))$ and $K(x'|x) = O(\log d(x,y))$. Note that the requirements on x' imply that x' is random: $K(x') \geqslant K(x) - K(x|x') - O(\log d(x,y)) \geqslant K(x|y) - O(\log d(x,y))$. In the next section we prove that sometimes such x' does not exist: there are strings x for which any x' that is simple conditional to x is unconditionally simple and hence is not random. It turns out that a modification of the arguments used in the construction of such x proves that it is impossible to strengthen Theorem 2 by requiring the equalities $l(p) = K(y|x)$, $l(q) = K(x|y)$, and $I(p:q) = 0$ to hold up to an additive $O(\log d(x,y))$ term (we prove this in Section 5).

4 Condensing information

Let x be an arbitrary string. Trying to "compress" it, we may consider the shortest program x' to compute x. Then $K(x|x') = O(1)$ and $K(x'|x) = O(\log K(x))$. (Indeed, x can be reconstructed from x' without additional information; to get x' from x it is enough to know the length of x', i.e., $K(x)$ which requires $O(\log K(x))$ bits.) It is easy to see also that x' is incompressible ("random"), its complexity is close to its length: $K(x') = l(x') + O(1)$. Therefore, for each x there exists a random string x' such that $K(x'|x)$ is small and length of x' is $K(x)$.

If we need a shorter random string that is simple relative to x, we can take a prefix of x' and come to the following

Proposition 3. *For any string x and for any $l \leqslant K(x)$ there is a string z of length l such that $|K(z) - l| \leqslant 2\log l + O(1)$ and $K(z|x) \leqslant \log K(x) + 2\log l + O(1)$.*

However, the bound for $K(z|x)$ in this proposition is rather weak (especially if $K(x)$ is much bigger than l); we would prefer to have $O(\log l)$ instead. Let us fix some constant d. We want to know for which pairs (m,l) the following holds: for any x of complexity m there exists z of length l such that $K(z|x) < d\log l$ and $K(z) > l - d\log l$.

Proposition 3 shows that the pair (m,l) has this property if m is bounded by a polynomial in l. (More precisely, if $d > 2$ we need $m = O(l^{d-2})$ to guarantee that $K(z|x)$ is bounded by $d\log l$.)

Such z exists also for very big m. Let m be so large that the equality $K(x) = m$ implies that x is bigger than the maximum time to compute $U(p,\Lambda)$ for all p of length at most l (we identify here x with its number in some computable enumeration of strings). Then given x and l we can find the list of all strings having complexity less than l and take any string z of length l that is not in the list.

Theorem 4 says that for intermediate values of m the existence of z is not guaranteed. The theorem shows that for some m (for some x) any string that is simple relative to x is unconditionally simple and therefore cannot be random.

Theorem 4. *There exists a constant c such that for any k, l, m there is a string x with the following properties:*

$$m \leqslant K(x), \qquad l(x) = m + 2^k(l+1)$$

and each z with $l(z) < l$, $K(z|x) < k$ is unconditionally simple: $K(z) < k + 2K(m, l|k) + c$.

Proof. We are looking for a string x of length $N = m + 2^k(l+1)$ such that all k-simple (relative to x) strings z of length less than l are (unconditionally) simple.

Consider a bipartite graph whose left-side vertices are strings of length less than l and right-side vertices are strings of length N. String z on the left is adjacent to x on the right if $K(z|x) < k$. Each vertex x on the right has degree less than 2^k.

It is convenient to use the following metaphor: Adversary enumerates edges, i.e., pairs (z, x) such that $K(z|x) < k$. We may declare some z as "simple", assigning short encoding for it. Of course, only limited number of z's could be declared as simple (this number will be 2^k).

Our goal is to guarantee that some x is "good" in the following sense: all z's declared by Adversary as simple relative to x are declared by us as (unconditionally) simple. To prove the theorem, we need many good strings x: if we have 2^m of them, then some good x has complexity at least m (as required by the statement of theorem).

Let us describe our strategy. At the first step we wait until at least $2^N - 2^m$ right-side vertices are connected to some left-side vertex. (If this never happens, we are done, because remaining 2^m vertices are good.) After that we choose string z_1 of length less than l that is adjacent to at least $(2^N - 2^m)/2^l$ different strings x.

After z_1 appears, we declare it as "simple", and consider only right-side vertices adjacent to z_1 at this moment and call them 1-vertices. We have at least $(2^N - 2^m)/2^l > 2^{N-(l+1)}$ 1-vertices. Now we wait until at least $2^{N-(l+1)} - 2^m$ 1-vertices are connected (by Adversary) to some left-side vertex different from z_1. Then we take some vertex $z_2 \neq z_1$ that is connected with at least $(2^{N-(l+1)} - 2^m)/2^l$ 1-vertices.

After that we declare z_2 as simple and consider only vertices adjacent to z_2, calling them 2-vertices. There are at least $(2^{N-(l+1)} - 2^m)/2^l > 2^{N-2(l+1)}$ 2-vertices.

After this procedure is used s times, vertices $z_1, \ldots, z_s$ are declared as simple and there are at least $2^{N-s(l+1)}$ s-vertices (we assume that $N - s(l+1) \geqslant 0$); each s-vertex is adjacent to $z_1, \ldots, z_s$. But it is possible that for some some s there are less than $2^{N-s(l+1)} - 2^m$ s-vertices that are adjacent to something except $z_1, \ldots, z_s$. In this case we are done, as we have at least 2^m good vertices. It remains to note that since $N = m + 2^k(l+1)$, then this must happen for some $s < 2^k$, otherwise we get a 2^k-vertex connected to $z_1, \ldots, z_{2^k}$ which contradicts our assumption.

It remains to explain why we have upper bound $K(z) < k + 2K(m, l|k) + c$ for any z declared as simple. Indeed, the process described above is an algorithmic process determined by m, k, l; the only additional information we need to specify z_i is the value of i which does not exceed 2^k and thus may be specified by a string of length exactly k. Given that string we find both i and k and then find m, l applying to k a program to compute m, l given k. (Factor 2 is needed for pairs encoding: $K(u, v) \leqslant K(u) + 2K(v) + O(1)$.) □

Recall the point we started from. We wanted to show that for intermediate values of m there are x's with $K(x) = m$ for which there are no random z that are simple relative to x. To make this goal precise fix d and call a string of length l *random* if $K(x) > l - d \log l$. Call a string *simple relative to x* if $K(z|x) < d \log l$. Letting in the above theorem $k = d \log l$ we obtain

Corollary 5. *There exists a constant c such that for any d, m and l with $K(m, l, d) \leqslant l/2 - d \log l - c$ there exists a word x such that $m \leqslant K(x) \leqslant m + l^d(l+2) + c$ and there is no random z of length l that is simple relative to x.*

5 Minimum overlap up to an additive $O(\log d(x,y))$ term

Recall that the question on minimum overlap (up to an additive $O(\log d(x,y))$ term) asks: Is it true that for every x, y there are p, q such that $U(p,x) = y$, $U(q,y) = x$ and $l(p) = K(y|x)$, $l(q) = K(x|y)$, $I(p:q) = 0$ (where the last three inequalities hold up to an additive $O(\log d(x,y))$ term)? We first obtain, as a direct corollary of Theorem 4, a negative answer to a stronger version of this question whether for every x, y there is p to transform y to x that is simple relative to x. That is, is it true that for every x, y there is p such that $U(p,x) = y$, $K(p|y) = 0$, $l(p) = K(y|x)$ (where the last two equalities hold up to an additive $O(\log d(x,y))$ term)? Note that a positive answer to the latter question would imply a positive answer to the former one. Indeed, assume that for any x, y such p exists. Then for any q with $K(x|q,y) = 0$ and $l(q) = K(x|y)$ we have

$$\begin{aligned} l(q) = K(x|y) &\leqslant K(p|y) + K(q|p) + K(x|q,y) \\ &\leqslant K(q|p) \leqslant l(q), \\ I(p:q) &= K(q) + K(p) - K(p,q) \\ &= K(q) - K(q|p) \leqslant l(q) - K(q|p) = 0 \end{aligned}$$

(all inequalities hold up to an additive $O(\log d(x,y))$ term). That is, p is independent of *every* minimum length program to transform y to x.

The following theorem shows that for some x, y there is no such p.

Theorem 6. *There exists a constant c such that for any $d \geqslant 1$ and n the following holds: there exist strings x and y such that*

$$\begin{aligned} l(x) &\leqslant n^d(n + d\log n + 2), \\ K(x|y) &\leqslant n + c, \qquad (1) \\ K(y|x) &\leqslant d\log n + \log(n + d\log n) + c \qquad (2) \end{aligned}$$

and there is no p of length less than $n + d\log n$ such that

$$\begin{aligned} K(p|x) &< d\log n, \\ K(x|p,y) &< n - (d+4)\log n - 3\log d - O(1). \end{aligned}$$

Proof. Let in Theorem 4 $k = d\log n$, $l = n + d\log n$, and $m = n$. Let x a string which exists by Theorem 4. There is a prefix y of x satisfying (1), (2) and the inequality $K(x|y) \geqslant n$. Indeed, $K(x|\Lambda) = K(x) \geqslant n$ and $K(x|x) = O(1)$. When we remove the last letter from y the complexity of y changes by $O(1)$, therefore, there is a prefix y of x such that $n \leqslant K(x|y) \leqslant n + O(1)$. Obviously, $K(y|x) \leqslant \log l(x) + O(1)$, the inequality (2) follows.

By Theorem 4 for any p of length less than $n + d\log n$ with $K(p|x) < d\log n$ we have

$$\begin{aligned} K(p) &< d\log n + 2K(n) + 2K(d) + O(1) \\ &< (d+2)\log n + 2\log d + O(1) \\ n \leqslant K(x|y) &\leqslant K(x|p,y) + K(p) + 2\log K(p) + O(1) \\ &\leqslant K(x|p,y) + (d+2)\log n + 2\log d \\ &\quad + 2\log K(p) + O(1) \\ &\leqslant K(x|p,y) + (d+4)\log n + 3\log d + O(1). \quad \square \end{aligned}$$

Let us show now that minimum overlap is impossible for some x, y (up to an additive $O(\log d(x,y))$ term).

Theorem 7. *For any integer n, f and g there are strings x, y such that $n \leqslant K(x|y), K(y|x) \leqslant n + 2K(f,g,K(n,f,g)|n) + O(1)$ and the the following holds. For any p, q of lengths less than $n + f$ such that $K(y|p,x) < g$, $K(x|q,y) < g$ it holds*

$$K(p,q) \leqslant n + 2f + 2g + 7K(n,f,g) + O(1).$$

The lengths of x, y are $(n+f)2^{n+2f+2g+5K(n,f,g)+O(1)}$.

Proof. Our plan is as follows. Theorem 4 shows that for any k, m there is x such with high complexity ($K(x) > m$) such that any z that is simple conditional to x ($K(z|x) < k$) is unconditionally simple ($K(z) < k$; we neglect additive terms of order $O(K(n,f,g))$). Apply Theorem 4 for $k = n+2f+2g$, $m = n$, $l = n + 2f$. Assume first that there is y such that both relative complexities $K(x|y), K(y|x)$ are close to n. Then x, y satisfy the statement of the theorem. Why? Let p, q be as in the condition of the theorem, that is, $l(p), l(q) < n+f$, $K(y|p,x), K(x|q,y) < g$. We have to show that $K(p,q) < n + 2f + 2g$. By the choice of x it suffices to prove that $K(p,q|x) <$

$n + 2f + 2g$. To prove the latter fact it suffices to show that $K(q|p,x) < f + 2g$, or (recalling that $K(y|p,x) < g$) to show that $K(q|x,y) < f + g$. To prove the inequality $K(q|x,y) < f + g$ consider the set $Q = \{q' \mid l(q') < n + f, K(x|y,q') < g\}$. It has at most 2^{f+g} elements. Indeed, the number of x' such that the set $Q_{x'} = \{q' \mid l(q') < n+f, K(x'|y,q') < g\}$ has significantly more than 2^{f+g} elements is significantly less than 2^n (it is less than the number of pairs $\langle q, x'\rangle$ such that $l(q') < n + f$, $K(x'|y,q') < g$ (2^{n+f+g}) divided by 2^{f+g}). All such x' are enumerable given y, hence their complexity relative to y is significantly less than n. As $K(x|y)$ is close to n we conclude that $|Q| \leqslant 2^{f+g}$. Since Q is enumerable given x, y and $q \in Q$, we derive that $K(q|x,y) \leqslant f + g$ (to enumerate Q we need also $n + f$ and g; these numbers can be specified by $K(n,f,g)$ bits).

To finish this argument we need the following assertion

- for any n with $K(x) \geqslant n$ there is y such that both relative complexities $K(x|y), K(y|x)$ are close to n (say $K(x|y), K(y|x) = n + O(\log n)$).

This assertion is true, however the proof is rather complicated (it will be published elsewhere). We will not use it, instead we will modify the argument of Theorem 4 to construct x as in Theorem 4 together with y such that $K(x|y), K(y|x) = n + O(\log n)$.

So we start with the following

Lemma 1. *For any parameters l, k and n ($k \geqslant n$) there exist strings x and y such that*

$$l(x) = l(y) = 2^k(2l+1),$$
$$n \leqslant K(x|y), K(y|x) \leqslant n + 2K(l,k|n) + O(1),$$

and for any z of length less than l such that $K(z|x) < k$ it holds $K(z) < k + 2K(l,n|k) + O(1)$.

Proof. Modify the proof of Theorem 4 as follows. Strings of length less than l are considered as left-side nodes and strings of length $N = 2^k(2l+1)$ as right-side nodes. The Adversary draws a directed edge from a right-side node x to a left-side node z if $K(z|x) < k$ and draws a directed edge from a right-side node x to another right-side node y if $K(y|x) < n$. We may declare at most 2^k left-side nodes as simple. And we may connect two right-side nodes by an (undirected) edge so that any node is connected with at most 2^n right-side nodes.

Our goal is to guarantee that some pair (x,y) of right-side nodes is "good" in the following sense: 1) we have connected x and y, 2) Adversary has not connected x and y (by a directed edge), and 3) all left-side nodes connected to x have been declared as simple.

Let us describe our strategy. Arrange all right-side vertices into pairs $(x_1,y_1),\ldots,(x_K,y_K)$ (where $K = 2^{N-1}$). For all $i \leqslant K$ connect x_i to y_i. Call all right-side nodes 0-vertices, and call all pairs $(x_1,y_1),\ldots,(x_K,y_K)$ 0-pairs. Repeat for $s = 1,2,\ldots,2^k$ the following.

1. Wait until for any $(s-1)$-pair (x_i,y_i) at least one of the conditions holds: Adversary has connected x_i and y_i, or Adversary has connected both x_i and y_i to some left-side nodes that has not been declared as simple. (If for some x_i, y_i neither of two conditions holds then either (x_i,y_i) or (y_i,x_i) is good and we are done.)

2. By Pigeon Hole Principle we can choose left-side nodes u, v that have not been declared as simple such that for a fraction at least 2^{-2l} of $(s-1)$-pairs either x_i is connected to u and y_i is connected to v, or Adversary has connected x_i to y_i. Remove all other $(s-1)$-pairs and declare both u and v simple. If there is a directed edge from x_i to y_i remove y_i and call x_i *alone*. If there is a directed edge from y_i to x_i remove x_i and call y_i *alone*. Call all the remaining nodes s-vertices.

3. Arrange all alone strings in pairs and connect each alone string to the other string from the same pair. Call remaining pairs s-pairs.

On step s we have at least $2^{N-1-s(2l+1)} \geqslant 1$ s-pairs and from each s-vertex at least s directed edges go to nodes that are not s-vertices. But it is possible that for some some s we will wait infinitely long in item 1. In this case we are done, as some pair is good. And for some $s < 2^k$ this should happen, as otherwise we get a 2^k-vertex connected by Adversary to 2^k different nodes.

Let (x,y) be any good pair. Note that we connected a right-side node to a new node only when so did Adversary. Therefore we have connected every node

to at most 2^n nodes. As our strategy is computable, this implies

$$K(x|y), K(y|x) \leqslant n + 2K(l, k|n) + O(1).$$

Any right-side node connected to x has been declared as simple and we have declared no more than $2^k - 1$ nodes as simple. Therefore we have

$$K(z) \leqslant k + 2K(l, n|k) + O(1)$$

for any z with $l(z) < l$, $K(z|x) < k$. Item 2) from the definition of a good pair implies that $K(x|y), K(y|x) \geqslant n$. □

According to our plan we apply this lemma for $l = 2(n+f)+1$ and for some k being close to $n+2f+2g$. To find the exact value of k we have to compute an upper bound for $K(q|x, y)$ more accurately (we used $f + g$ as such bound).

Lemma 2. *There is a constant c such that for any x, y, q, n, f, g with $K(x|y) \geqslant n$, $l(q) < n + f$, $K(x|q, y) < g$ it holds $K(q|x, y, n, f, g) < f + g + 3K(n, f, g) + c$.*

Proof. Let c' be a constant to be specified later. Let X be the set of all x' such that the set

$$Q_{x'} = \{q \mid l(q) < n + f, K(x'|y, q) < g\}$$

has more than $2^{f+g+3K(n,f,g)+c'}$ elements. Then

$$|X| \leqslant 2^{n+f+g}/2^{f+g+3K(n,f,g)+c'} = 2^{n-3K(n,f,g)-c'}.$$

The set X is enumerable given y, n, f, g, $K(n, f, g)$ and c'. Therefore complexity of all $x' \in X$ relative to y is less than

$$(n - 3K(n, f, g) - c') + 2K(n, f, g) + 2\log K(n, f, g) + 2\log c' + c'' \leqslant n - c' + 2\log c' + c'' < n$$

(we choose c' so large that $c' > 2\log c' + c''$). As $K(x|y) \geqslant n$, we have $|Q_x| \leqslant 2^{f+g+3K(n,f,g)+c'}$. As Q_x is enumerable given x, y, n, f, and g, we have

$$K(q|x, y, n, f, g) < f + g + 3K(n, f, g) + c$$

for any such q. □

Assume that $l(p), l(q) < n + f$ and $K(y|x, p) < g$, $K(x|y, q) < g$. To apply the statement of Lemma 1 to $z = (p, q)$ we have to encode the pair $\langle p, q\rangle$ in less than $l = 2(n + f) + 1$ bits. Let $[p, q] = 0^{n+f-l(p)-1}1p0^{n+f-l(q)-1}1q$ be such encoding.

Assume that $K(x|y) \geqslant n$. To find $[p, q]$ given x, n, f, g it suffices to know: 1) p, which has less than $n + f$ bits, 2) a string r of length less than g that identifies y given p, x, and 3) a string s of length less than $f + g + 3K(n, f, g) + c$ that identifies q given x, y, n, f, g (such s exists by Lemma 2). This information can be encoded into string

$$0^{n+f-l(p)-1}10^{g-l(r)-1}1prs$$

of length $(n + f) + g + (f + g + 3K(n, f, g) + c) = n + 2f + 2g + 3K(n, f, g) + c$, which identifies p, r, s provided n, f, g are known. Hence

$$\begin{aligned} K([p, q]|x, n, f, g) &< n + 2f + 2g + 3K(n, f, g) + c', \\ K([p, q]|x) &< n + 2f + 2g + 5K(n, f, g) + c''. \end{aligned}$$

Thus we see that we can apply Lemma 1 for $l = 2(n + f) + 1$ and $k = n + 2f + 2g + 5K(n, f, g) + c''$. The pair (x, y) which exists by Lemma 1 satisfies our theorem. Indeed, for any strings p, q with $l(p), l(q) < n + f$ and $K(y|x, p) < g$ and $K(x|y, q) < g$ we have $K([p, q]|x) < k$ hence

$$\begin{aligned} K([p, q]) &< k + 2K(l, n|k) + O(1) \\ &< n + 2f + 2g + 5K(n, f, g) \\ &\quad + 2K(l, n|k) + O(1) \\ &< n + 2f + 2g + 5K(n, f, g) \\ &\quad + 2K(n, f, g) + O(1). \end{aligned}$$

The last inequality holds, as given f, n, g and k we can find $K(f, n, g)$ (recall that $k = n + 2f + 2g + 5K(n, f, g) + c''$). Therefore,

$$\begin{aligned} K(p, q) &\leqslant K([p, q]) + O(1) \\ &\leqslant n + 2f + 2g + 7K(n, f, g) + O(1). \end{aligned}$$

It remains to note that

$$\begin{aligned} n &\leqslant K(x|y), K(y|x) \leqslant n + 2K(l, k|n) + O(1) \\ &\leqslant n + 2K(f, g, K(n, f, g)|n) + O(1). \quad \square \end{aligned}$$

Corollary 8. *For any d there is c such that for all n the following holds. There are x, y such that $n \leqslant K(x|y), K(y|x) < n + 2\log\log n + c$ and for any p, q of lengths less than $n + d\log n$ with $U(p, x) = y$, $U(q, y) = x$ it holds*

$$K(p, q) \leqslant n + (2d + 7)\log n + c.$$

Proof. Let in the above theorem $f = d\log n$, $g = O(1)$ and note that

$$\begin{aligned} &K(n, f, g) = K(n) + O(1) \leqslant \log n + O(1), \\ &K(f, g, K(n, f, g)|n) = K(K(n, f, g)|n)) + O(1) \\ &\leqslant \log K(n, f, g) + O(1) \leqslant \log\log n + O(1) \end{aligned}$$

(the constants $O(1)$ depend on d). □

References

[1] C.H. Bennett, P. Gács, M. Li, P.M.B. Vitányi, and W.H. Zurek. "Information Distance", IEEE Trans. on Information Theory **44** (1998), No 4, 1407–1423.

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

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'