Variations on Muchnik's Conditional Complexity Theorem

Daniil Musatov, Andrei Romashchenko, Alexander Shen

Variations on Muchnik's Conditional Complexity Theorem*

Daniil Musatov[1], Andrei Romashchenko[2], Alexander Shen[3]

[1] Lomonosov Moscow State University, musatych@gmail.com
[2] LIF Marseille, CNRS & Univ. Aix–Marseille, on leave from IITP RAS, andrei.romashchenko@gmail.com
[3] LIF Marseille, CNRS & Univ. Aix–Marseille, on leave from IITP RAS, alexander.shen@lif.univ-mrs.fr

Abstract. Muchnik's theorem about simple conditional descriptions states that for all strings a and b there exists a program p transforming a to b that has the least possible length and is simple conditional on b. In this paper we present two new proofs of this theorem. The first one is based on the on-line matching algorithm for bipartite graphs. The second one, based on extractors, can be generalized to prove a version of Muchnik's theorem for space-bounded Kolmogorov complexity. Another version of Muchnik's theorem is proven for a resource-bounded variant of Kolmogorov complexity based on Arthur–Merlin protocols.

1 Muchnik's Theorem

In this section we recall a result about conditional Kolmogorov complexity due to An. Muchnik [7]. By $\mathrm{C}(u)$ we denote Kolmogorov complexity of string u, i.e., the length of a shortest program generating u. The conditional complexity of u given v, the length of a shortest program that translates v to u, is denoted by $\mathrm{C}(u \mid v)$, see [4].

Theorem 1. *Let a and b be two binary strings, $\mathrm{C}(a) < n$ and $\mathrm{C}(a \mid b) < k$. Then there exists a string p such that*

- $\mathrm{C}(a \mid p, b) \le O(\log n)$;
- $\mathrm{C}(p) \le k + O(\log n)$;
- $\mathrm{C}(p \mid a) \le O(\log n)$.

This is true for all a, b, n, k, and the constants hidden in $O(\log n)$ do not depend on them.

Remarks. 1. In the second inequality we can replace complexity $\mathrm{C}(p)$ of a string p by its length $|p|$. Indeed, we can use the shortest description of p instead of p.

2. We may let $k = \mathrm{C}(a \mid b) + 1$ and replace $k + O(\log n)$ by $\mathrm{C}(a \mid b) + O(\log n)$ in the second inequality. We may also let $n = \mathrm{C}(a) + 1$.

* Supported by ANR Sycomore, NAFIT ANR-08-EMER-008-01 and RFBR 09-01-00709-a grants.

3. Finally, having $|p| \le \mathrm{C}(a \mid b) + O(\log n)$, we can delete $O(\log n)$ last bits in p, and the first and third inequalities will remain true. We come to the following reformulation of Muchnik's theorem: *for every two binary strings a and b there exist a binary string p of length at most $\mathrm{C}(a \mid b)$ such that $\mathrm{C}(a \mid p, b) \le O(\log \mathrm{C}(a))$ and $\mathrm{C}(p \mid a) \le O(\log \mathrm{C}(a))$.*

Informally, Muchnik's theorem says that there exists a program p that transforms b to a, has the minimal possible complexity $\mathrm{C}(a \mid b)$ up to a logarithmic term, and, moreover, can be easily obtained from a. The last requirement is crucial, otherwise the statement becomes a trivial reformulation of the definition of conditional Kolmogorov complexity.

This theorem is an algorithmic counterpart of Slepian–Wolf theorem [11] in multisource information theory. Assume that some person **S** knows b and wants to know a. We know a and want to send some message p to **S** that will allow **S** to reconstruct a. How long should be this message? Do we need to know b to be able to find such a message? Muchnik's theorem provides kind of a negative answer to the last question, though we still need a logarithmic advice. Indeed, the absolute minimum for a complexity of a piece of information p that together with b allows **S** to reconstruct a, is $\mathrm{C}(a \mid b)$. It is easy to see that this minimum can be achieved with logarithmic precision by a string p that has logarithmic complexity conditional on a and b. But it turns out that in fact b is not needed and we can provide p that is simple conditional on a and still does the job.

In many cases statements about Kolmogorov complexity have combinatorial counterparts, and sometimes it is easy to show the equivalence between complexity and combinatorial statements. In the present paper we study two different combinatorial objects closely related to Muchnik's theorem and its proof.

First, in Sect. 2, we define the *on-line matching* problem for bipartite graphs. We formulate some combinatorial statement about on-line matchings. This statement: (1) easily implies Muchnik's theorem and (2) can be proven using the same ideas that were used by Muchnik in his original proof, with some adjustments.

Second, in Sect. 3, following [3], we use extractors and their combinatorial properties. Based on this technique, we give a new proof of Muchnik's theorem. With this method we prove versions of this theorem for polynomial space Kolmogorov complexity and also for some very special version of polynomial time Kolmogorov complexity.

This work was presented on the CSR2009 conference in Novosibirsk, Russia on 18–23 August, 2009, and the conference version of the paper was published in CSR2009 Proceedings by Springer-Verlag. This version of the paper is slightly rearranged and extended.

2 Muchnik's Theorem and On-line Matchings

In this section we introduce a combinatorial problem that we call *on-line matching*. It can be considered as an on-line version of the classical matching problem. Then we formulate some combinatorial statement about on-line matchings and explain how it implies Muchnik's theorem. Finally, we provide a proof of this

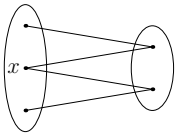
combinatorial statement, starting with the off-line version of it. This finishes the proof of Muchnik's theorem.

2.1 On-line Matchings

Consider a bipartite graph with the left part L, the right part R and a set of edges $E \subset L \times R$. Let s be some integer. We are interested in the following property of the graph:

> *for any subset L' of L of size at most s there exists a subset $E' \subset E$ that performs a bijection between L' and some $R' \subset R$.*

A necessary and sufficient condition for this property is provided by well-known Hall's theorem. It says that *for each set $L' \subset L$ of size $t \leq s$ the set of all neighbors of elements of L' contains at least t elements.*



This condition is not sufficient for the following on-line version of matching. We assume that an adversary gives us elements of L one by one, up to s elements. At each step we should provide a counterpart for each given element x, i.e., to choose some neighbor $y \in R$ not used before. This choice is final and cannot be changed later.

Providing a matching on-line, when next steps of the adversary are not known in advance, is a more subtle problem than the usual off-line matching. Now Hall's criterion, while still being necessary, is no more sufficient. For example, for the graph shown in the picture, one can find a matching for each subset of size at most 2 of the left part, but this cannot be done on-line. Indeed, we are blocked if the adversary starts with x.

Now we formulate a combinatorial statement about on-line matching; then in Sect. 2.2 we show that this property implies Muchnik's theorem, and in Sect. 2.3 we prove this property.

Combinatorial statement about on-line matchings (**OM**). *There exists a constant c such that for every integers n and k, where $k \leq n$, there exists a bipartite graph G whose left part L has size 2^n, right part R has size $2^k n^c$, each vertex in L has at most n^c neighbors in R, and for which on-line matching is possible up to size 2^k.*

Note that the size of the on-line matching is close to the size of R up to a polynomial factor, and the degrees of all L-elements are polynomially bounded, so we are close to Hall's bound.

2.2 Proof of Muchnik's theorem

First we show how (OM) implies Muchnik's theorem. We may assume without loss of generality that the *length* of the string a (instead of its complexity) is less than n. Indeed, if we replace a by a shortest program that generates a, all complexities involving a change by only $O(\log n)$ term: knowing the shortest program for a, we can get a without any additional information, and to get a

shortest program for a given a we need only to know the value of $\mathrm{C}(a)$, because we can try all programs of length $\mathrm{C}(a)$ until one of them produces a. There may exist several different shortest programs for a; we take that one which appears first when trying in parallel all programs of length $\mathrm{C}(a)$. As we have said, for similar reasons it does not matter whether we speak about $\mathrm{C}(p)$ or $|p|$ in the conclusion of the theorem. We used $\mathrm{C}(p)$ to make the statement more uniform; however, in the proof we get the bound for $|p|$ directly.

We may assume that $n \geq k$, otherwise the statement of theorem 1 is trivial (let $p = a$). Consider the graph G provided by (OM) with parameters n and k. Its left part L is interpreted as the set of all strings of length less than n; therefore, a is an element of L. Knowing b, we can enumerate all strings x of length less than n such that $\mathrm{C}(x \mid b) < k$. There exist at most 2^k such strings, and a is one of them. The property (OM) implies that it is possible to find an on-line matching for all these strings, in the order they appear during the enumeration. Let p be an element of R that corresponds to a in this matching.

Let us check that p satisfies all the conditions of Muchnik's theorem. First of all, note that the graph G can be chosen in such a way that its complexity is $O(\log n)$. Indeed, (OM) guarantees that a graph with the required properties exists. Given n and k, we can perform an exhaustive search until the first graph with these properties is found. This graph is a computable function of n and k, so its complexity does not exceed the complexity of the pair (n, k), which is $O(\log n)$.

If a is given (as well as n and k), then p can be specified by its ordinal number in the list of a-neighbors. This list contains at most n^c elements, so the ordinal number contains $O(\log n)$ bits.

To specify p without knowing a, we give the ordinal number of p in R, which is $k+O(\log n)$ bits long. Here we again need n and k, but this is another $O(\log n)$ bits.

To reconstruct a from b and p, we enumerate all strings of lengths less than n that have conditional complexity (relative to b, which is known) less than k, and find R-counterparts for them using (OM) until p appears. Then a is the L-counterpart of p in this matching.

Formally speaking, for given n and k we should fix not only a graph G but also some on-line matching procedure, and use the same procedure both for constructing p and for reconstructing a from b and p. □

2.3 On-line Matchings Exist

It remains to prove the statement (OM). Our proof follows the original Muchnik's argument adapted for the combinatorial setting.

First, let us prove a weaker statement when on-line matchings are replaced by off-line matchings. In this case the statement can be reformulated using Hall's criterion, and we get the following statement:

Off-line version of (OM). *There exists a constant c such that for any integers n and k, where $n > 1$ and $k \leq n$, there exists a bipartite graph G whose left part L is of size 2^n, the right part R is of size $2^k n^c$, each vertex in L has at*

most n^c neighbors in R and for any subset $X \subset L$ of size $t \le 2^k$ the set $N(X)$ of all neighbors of all elements of X contains at least t elements.

We prove this statement by probabilistic arguments. We choose at random (uniformly and independently) n^c neighbors for each vertex $l \in L$. In this way we obtain a (random) graph where all vertices in L have degree at most n^c; the degree can be less, as two independent choices for some vertex may coincide.

We claim that this random graph has the required property with positive probability. If it does not, there exists a set $X \subset L$ of some size $t \le 2^k$ and a set Y of size less than t such that all neighbors of all elements of X belong to Y. For fixed X and Y the probability of this event is bounded by $\left(\frac{1}{n^c}\right)^{tn^c}$ since we made tn^c independent choices (n^c times for each of t elements) and for each choice the probability to get into Y is at most $1/n^c$ (the set Y covers at most $1/n^c$ fraction of points in R).

To bound the probability of violating the required property of the graph, we multiply the bound above by the number of pairs X, Y. The set X can be chosen in at most $(2^n)^t$ different ways, since for each of t elements we have at most 2^n choices; actually the number is smaller since the order of elements does not matter. For Y we have at most $(2^k n^c)^t$ choices. Further we sum up these bounds for all $t \le 2^k$. Therefore the total bound is

$$\sum_{t=1}^{2^k} \left(\frac{1}{n^c}\right)^{tn^c} (2^n)^t \left(2^k n^c\right)^t.$$

This is a geometric series; the sum is less than 1 (which is our goal) if the base is small. The base is

$$\left(\frac{1}{n^c}\right)^{n^c} (2^n) \left(2^k n^c\right) = \frac{2^{n+k}}{n^{c(n^c-1)}}$$

and $c = 2$ makes it small enough for all $n > 1$ and $k \le n$. It even tends to zero as $n \to \infty$. Off-line version is proven. □

Now we have to prove (OM) in its original on-line version. Fix a graph G that satisfies the conditions for the off-line version for given n and k. Let us use the same graph in the on-line setting with the following straightforward "greedy" strategy. When a new element $x \in L$ arrives, we check if it has neighbors that are not used yet. If yes, one of these neighbors is chosen to be a counterpart of x. If not, x is "rejected".

Before we explain what to do with the rejected elements, let us prove that at most half of 2^k given elements could be rejected. Assume that more than 2^{k-1} elements are rejected. Then less than 2^{k-1} elements are served and therefore less than 2^{k-1} elements of R are used as counterparts. But all neighbors of all rejected elements are used; this is the only reason for rejection. So we get the contradiction with the condition $\#N(X) \ge \#X$ if X is the set of rejected elements.

Now we need to deal with rejected elements. They are forwarded to the "next layer" where the new task is to find on-line matching for 2^{k-1} elements. If we can

do this, then we combine both graphs using the same L and disjoint right parts R_1 and R_2; the elements rejected at the first layer are sent to the second one. In other terms: (n, k) on-line problem is reduced to (n, k) off-line problem and $(n, k-1)$ on-line problem. The latter can then be reduced to $(n, k-1)$ off-line and $(n, k-2)$ on-line problems etc.

Finally we get k levels. At each level we serve at least half of the requests and forward the remaining ones to the next layer. After k levels of filtering only one request can be left unserved, so one more layer is enough. Note also that we may use copies of the same graph on all layers.

More precisely, we have proven the following statement: *Let G be a bipartite graph with left side L and right side R that satisfies the conditions of the off-line version for given n and k. Replace each element in R by $(k+1)$ copies, all connected to the same elements of L as before. Then the new graph provides on-line matchings up to size 2^k.*

Note that this construction multiplies both the size of R and the degree of vertices in L by $(k+1)$, which is a polynomial in n factor. The statement (OM) is proven. □

3 Muchnik's Theorem and Extractors

In this section we present another proof of Muchnik's theorem based on the notion of extractors. This technique was first used in a similar situation in [3]. With this technique we prove some versions of Muchnik's theorem for resource-bounded Kolmogorov complexity. This result was presented in the Master Thesis of one of the authors [5].

3.1 Extractors

Let G be a bipartite graph with N vertices in the left part and M vertices in the right part. The graph may have multiple edges. Let all vertices of the left part have the same degree D. Let us fix an integer $K > 0$ and a real number $\varepsilon > 0$.

Definition 1. *A bipartite graph G is a (K, ε)-extractor if for all subsets S of its left part such that $\#S \geq K$ and for all subsets Y of the right part the inequality*

$$\left| \frac{\#E(S,Y)}{D \cdot \#S} - \frac{\#Y}{M} \right| < \varepsilon \tag{1}$$

holds, where $E(S, Y)$ stands for the set of edges between S and Y.

In the sequel we always assume that N, M, D, and sometimes other quantities denoted by uppercase letters are powers of 2, and use corresponding lowercase letters (n, m, d, etc.) to denote their logarithms. In this case the extractor may be seen as a function that maps a pair of binary strings of length $n = \log N$ (an index of a vertex on the left) and of length $d = \log D$ (an index of an edge

incident to this vertex) to a binary string of length $m = \log M$ (an index of the corresponding vertex on the right).

The extractor property may be reformulated as follows: consider a uniform distribution on a set S of left-part vertices. The probability of getting a vertex in Y by taking a random neighbor of a random vertex in S is equal to $\#E(S,Y)/(D\cdot \#S)$; this probability must be ε-close to $\#Y/M$, i.e. the probability of getting a vertex in Y by taking a random vertex in the right part.

It can be proven that for an extractor graph a similar property holds not only for uniform distributions on S, but for all distributions with min-entropy at least $k = \log K$ (this means that no element of L appears with probability greater than $1/K$). That is, an extractor extracts m almost random bits from n quasi-random bits, with min-entropy k or more, using d truly random bits. For a good extractor m should be close to $k + d$ and d should be small, as well as ε. Standard probabilistic argument shows that for all n, k and ε extractors with near-optimal parameters m and d do exist:

Theorem 2. *For all K, N, M and ε such that $1 < K \le N$, $M > 0$, $\varepsilon > 0$, there exists an (K, ε)-extractor with*

$$D = \left\lceil \max\left\{ \frac{M}{K} \cdot \frac{\ln 2}{\varepsilon^2},\ \frac{1}{\varepsilon^2}\left(\ln \frac{N}{K} + 1\right)\right\}\right\rceil .$$

So for given n and k we may choose the followings values of parameters (in logarithmic scale):

$$d = \log(n-k) + 2\log(1/\varepsilon) + O(1) \quad \text{and} \quad m = k + d - 2\log(1/\varepsilon) - O(1).$$

The proof may be found in [1]; it is also shown there that these parameters are optimal up to an additive term $O(\log(1/\varepsilon))$.

So far no explicit constructions of optimal extractors have been invented. By saying the extractor is *explicit* we mean that there exists a family of extractors for arbitrary values of n and k, other parameters are computable in time $\mathrm{poly}(n)$, and the extractor itself as a function of two arguments is computable in $\mathrm{poly}(n)$ time. All known explicit constructions are not optimal in at least one parameter: they either use too many truly random bits, or not fully extract randomness (i.e., $m \ll k + d$), or work not for all values of k. In the sequel we use the following theorem proven in [2]:

Theorem 3. *For all k, n and ε such that $1 < k \le n$ and $\varepsilon > 1/\mathrm{poly}(n)$, there exists an explicit $(2^k, \varepsilon)$-extractor with $m = k + d$ and $d = O((\log n \log\log n)^2)$.*

For the sake of brevity we use shorter and slightly weaker bound $O(\log^3 n)$ instead of $O((\log n \log\log n)^2)$ in the sequel.

3.2 The Proof of Muchnik's Theorem

Now we show how to prove Muchnik's theorem using the extractor technique. Consider an extractor with some N, K, D, M and ε. Let S be a subset of its left

part such that $\#S \le K$. We say that a right-part element is *bad for* S if it has more than $2DK/M$ neighbors in S, that is, twice more than the expected value if neighbors in the right part are chosen at random and S has maximal possible size K. We say that a left-part element is *dangerous in* S if all its neighbors are bad for S.

Lemma 1. *The number of dangerous elements in S is less than $2\varepsilon K$.*

Proof. We reproduce a simple proof from [3]. Without loss of generality we may assume that S contains exactly K elements; indeed, the sets of bad and dangerous elements can only increase when S increases.

For any graph, the fraction of bad right-part vertices is at most $1/2$, because the degree of a bad vertex is at least twice as large as the average degree. The extractor property reduces this bound from $1/2$ to ε. Indeed, let δ be the fraction of bad elements in the right part. Then the fraction of edges going to bad elements (among all edges starting at S) is at least 2δ. Due to the extractor property, the difference between these fractions should be less than ε. The inequality $\delta < \varepsilon$ follows.

Now we count dangerous elements in S. If their fraction in S is 2ε or more, then the fraction of edges going to the bad elements (among all edges leaving S) is at least 2ε. But the fraction of bad vertices is less than ε, and the difference between two fractions should be less than ε due to the extractor property. □

Now we present a new proof of Muchnik's theorem. As we have seen before, we may assume without loss of generality that the length of a is less than n. Moreover, as we have said, we may assume that conditional complexity $\mathrm{C}(a \mid b)$ equals $k-1$ (otherwise we decrease k) and that $k < n$ (otherwise the theorem is obvious, take $p = a$).

Consider an extractor with given n, k; let $d = O(\log n)$, $m = k$ and $\varepsilon = 1/n^3$; such an extractor exists due to Theorem 2. The choice of ε will become clear later. We choose an extractor whose complexity is at most $2\log n + O(1)$. It is possible, because only n and k are needed to describe such an extractor: other parameters are functions of n and k, and we can search through all bipartite graphs with given parameters in some natural order until the first extractor with required parameters is found. This search requires a very long time, so this extractor is not explicit.

Now assume that an extractor is fixed. We treat the left part of the extractor as the set of all binary strings of length less than n (including a), and the right part as the set of all binary strings of length $m = k$ (we will choose p among them). Consider the set S_b of all strings in the left part such that their complexity conditional on b is less than k; note that a belongs to this set.

We want to apply Lemma 1 to the set S_b and prove that a is not dangerous in S_b by showing that otherwise $\mathrm{C}(a \mid b)$ would be too small. So a has a neighbor p that is not bad for S_b, and this p has the required properties.

According to this plan, let us consider two cases.

Case 1. If a is not dangerous in S_b, then a has a neighbor p that is not bad for S_b. Let us show that p satisfies the claim of the theorem.

Complexity of p is at most $k + O(1)$ because its length is k.

Conditional complexity $\mathrm{C}(p \mid a)$ is logarithmic because p is a neighbor of a in the extractor and to specify p we need a description of the extractor ($2 \log n + O(1)$ bits) and the ordinal number of p among the neighbors of a ($d = \log D = O(\log n)$ bits).

As p is not bad for S_b, it has less than $2D$ neighbors in S_b. If b is known, the set S_b can be enumerated; knowing p, we select neighbors of p in this enumeration. Thus, to describe a given p and b, we need only a description of the extractor and the ordinal number of a in the enumeration of the neighbors of p in S_b, i.e., $O(\log n)$ bits in total.

Case 2. Assume that a is dangerous in S_b. Since the set S_b can be enumerated given b, the sets of all bad vertices (for S_b) and all dangerous elements in S_b can also be enumerated. Therefore, a can be specified by the string b, the extractor and the ordinal number of a in the enumeration of all dangerous elements in S_b. This ordinal number consists of $k - 3 \log n + O(1)$ bits due to the choice of ε (Lemma 1). So, the full description of a given b consists of $k - \log n + O(\log \log n)$ bits; $O(\log \log n)$ additional bits are needed for separating n, k and the ordinal number. This contradicts the assumption that $\mathrm{C}(a \mid b) = k - 1$. Thus, the second case is impossible and Muchnik's theorem is proven. □

3.3 Several Conditions and Prefix Extractors

In [7] An. Muchnik proved also the following generalization of Theorem 1:

Theorem 4. *Let a, b and c be binary strings, and let n, k and l be numbers such that $\mathrm{C}(a) < n$, $\mathrm{C}(a \mid b) < k$ and $\mathrm{C}(a \mid c) < l$. Then there exist binary strings p and q of length k and l respectively such that one of them is a prefix of the other one and all the conditional complexities $\mathrm{C}(a \mid p, b)$, $\mathrm{C}(a \mid q, c)$, $\mathrm{C}(p \mid a)$, $\mathrm{C}(q \mid a)$ are of order $O(\log n)$.*

This theorem is quite non-trivial: indeed, it says that information about a that is missing in b and c can be represented by two strings such that one is a prefix of the other, even if b and c are completely unrelated. It implies also that for every three strings a, b, c of length less than n, the minimal length of a program that transforms b to a and at the same time transforms c to a is at most $\max\{\mathrm{C}(a \mid b), \mathrm{C}(a \mid c)\} + O(\log n)$.

In fact a similar statement can be proven not only for two but for many (even for $\mathrm{poly}(n)$) conditions. For the sake of brevity we consider only the statement with two conditions.

This theorem also can be proven using extractors. Any extractor can be viewed as a function $E\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$.

Definition 2. *We say that a $(2^k, \varepsilon)$-extractor $E\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$, where $m \geq k$, is a* prefix extractor *if for every $i \leq k$ its prefix of length $m - i$, i.e., a function $E_i\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^{m-i}$ obtained by truncating i last bits, is a $(2^{k-i}, \varepsilon)$-extractor.*

By using probabilistic method the following theorem can be proven:

Theorem 5. *For all k, n, and ε such that $1 < k \le n$ and $\varepsilon > 0$, there exists a prefix $(2^k, \varepsilon)$-extractor with parameters $d = \log n + 2\log(1/\varepsilon) + O(1)$ and $m = k + d - 2\log(1/\varepsilon) - O(1)$.*

Proof: This proof is quite similar to the standard proof of Theorem 3. In that proof the probabilistic argument is used to show that a random graph has the required property with positive probability. In fact it is shown that this probability is not only positive but close to 1. Then we note that the restriction of a random graph is also a random graph, and the intersection of several events having probability close to 1 has a positive probability. Let us explain these arguments in more detail.

We want to show that a random bipartite graph with given parameters is a prefix extractor with a positive probability. First of all we note that it is enough to show that inequality (1) holds for S of size exactly K. Then this condition is true also for every bigger set S, since the uniform distribution on S is an average of the distributions on its subsets of size K. Second, it is enough to check the bound (1) only in one direction:

$$\frac{\#E(S,Y)}{D \cdot \#S} < \frac{\#Y}{M} + \varepsilon$$

for all sets S of cardinality K and for all Y. Indeed, the inequality

$$\frac{\#E(S,Y)}{D \cdot \#S} > \frac{\#Y}{M} - \varepsilon$$

follows from the previous one applied to the complement of Y: if there are too few edges from S to Y then there are too many edges from S to the complement of Y.

Now we specify the distribution on graphs. For every string of length n (a vertex of the left part) we choose at random (uniformly and independently) $D = 2^d$ strings of length m (its neighbors in the right part). Now we bound the probability of the event *a random graph is not a prefix extractor*.

If the extractor property is violated for some prefix of length $m-i$ then there exists a set S of $K/2^i$ elements from the left part and a set $Y \subset \{0,1\}^{m-i}$ of size $\alpha 2^{m-i}$ (for some $\alpha > 0$) such that the number of edges between S and Y is greater than $(\alpha + \varepsilon)KD/2^i$. From the Chernoff-Hoeffding bound it follows that probability of this event is not greater than $\exp(-2\varepsilon^2 KD/2^i)$. Hence, probability of the event *a random graph is not a prefix extractor* can be limited by the sum of such bounds for all i, S, and Y:

$$\sum_{i=0}^{k} \binom{N}{K/2^i} \cdot 2^{M/2^i} \exp(-2\varepsilon^2 KD/2^i).$$

Since $\binom{u}{v} \le u^v/v! \le (ue/v)^v$, this sum does not exceed

$$\sum_{i=0}^{k} \left(\frac{eN}{K/2^i}\right)^{K/2^i} 2^{M/2^i} \exp(-2\varepsilon^2 KD/2^i) =$$

$$= \sum_{i=0}^{k} \left(e^{(K/2^i)(1+\ln(2^i N/K))} \cdot e^{-\varepsilon^2 KD/2^i}\right) \cdot \left(e^{M \ln 2/2^i} \cdot e^{-\varepsilon^2 KD/2^i}\right).$$

The condition of the theorem implies that $D \ge \frac{M}{K} \cdot \frac{\ln 2}{\varepsilon^2}$, assuming that $O(1)$ constant is large enough. Hence, the second factor in each term of the sum is not greater than 1. On the other hand, the first factor equals

$$e^{(K/2^i)(1+\ln(2^i N/K)-\varepsilon^2 D)} \le e^{(K/2^i)(1+\ln N-\varepsilon^2 D)},$$

which is less than $(1/2)^{(K/2^i)}$, since $D\varepsilon^2 \ge 1 + \ln 2 + \ln N$. The sum of these terms is strictly less than 1. Thus, probability of the event *a random graph is a prefix extractor* must be positive. □

However, using prefix extractors is not enough; we need to modify the argument, since now we need to find two related neighbors in two graphs. So we modify the notion of a dangerous vertex and use the following analog of Lemma 1:

Lemma 2. *Let us call a left-part element weakly dangerous in S if at least half of its neighbors are bad for S. Then the number of weakly dangerous elements in S is at most $4\varepsilon K$.*

Proof: is similar to the proof of Lemma 1. Since only half of all neighbors are bad, we need twice more elements. □

Now we give a new proof of Theorem 4 based on prefix extractors. Fix a prefix extractor E with parameters n, k, $d = O(\log n)$, $m = k$ and $\varepsilon = 1/n^3$. Again, we may assume that complexity of this extractor is $2\log n + O(1)$. We also may assume that $\mathrm{C}(a \mid b) = k - 1$, $\mathrm{C}(a \mid c) = l - 1$ and (without loss of generality) $k \ge l$.

Let S_b and S_c be the sets of strings of conditional complexity less than k and l conditional on b and c respectively. Call an element *weakly dangerous in S_b* if it is weakly dangerous (in S_b) for the original extractor and *weakly dangerous in S_c* if it is weakly dangerous (in S_c) for the l-bit prefix of E. Since this prefix E_{k-l} is also an extractor, the statement of Lemma 2 holds for S_c. The string a belongs to the intersection of S_b and S_c and is not weakly dangerous in both. Hence, a random neighbor of a and its prefix are not bad for S_b [resp. S_c] with probability greater than 1/2. So we can find a k-bit string p such that p and its l-bit prefix q are not bad for S_b and S_c respectively.

They satisfy the requirements. Indeed, the conditional complexities $\mathrm{C}(p \mid a)$ and $\mathrm{C}(q \mid a)$ are logarithmic because p and q can be specified by their ordinal numbers among the neighbors of a in the extractor. The string a may be obtained from p and b with logarithmic advice because p is not bad for S_b in E; similarly, a can be obtained from q and c with logarithmic advice because q is not bad for S_c in E_{k-l}. This completes the proof of Muchnik's theorem for two conditions.

□

3.4 Muchnik's Theorem about Space-Bounded Complexity

The arguments from Sect. 3.2 together with constructions of explicit extractors imply some versions of Muchnik's theorem for resource-bounded Kolmogorov complexity. In this section we present such a theorem for the space-bounded complexity.

First of all, the definitions. Let φ be a multi-tape Turing machine that transforms pairs of binary strings to binary strings. Conditional complexity $\mathrm{C}^{t,s}_{\varphi}(a \mid b)$ is the length of the shortest x such that $\varphi(x, b)$ produces a in at most t steps using space at most s. It is known (see [4]) that there exists an *optimal description method* ψ in the following sense: for every φ there exists a constant c such that

$$\mathrm{C}^{ct\log t, cs}_{\psi}(a \mid b) \leq \mathrm{C}^{t,s}_{\varphi}(a \mid b) + c.$$

We fix such a method ψ, and in the sequel use notation $\mathrm{C}^{t,s}$ instead of $\mathrm{C}^{t,s}_{\psi}$.

Now we present our variant of Muchnik's theorem for space-bounded Kolmogorov complexity:

Theorem 6. *Let a and b be binary strings and n, k and s be numbers such that $\mathrm{C}^{\infty,s}(a) < n$ and $\mathrm{C}^{\infty,s}(a \mid b) < k$. Then there exists a binary string p such that*

- $\mathrm{C}^{\infty,O(s)+\mathrm{poly}(n)}(a \mid p, b) = O(\log^3 n)$;
- $\mathrm{C}^{\infty,O(s)}(p) \leq k + O(\log n)$;
- $\mathrm{C}^{\infty,\mathrm{poly}(n)}(p \mid a) = O(\log^3 n)$,

where all constants in O- and poly*-notation depend only on the choice of the optimal description method.*

Proof. The proof of this theorem starts as an effectivization of the argument of Sect. 3.2. To find p effectively, we use an explicit extractor with parameters n, k, $d = O(\log^3 n)$, $m = k$ and $\varepsilon = 1/n^3$. We increase d and respectively the conditional complexity of p when a is given from $O(\log n)$ to $O(\log^3 n)$, because currently known explicit extractors use more random bits than the ideal extractors from Theorem 2.[4]

First we prove a weaker version of the theorem assuming that the value of s is added as a condition (in three complexities that are bounded by the theorem). Later we explain how to get rid of this restriction.

Recall that a right-part element is bad if it has more than DK/M neighbors on the left and a left-part element is dangerous if all its neighbors are bad. Let us show that if a is not dangerous and p is a neighbor of a that is not bad, then we can recover a from b and p using $O(\log^3 n)$ extra bits of information and $O(s) + \mathrm{poly}(n)$ space. For any string a' we can test in $O(s) + \mathrm{poly}(n)$ space whether $\mathrm{C}^{\infty,s}(a' \mid b) < k$: We test sequentially all programs of length less than k and check if they produce a' on space s given b. Simulating every such a program,

[4] *Note added in proof.* Using Nisan–Wigderson construction of pseudorandom bit generator one may improve this result and replace $\log^3$ by log, as in the original Muchnik's theorem. This argument will be published elsewhere.

we limit its workspace to s, and prevent infinite loops by counting the number of steps. If a program makes more than c^s steps in space s then it loops; here c is some constant that depends only on the choice of the universal Turing machine. This counter uses only $O(s)$ space. Therefore, given b and p we can enumerate all the strings a' that are neighbors of p and $\mathrm{C}^{\infty,s}(a' \mid b) < k$, and wait until a string with a given ordinal number appears.

The difficulty arises when we try to prove that a is not dangerous. Let us try to repeat our arguments taking into account the space restrictions. First we note that one can enumerate (or recognize: for space complexity it is the same) all bad elements in the right part using space $O(s) + \mathrm{poly}(n)$. As before, we assume here that s is given in addition to n, k, and b. Indeed, bad elements (as defined above) have many neighbors among strings a' such that $\mathrm{C}^{\infty,s}(a' \mid b) < k$, and those strings can be enumerated.

Therefore, we can also enumerate all dangerous elements in the left part using space $O(s) + \mathrm{poly}(n)$. We know also that the number of dangerous elements is small, but this does not give us a contradiction (as it did before) since the space used by this enumeration increases from s to $O(s) + \mathrm{poly}(n)$, and even a small increase destroys the argument. So we cannot claim that a is not dangerous and need to deal somehow with dangerous elements.

To overcome this difficulty, we use the same argument as in Sect. 2.3. We treat the dangerous elements at the next layer, with reduced k and other extractor graph. We need $O(k)$ layers (in fact even $O(k/\log n)$ layers) since by Lemma 1 at every next layer the number of dangerous elements that still need to be served is reduced at least by the factor 2ε. Note also that the space overhead needed to keep the accounting information is $\mathrm{poly}(n)$ and we never need to run in parallel several computations that require space s; this space is needed only at the bottom level of the recursion, in all other cases $\mathrm{poly}(n)$ is enough.

So we get the theorem in its weak form (with condition s). For the full statement some changes are needed. Let us sequentially use space bounds $s' = 1, 2, \ldots$: to enumerate all strings a' such that $\mathrm{C}^{\infty,s}(a' \mid b) < k$, we sequentially enumerate all strings that can be obtained from b and a k-bit encoding using space $s' = 1, 2$, etc. The corresponding set increases as s' increases, and at some point we enumerate all strings a' such that $\mathrm{C}^{\infty,s}(a' \mid b) < k$, though this moment is not known to us. Note that we can avoid multiple copies of the same string for different values of s': performing the enumeration for s', we check for every string whether it has appeared earlier, using $s' - 1$ instead of s'. This requires a lot of time, but only $O(s)$ space. Knowing the ordinal number of a in the entire enumeration, we stop as soon as it is achieved; hence, the enumeration process requires only space $O(s) + \mathrm{poly}(n)$, though s is not specified explicitly.

Similarly, the set of dangerous strings a (that go to the second or higher layer) increases as s' increases, and can be enumerated sequentially for $s' = 1, 2, 3 \ldots$ without repetitions in $O(s') + \mathrm{poly}(n)$ space. Therefore, at every layer we can use the same argument, enumerating all the elements that reach this layer and at the same time are neighbors of p, until we produce as many of them as required. □

Remarks. 1. The process of enumerating a' such that $\mathrm{C}^{\infty,s'}(a' \,|\, b) < k$ sequentially for $s' = 1, 2, 3, \ldots$ can be considered as the enumeration of all a' such that $\mathrm{C}(a' \,|\, b) < k$. So we just get the proof for the unrestricted version of Muchnik's theorem with an additional remark: if an explicit extractor is used, then the short programs provided by this theorem require only slightly more space than the programs given in the condition.

2. When we use several layers (instead of a contradiction with the assumption that the complexity $\mathrm{C}(a \,|\, b)$ is exactly $k-1$) we in fact do not need ε to be as small as $1/n^3$; it is enough to use a small constant value of ε.

3.5 Muchnik's Theorem for CAM-complexity

The arguments from the previous sections cannot be applied for Kolmogorov complexity with polynomial *time* bound. Roughly speaking, the obstacle is the fact that we cannot implement an exhaustive search over the list of 'bad' strings in polynomial time unless P = NP. The best result that we can prove for poly-time bounded complexity involves a version of Kolmogorov complexity introduced in [9]:

Definition 3. *Let U_n be a non-deterministic universal Turing machine. Arthur-Merlin complexity* $\mathrm{CAM}^t(x \,|\, y)$ *is the length of a shortest string p such that*

1. $\mathrm{Prob}_r[U_n(y, p, r)$ *can print x and cannot print any other string* $] > 2/3$
2. $U_n(y, p, r)$ *stops in time at most t (for all branches of non-deterministic computation).*

As always, $\mathrm{CAM}^t(x) := \mathrm{CAM}^t(x \,|\, \lambda)$.

This definition is typically used for $t = \mathrm{poly}(|x|)$. Intuitively, a CAM-description p of a string x given another string y is an interactive Arthur–Merlin protocol: Arthur himself can do probabilistic polynomial computations, and can ask questions to all-powerful but not trustworthy Merlin; Merlin can do any computations and provide to Arthur any requested certificate. So, Arthur should ask such questions that the certificates returned by Merlin could be effectively used to generate x. With this version of resource-bounded Kolmogorov complexity we have a variant of Muchnik's theorem:

Theorem 7. *For every polynomial t_1, there exists a polynomial t_2 such that the following condition holds. Let a, b be strings such that $\mathrm{C}^{t_1(n),\infty}(a \,|\, b) < k$, where $n = |a|$. Then there exists a string p of length $k + O(\log^3 n)$ such that*

- $\mathrm{C}^{t_2(n),\infty}(p \,|\, a) = O(\log^3 n)$ *and*
- $\mathrm{CAM}^{t_2(n)}(a \,|\, b, p) = O(\log^3 n)$.

Proof: In the proof of this theorem we cannot use an arbitrary effective extractor. We employ very essentially properties of one particular extractor constructed by L. Trevisan [10]. Our arguments mostly repeat the proof of Theorem 3 from [9].

First of all we remind the definition of the Trevisan extractor, which is based on the technique from the seminal paper by Nisan and Wigderson [12]. The first crucial ingredient of the Trevisan function is a weak design. A system of sets

$$S_1, \ldots, S_m \subset \{1, \ldots, d\}$$

is called a weak design with parameters (l, d) if each S_i consists of l elements and for every $i > 1$ the sum $\sum_{j=1}^{i-1} 2^{\#(S_i \cap S_j)}$ is bounded by $(m-1)$. Weak designs exist; moreover, they can be constructed effectively. More precisely, there exists an algorithm that for any given l, m generates a week design with $d = O(l^2 \log m)$ in time polynomial in l and m, see [12].

Let us fix a weak design as above. For $x \in \{0,1\}^d$ we use the following notation: $x|_{S_i}$ denotes the l-bit string that is obtained by projecting x onto coordinates specified by S_i.

The second important ingredient of Trevisan's construction is an error correcting code. For every positive integer n and $\delta > 0$, there exists a list decodable code

$$\mathrm{LDC}_{n,\delta} : \ \{0,1\}^n \to \{0,1\}^{\bar{n}}$$

where $\bar{n} = \mathrm{poly}(n/\delta)$, such that

1. $\mathrm{LDC}_{n,\delta}(x)$ can be computed in polynomial time;
2. given any $y \in \{0,1\}^{\bar{n}}$, the list of all $x \in \{0,1\}^n$ such that $\mathrm{LDC}_{n,\delta}(x)$ and y agree in at least $(1/2+\delta)$ fraction of bits, can be generated in time $\mathrm{poly}(n/\delta)$. In particular, this property means that the number of words x in this list is not greater than $\mathrm{poly}(n/\delta)$;

(see, e.g., [13]). In the sequel we will assume that $\bar{n}$ is a power of 2.

Let us fix an encoding as above and denote $l(n) = \log \bar{n}$. For $u \in \{0,1\}^n$ the value $\mathrm{LDC}_{n,\delta}(u)$ is a string of length 2^l. So, we can view $\mathrm{LDC}_{n,\delta}(u)$ as a Boolean function

$$\hat{u} : \{0,1\}^l \to \{0,1\}$$

Having fixed a weak design $S_1, \ldots, S_m$ and an encoding $\mathrm{LDC}_{n,\delta}$, we define the Trevisan function $\mathrm{TR}_\delta : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ as

$$\mathrm{TR}_\delta(u, y) = \hat{u}(y|_{S_1}) \ldots \hat{u}(y|_{S_m}).$$

We do not need to show that TR is an extractor (for suitable values of n, d, m); in our proof we refer directly to the definition of this function. We will use the Trevisan function for $\delta = \frac{1}{8m}$ and $m = k + d + 1$. More precisely, the parameters are chosen as follows. Numbers k and n are taken from the statement of the theorem; $l(n) = \log \bar{n}$ is obtained from the construction of $\mathrm{LDC}_{n,\delta}$; further, we can choose appropriate m and $d = O(l^2 \log m) = O(\log^3 n)$ so that (i) there exists a weak design with parameters m, l, d, and (ii) it holds $m = k + d + 1$.

Denote by L_b the set of all strings whose time-bounded complexity conditional on b is less than k:

$$L_b = \{u \in \{0,1\}^n \mid C^{t_1(n),\infty}(u \,|\, b) < k\}.$$

Then $a \in L_b$ and $\#L_b < 2^k$. We have chosen such an m that the TR-image of $L_b \times \{0,1\}^d$ covers at most 50% of the set $\{0,1\}^m$. Denote by B the predicate *being in the* TR*-image of* $L_b \times \{0,1\}^d$. For every $u \in L_b$

$$\mathrm{Prob}_{r_1 \ldots r_d}[B(\mathrm{TR}_\delta(u, r_1 \ldots r_d)) = 1] - \mathrm{Prob}_{r_1 \ldots r_m}[B(r_1 \ldots r_m) = 1] \geq 1/2$$

since the first probability is equal to 1 and the second one is not greater than 1/2. In other notation, we have

$$\begin{aligned}\mathrm{Prob}_{y \in \{0,1\}^d}[B(\hat{u}(y|_{S_1})\hat{u}(y|_{S_2}) \ldots \hat{u}(y|_{S_m})) = 1] - \\ - \mathrm{Prob}_{r_1 \ldots r_m}[B(r_1 \ldots r_m) = 1] \geq 1/2.\end{aligned}$$

We apply the standard 'hybridization' trick: we note that for some i,

$$\begin{aligned}&\mathrm{Prob}_{y, r_{i+1}, \ldots, r_m}[B(\hat{u}(y|_{S_1})\hat{u}(y|_{S_2}) \ldots \hat{u}(y|_{S_i}) r_{i+1} \ldots r_m) = 1] - \\ &- \mathrm{Prob}_{y, r_i, r_{i+1}, \ldots, r_m}[B(\hat{u}(y|_{S_1})\hat{u}(y|_{S_2}) \ldots \hat{u}(y|_{S_{i-1}}) r_i \ldots r_m) = 1] \geq 1/(2m).\end{aligned} \tag{2}$$

Further, we can somehow fix the bits of y outside S_i so that (2) remains true. Denote $y|_{S_i}$ by x. Now each function $\hat{u}(y|_{S_j})$ depends on $\#(S_j \cap S_i)$ bits from x (the other bits of y are fixed). We denote this function by f_j. The truth table of

$$f_j \ : \ \{0,1\}^{\#(S_j \cap S_i)} \to \{0,1\}$$

consists of $2^{\#(S_j \cap S_i)}$ bits. With a slight abuse of notations we will write $f_j(x)$ (though f_j depends on only $\#(S_j \cap S_i)$ bits of string x). Note that the definition of f_j involves implicitly the string u and those bits in y that we fixed outside positions S_i.

To specify the truth tables of all functions $f_1, \ldots, f_{i-1}$ we need

$$\sum_{j=1}^{i-1} 2^{\#(S_j \cap S_i)} < m \tag{3}$$

bits (the last inequality follows from the definition of weak designs).

The explained construction of functions $f_1, \ldots, f_{i-1}$ works for every $u \in \{0,1\}^n$. We denote by p the concatenation of the truth tables of $f_1, \ldots, f_{i-1}$ for $u = a$, where a is the string from the statement of the theorem. By (3) the length of p is less than m.

To specify this p given a, we need to know only m, i and the bits of y fixed outside S_i. Hence, $\mathrm{C}^{\mathrm{poly}(n),\infty}(p \,|\, a) = O(\log^3 n)$.

In the rest of the proof we show that there exists an Arthur–Merlin protocol that reconstructs a given b, p and some small additional information. Since $u = a$, it is enough to reconstruct string $\hat{u}$ (then we apply the decoding procedure and find $a = \mathrm{LDC}^{-1}_{n,\delta}(\hat{u})$).

Let us investigate inequality (2). To make the notations more concise, we denote

$$g_{r_i}(x, r_{i+1} \ldots r_m) = \begin{cases} r_i & \text{if } B(f_1(x) \ldots f_{i-1}(x) r_i \ldots r_m) = 1 \\ 1 - r_i & \text{otherwise.} \end{cases}$$

By a standard argument from the computational XOR Lemma [14] we get

$$\mathrm{Prob}_{x\in\{0,1\}^l, r_i\dots r_m\in\{0,1\}^{m-i+1}}[\hat{u}(x) = g_{r_i}(x, r_{i+1}\dots r_m)] \geq 1/2 + 1/(2m). \quad (4)$$

Now we fix a value of r_i (set it to 0 or 1) so that inequality (4) remains true. This bit must be included into the description of a given b and p. Without any loss of generality we assume that $r_i = 1$, and in the sequel we omit r_i in our notations. In other words, instead of $g_{r_i}(x, r_{i+1}\dots r_m)$ we write

$$g(x, r_{i+1}\dots r_m) = \begin{cases} 1 & \text{if } B(f_1(x)\dots f_{i-1}(x)1r_{i+1}\dots r_m) = 1 \\ 0 & \text{otherwise.} \end{cases}$$

If the word p defined above and a "typical" sequence $r_{i+1}\dots r_m$ are given, Arthur can approximate $\hat{u}$ and then reconstruct a (using decoding algorithm for $\mathrm{LDC}_{n,\delta}$). So, Arthur chooses at random several copies of $r_{i+1}\dots r_m$ and tries to approximate $\hat{u}$ with each copy. Further we explain how it works.

First, we need some notation. We say a string $v' \in \{0,1\}^{\bar{n}}$ is an α-approximation to a string $v \in \{0,1\}^{\bar{n}}$ if these strings coincides in at least $\alpha\bar{n}$ bits. In particular, we will be interested in α-approximation to $\hat{u}$.

If we fix in $g(x,r)$ the second argument r, we get some Boolean function $g^{(r)}(x)$ that depends on $x \in \{0,1\}^l$. For every fixed r we identify the corresponding function $g^{(r)}(x)$ with its truth table, i.e., with the string $z^{(r)}$ of length $\bar{n} = 2^l$ where every x-th bit equals 1 iff $g(x,r) = 1$. So, the number of 1's in $z^{(r)}$ is equal to the number of strings x such that $B(f_1(x)\cdots f_{i-1}(x)1r) = 1$.

We say that a string $v \in \{0,1\}^{\bar{n}}$ is a *candidate* if v is a codeword of $\mathrm{LDC}_{n,\delta}$, and for at least $1/32m$ of all $r \in \{0,1\}^{m-i}$ the corresponding string $z^{(r)}$ is an $(1/2 + 1/8m)$-approximation to v. From the decoding property of the code $\mathrm{LDC}_{n,\delta}$, each $z \in \{0,1\}^{\bar{n}}$ can be an $(1/2 + 1/8m)$-approximation for at most $q = \mathrm{poly}(m)$ different codewords $\mathrm{LDC}_{n,\delta}(u)$. Hence, there exist at most $32mq$ candidates (of course, $\hat{u}$ is a candidate). By Sipser's CD-coding theorem [15] there exists a poly-time program p' of length $2\log(32mq) = O(\log n)$ that accepts $\hat{u}$ and rejects all other candidates (no warranty about non-candidates: p' may accept or reject any of them).

First part of the Arthur–Merlin protocol: Denote

$$\bar{g} = \sum_{x,r} g(x,r)/2^{m-i}.$$

This is the average number of strings $x \in \{0,1\}^l$ such that $g(x,r) = 1$ for a random $r \in \{0,1\}^{m-i}$.

At first Arthur chooses s random strings $r(1), \dots, r(s)$ of length $(m-i)$ (a polynomial $s = s(n)$ is specified below). He asks Merlin to generate $s \cdot (\bar{g} - \gamma)$ ($\gamma = \gamma(n)$ is also specified below) certificates for the facts that different tuples $\langle x, r(j)\rangle$ satisfy $g(x, r(j)) = 1$, and verifies these certificates.

Indeed, if $B(w) = 1$ for some string w, Merlin can provide a certificate for this fact: he communicates to Arthur (i) some u, y such that $\mathrm{TR}_\delta(u,y) = w$, and (ii) provides a poly-time program π of length less than k such that $\pi(b)$ stops in t_1 steps and returns u; that is, Merlin proves to Arthur that $u \in L_b$.

If at least one certificate is false, Arthur stops without any answer. If the certificates are OK, Arthur calculates $\tilde{z}_1, \ldots, \tilde{z}_s$, where x-th bit of $\tilde{z}_j$ is 1 iff Merlin provided a certificate of the fact that $g(x, r(j)) = 1$.

We need the following probabilistic lemma:

Lemma 3. *For some rational $\gamma = \bar{n}/\operatorname{poly}(m)$ and integer $s = \operatorname{poly}(n)$, for randomly chosen strings $r(1), \ldots, r(s)$ of length $m - i$, with probability more than $2/3$ Merlin can provide some certificates for at least $s \cdot (\bar{g} - \gamma)$ strings $r(j)$ (each certificate must prove for one of $r(j)$ that $g(x, r(j)) = 1$), and, whatever certificates are chosen by Merlin, the following two conditions hold:*

- *At least $(s/16m)$ of s strings $\tilde{z}_{r(1)}, \ldots, \tilde{z}_{r(s)}$ (corresponding to the certificates given by Merlin) are $(1/2 + 1/8m)$-approximations to $\hat{u}$.*
- *For every codeword v of $\mathrm{LDC}_{n,\delta}$, if at least $s/16m$ of s strings $\tilde{z}_{r(1)}, \ldots, \tilde{z}_{r(s)}$ are $(1/2 + 1/8m)$-approximations to v, then v is a candidate.*

Proof: see Claims 17 and 18 in [9].

In our Arthur–Merlin protocol we use the parameters s and γ from Lemma 3.

Second part of the Arthur–Merlin protocol. Arthur does not need anymore to communicate with Merlin. Now he composes the list of all codewords v that are $(1/2 + 1/8m)$-approximated by at least $s/16m$ of strings $\tilde{z}_1, \ldots, \tilde{z}_s$. From Lemma 3 it follows that with probability more than $2/3$ all strings in this list are candidates, and the string $\hat{u}$ is included in the list. The program p' defined above can distinguish $\hat{u}$ from other strings from the list.

Thus, Arthur can find $\hat{u}$ in polynomial time if he is given b, p and the following additional information: the index i, the bit r_i, the mean value $\bar{g}$, and the distinguishing program p'. In fact, it is enough to know not the exact value of $\bar{g}$ but only an approximation to this number; this approximation must be precise enough so that Arthur can find the integer part of $s\bar{g}$. Thus, the required additional information contains only $O(\log n)$ bits.

Now we check that the described protocol of generating a satisfies the definition of CAM-complexity. The CAM-program for a consists of (i) the truth tables of functions $f_1, \ldots, f_{i-1}$ constructed from $\hat{u}(y|_{S_1})$, $\ldots$, $\hat{u}(y|_{S_{i-1}})$ for $u = a$ (this is the longest part of the program; we denoted it by p), (ii) the bit r_i chosen so that (4) is true, (iii) a rational γ and an approximation to a rational $\bar{g}$, and (iv) Sipser's code p' that distinguishes $\hat{u}$ between all "candidates". The Arthur–Merlin protocol works as follows. Arthur chooses at random strings $r(1), \ldots, r(s)$. Merlin provides $s \cdot (\bar{g} - \gamma)$ certificates corresponding to these $r(j)$. Arthur computes $\tilde{z}_1, \ldots, \tilde{z}_s$ corresponding to the obtained certificates and finds the list of all $\mathrm{LDC}_{n,\delta}$-codewords v that are $(1/2 + 1/8m)$-approximated by at least $s/16m$ of these $\tilde{z}_j$. Then Arthur selects $\hat{u}$ from this list of strings using distinguishing program p', and computes $a = \mathrm{LDC}^{-1}_{n,\delta}(\hat{u})$.

If Merlin is fair, this plan works OK with probability more than $2/3$ (Lemma 3). If Merlin wants to cheat, he has two options: provide a list of certificates such that the required string $\hat{u}$ is not approximated by $s/16m$ of $\tilde{z}_1, \ldots, \tilde{z}_s$, or such that at least $s/16m$ of $\tilde{z}_j$ approximate some *non-candidate* codeword v (in these cases Arthur fails to select $\hat{u}$ using p'). However from Lemma 3 it follows that

for random $r(1), \ldots, r(s)$ both these ways of cheating are impossible with probability more than 2/3. □

Acknowledgments

This article is based on several discussions and reports presented at the Kolmogorov seminar (Moscow). Preliminary versions appeared as [8] and [5]. The authors are grateful to all participants of the seminar for many useful comments. The authors thank also anonymous referees for very detailed comments and helpful suggestions leading to a significant revision of this paper.

References

1. J. Radhakrishnan, A. Ta-Shma, Bounds for dispersers, extractors, and depth-two superconcentrators, *SIAM Journal on Discrete Mathematics*, **13**(1): 2–24, 2000.
2. O. Reingold, R. Shaltiel, A. Wigderson, Extracting randomness via repeated condensing, *SIAM Journal on Computing* **35**(5):1185–1209, 2006.
3. H. Buhrman, L. Fortnow, S. Laplante, Resource bounded Kolmogorov complexity revisited, *SIAM Journal on Computing*, **31**(3):887–905, 2002.
4. M. Li, P. Vitanyi, An Introduction to Kolmogorov Complexity and Its Applications, 2 ed., 1997. Springer-Verlag.
5. D. Musatov, Extractors and an effective variant of Muchnik's theorem. Diplom (Master thesis). *Faculty of Mechanics and Mathematics, MSU*, 2006. `http://arxiv.org/abs/0811.3958` (in Russian).
6. An.A. Muchnik, On basic structures of the descriptive theory of algorithms. *Soviet Math. Dokl.*, **32**:671–674, 1985.
7. An. Muchnik, Conditional complexity and codes, *Theoretical Computer Science*, **271**(1–2):97–109, 2002.
8. A. Shen, Combinatorial proof of Muchnik's theorem, *Kolmogorov complexity and applications*, M. Hutter, W. Merkle, P. Vitanyi, eds., Dagstuhl Seminar Proceedings 06051, ISSN 1862–4405, `http://drops.dagstuhl.de/opus/volltexte/2006/625`.
9. H. Buhrman, T. Lee, D. van Melkebeek. Language compression and pseudorandom generators. In *Proc. of the 15th IEEE Conference on Computational Complexity*, IEEE, 2004, 228–255.
10. L. Trevisan. Construction of extractors using pseudo-random generators. In *Proc. 31 Annual ACM Symposium on Theory of Computing*, 1999, 141–148.
11. D. Slepian and J. K. Wolf. Noiseless coding of correlated information sources. *IEEE Transactions on information Theory*, **19**:471–480, 1973.
12. N. Nisan and A. Wigderson. Hardness vs. Randomness. *Journal of Computer and System Sciences.* **49**: 149–167, 1994.
13. M. Sudan. Decoding of Reed Solomon codes beyond the error-correcting bound. *Journal of complexity*, **13**(1):180–193, 1997.
14. O. Goldreich. Three XOR-Lemmas – An Exposition. ECCC TR95-056, 1995.
15. M. Sipser. A complexity theoretic approach to randomness. In *Proc. of the 15th Annual ACM Symposium on Theory of Computing*, 1983, 330–335.