Inequalities for space-bounded Kolmogorov complexity

Peter Gács, Andrei Romashchenko, Alexander Shen

Inequalities for space-bounded Kolmogorov complexity*

Peter Gács,† Andrei Romashchenko,‡ Alexander Shen§

Abstract

There is a parallelism between Shannon information theory and algorithmic information theory. In particular, the same linear inequalities are true for Shannon entropies of tuples of random variables and Kolmogorov complexities of tuples of strings (Hammer et al., 1997), as well as for sizes of subgroups and projections of sets (Chan, Yeung, Romashchenko, Shen, Vereshchagin, 1998–2002). This parallelism started with the Kolmogorov–Levin formula (1968) for the complexity of pairs of strings with logarithmic precision. Longpré (1986) proved a version of this formula for space-bounded complexities.

In this paper we prove an improved version of Longpré's result with a tighter space bound, using Sipser's trick (1980). Then, using this space bound, we show that every linear inequality that is true for complexities or entropies, is also true for space-bounded Kolmogorov complexities with a polynomial space overhead.

*Authors want to thank the members of the ESCAPE team (especially Ruslan Ishkuvatov for the help with the proof of Lemma 1), participants of the Kolmogorov seminar (Moscow) and Algorithmic Randomness workshop for discussions.

†Boston University, gacs@bu.edu, ORCID 0000-0003-2496-0332

‡LIRMM, University of Montpellier, CNRS, Montpellier, France and IITP RAS, Moscow (on leave), https://www.lirmm.fr/~romashchen/, andrei.romashchenko@lirmm.fr, ORCID 0000-0001-7723-7880.
Supported by ANR-15-CE40-0016 RaCAF grant

§LIRMM, University of Montpellier, CNRS, Montpellier, France and IITP RAS, Moscow (on leave), www.lirmm.fr/~ashen, alexander.shen@lirmm.fr, ORCID 0000-0001-8605-7734. Supported by ANR-15-CE40-0016 RaCAF. Part of the work was done while participating in the American Institute of Mathematics Workshop on Algorithmic Randomness (August 2020)

1 Space-bounded Kolmogorov complexity

Kolmogorov in his seminal paper of 1965 [4] defined the complexity of a finite string as the minimal length of a program that produces this string:

$$\mathrm{C}_I(x) = \min\{|p| : I(p) = x\}.$$

Here I is a machine (considered as an interpreter of some programming language) and p is a binary string (considered as a program without input). In a similar way Kolmogorov defined $\mathrm{C}_I(y|x)$, the conditional complexity of y given x, as the minimal length of a program that transforms x to y:

$$\mathrm{C}_I(y|x) = \min\{|p| : I(p,x) = y\}.$$

In this case the interpreter I has two arguments (considered as a program and an input for this program).

Since there is an interpreter that is optimal to within an additive constant, these functions can be considered as an intrinsic property of the strings involved—their individual information content. Their relation to the entropy of probability distributions is well established: in particular it is shown in [3] that the same linear inequalities hold for both.

To information theorists, the uncomputability of the complexity functions may obscure somewhat their combinatorial significance. A natural approach to the question is to consider versions of Kolmogorov complexity in which the interpreter has some resource (time, space) bounds. However, the basic inequalities for these resource bounded versions become more complex, with different resource bounds on the two sides. The present paper shows a way to overcome these difficulties, on the example of space bounds (see the remarks after Theorem 4). Each linear inequality holding for entropies holds also for many space-bounded versions of Kolmogorov complexity: the space bound can be chosen from a dense infinite hierarchy of possibilities.

Kolmogorov was aware of these issues: in the last paragraph of [4], he writes that the description complexities introduced above:

> "... have one important disadvantage: They do not take into account the *difficulty* of transforming a program p and an object x into an object y.[1] Introducing necessary definitions, one can prove some mathematical statements that may be interpreted as existence of objects that have very short programs, so their complexity is very small, but the reconstruction of an object from the program requires enormous amount of time. I plan to study elsewhere[2] the dependence of the necessary program complexity $K^t(x)$ on the allowed difficulty t for its transformation into x. Then the complexity $K(x)$ as defined earlier will be the minimum of $K^t(x)$ for unbounded t."[4, p. 11][3].

[1] The English translation mentions the "difficulty of preparing a program p for passing from an object x to an object y". However, the original Russian text is quite clear: Kolmogorov speaks about complexity of decompression, not compression.

[2] Unfortunately, Kolmogorov did not publish those "mathematical statements" about resource-bounded complexity (though he gave some talks on this topic), and his ideas about algorithmic statistics, as the subject is known now, were understood only much later. It turned out that the dependence of $K^t(x)$ on t (if the resource bound t is measured in "busy beaver units") gives, for every string x, some curve that can be equivalently defined in terms of Kolmogorov structure function, (α,β)-stochasticity, or two-part descriptions. See [13, 14] for a survey of algorithmic statistics.

[3] Kolmogorov used the notation $K(x)$ for complexity function; now it is usually denoted by $\mathrm{C}(x)$, while the notation $\mathrm{K}(x)$ is used for the so-called "prefix complexity". We follow this convention.

Defining $\mathrm{C}^t(x)$ and $\mathrm{C}^t(y|x)$ as the minimal length of the programs that generate x or transform x to y with resource bound t, we need to fix some computational model and the exact meaning of the resource bound. It is natural to consider time-bounded or space-bounded computations. The study of time-bounded complexity immediately bumps into P vs. NP problem [6, 7], so in this paper we consider only the space-bounded version of complexity.

We need to fix some computational model. For technical reasons, it is convenient to use machines that have one-directional read-only input tapes (with end markers), one-directional write-only output tape, and two binary stacks as memory devices. All these devices are connected to the finite-state control unit. The alphabet is binary, and the sum of the stack lengths is considered as the amount of memory used. Note that two stacks are equivalent to a finite tape that can be extended by inserting an empty cell near the head of the Turing machine (stacks correspond to the parts of the tape on the left and on the right of the head). We will refer to this combination of stacks as "work tape" in the sequel.

One could consider other models (say, multitape Turing machines) and other ways of providing inputs and generating outputs. For example, the input and output can be written on a work tape. This changes the space complexity: simulation of one model by another one requires some overhead. In our example, if we do not consider separate input and output tapes, we need to allocate space for input and output on the work tape, and the overhead is proportional to the input/output size. If we consider machines with a different number of tapes, the simulation of space s computation with large number of tapes on a machine with a smaller number of tapes requires $s + O(\log s)$ space (we need to combine the information from different tapes, and the information about the head position and the length of the used part of each simulated tape requires $O(\log s)$ bits). The same is true for simulations on the insertable tape (= two stacks). Most of our results are insensitive to these differences since we will allow $O(\log s)$ and $O(\text{input/output length})$ increase in the space bounds anyway. Still for some arguments it is convenient to use separate one-directional input/output tapes and two stacks.

Definition 1. Let I be a machine of the described type with two one-directional read-only input tapes, one one-directional write-only output tape and two binary stacks. We say that $I^s(p,x) = y$ if machine I with inputs p and x produces y and the amount of used memory does not exceed s during the computation. We define the conditional space-bounded Kolmogorov complexity as

$$\mathrm{C}^s_I(y|x) = \min\{|p| : I^s(p,x) = y\}.$$

The unconditional version is obtained when the condition is an empty string.

Then, as usual, we need a version of the Kolmogorov–Solomonoff universality theorem that says that there exists an optimal machine making the complexity minimal. Now the space bounds should be taken into account, and for our model $O(1)$ additional space is enough:

Proposition 1. *There exists an optimal machine V such that for every machine P there exists a constant c such that*

$$\mathrm{C}^{s+c}_V(y|x) \leqslant \mathrm{C}^s_P(y|x) + c$$

for all x, y.

Proof sketch. As usual, consider a universal machine $U(r,p,x)$ that can simulate the behavior of an arbitrary machine (an arbitrary finite-state program for the control unit) knowing its description r, for arbitrary inputs p, x:

$$U(r,p,x) = M_r(p,x).$$

Here r is a binary string that describes some machine M_r.

The construction of the optimal machine goes as usual. Let us double each bit in r and denote the result by $\overline{r}$. Then the required optimal machine V is defined in such a way that

$$V(\overline{r}01p,x) = U(r,p,x) = M_r(p,x).$$

This guarantees, if r is the description of machine P (i.e., $M_r = P$), that

$$\mathrm{C}_V(y|x) \leqslant \mathrm{C}_P(y|x) + 2|r| + 2,$$

but now we have to care about the space bounds and should specify in more detail how the machine V works. It starts with copying the description r onto the work tape with low density (say, using one cell out of two), so the remaining cells can be used for the auxiliary computations. This part of the tape is a "simulation block" that is moved along the tape; this block also has some zone that keeps the current internal state of the simulated machine, and some free space of fixed size needed for the local computations of the simulation (comparison of the current state of simulated machine with the transition table, etc.). Then the actions of machine P are simulated step by step, using the computations inside the simulation block, while the tape outside the simulation block is exactly the same as the tape for P. Note that we have one-sided input tapes, so after reading all bits of $\overline{r}01$ in $\overline{r}01p$ the input tape behaves exactly as for the input p of P.[4] Therefore, the space used is the space used by P, plus the size of the simulation block (that depends on r but not on p or x).

□

Now we fix some optimal machine V, call the corresponding function $\mathrm{C}^s_V(y|x)$ the *space-bounded Kolmogorov conditional complexity function* and denote it by $\mathrm{C}^s(y|x)$. The unconditional space-bounded complexity $\mathrm{C}^t(x)$ can be defined then as $\mathrm{C}^t(x|\varepsilon)$ for empty condition ε.

2 Space-bounded complexity of pairs

The Kolmogorov–Levin theorem (formula for the complexity of pairs) says that

$$\mathrm{C}(x,y) = \mathrm{C}(x) + \mathrm{C}(y|x) + O(\log \mathrm{C}(x,y)).$$

Here $\mathrm{C}(x,y)$ is the complexity of a pair of strings that is defined as the complexity of some computable encoding for it. For the unbounded complexity the choice of encoding is not important, since any computable transformation changes the unbounded complexity only by an $O(1)$ additive term. For the space-bounded version this is no more the case, and we define the complexity $\mathrm{C}^s(x,y)$ as $\mathrm{C}^s(\overline{x}01y)$.

Proposition 2.

$$\mathrm{C}^{s+O(|x|+|y|)}(y,x) \leqslant \mathrm{C}^s(x,y) + O(1)$$

for all s,x,y.

As usual, this means that there exists some c such that $\mathrm{C}^{s+c(|x|+|y|)}(y,x)$ is bounded by $\mathrm{C}^s(x,y) + c$ for all s, x, and y.

[4] For two-sided input tapes we would have a problem: during the simulation the machine should know when it reaches the left end of p (and symbols 01 could appear anywhere in p).

Proof sketch. Consider the optimal machine, and modify it by transforming the output: if it were $\bar{x}01y$, now it should be $\bar{y}01x$. Then we use the Kolmogorov–Solomonoff theorem to compare the complexity functions for this modified machine and the optimal one, and get the required inequality with $O(1)$ precision. However, we need to estimate the space overhead caused by the output transformation. First, instead of writing the pair x, y on the output tape, we need to keep it in a special zone on the work tape that is carried along the tape with the head. In our model with "insertable tape" (or two stacks) this is easy. Note that we do not need to know in advance the output size, since we can enlarge this zone when necessary; this is the advantage of the insertable tape.[5] Then we should copy x, y from this zone to the output tape in the reversed order, doubled bits for y and 01 separated. The maximal size of the special zone is $O(|x| + |y|)$, as claimed. □

The formula for the complexity of pairs consists of two inequalities: one in each direction. The first is almost straightforward:

Proposition 3.

$$\mathrm{C}^{s+O(|x|+|y|)}(x,y) \leqslant \mathrm{C}^s(x) + \mathrm{C}^s(y|x) + O(\log \mathrm{C}^s(x)).$$

Note the general structure of this statement: we consider an arbitrary bound s on the right-hand side, and on the left side we have to use a slightly bigger bound (for our model $s + O(|x| + |y|)$ is enough).

Proof sketch. Let p and q be the minimal programs for x and for $x \mapsto y$. Then the pair (x, y) can be described by a string $\bar{l}01pq$ where l is the length of p in binary. The decoding machine first reads l and stores it in the special zone, then reads and stores p (knowing its length), and then simulates the program p, carrying the special zone around the head position, and keeping the output bits (i.e., x) in the special zone. Then it reads and stores also q and simulates the computation of q using the bits of x instead of the input bits. The space overhead is then $O(|p| + |q| + |x|)$, and there is some subtle (though trivial) problem: for small s we cannot guarantee that $|p| + |q| \leqslant O(|x| + |y|)$, so the space overhead (that includes $|p| + |q|$) may not be $O(|x| + |y|)$. However, if $|p| + |q|$ significantly exceeds $|x| + |y|$, then we may use the inequality $\mathrm{C}^{O(|x|+|y|)}(x, y) \leqslant |x| + |y| + O(\log |x|)$ instead. □

Another subtle but trivial remark: we can replace $O(\log \mathrm{C}^s(x))$ by $O(\log \mathrm{C}^s(x, y))$ on the right-hand side. It is subtle, because we cannot bound $\mathrm{C}^s(x)$ by $\mathrm{C}^s(x, y)$ with exactly the same s — and trivial, since for the "paradoxical" case $\mathrm{C}^s(x, y) < \mathrm{C}^s(x)$ the entire inequality is obviously true.

The other direction is more difficult (both for unbounded and space-bounded complexity).

Theorem 1. *For all strings x, y and numbers s we have*

$$\mathrm{C}^{s'}(x) + \mathrm{C}^{s'}(y|x) \leqslant \mathrm{C}^s(x, y) + O(\log \mathrm{C}^s(x, y)) \tag{1}$$

for $s' = s + O(\log s) + O(|x| + |y|)$.

Here we use the notation s' for the space bound on the left-hand side to avoid repetitions. The exact meaning of this statement: there exists a constant c such that for all x, y, s we have $\mathrm{C}^{s'}(x) + \mathrm{C}^{s'}(y|x) \leqslant \mathrm{C}^s(x, y) + c \log \mathrm{C}^s(x, y)$ for $s' = s + c \log s + c(|x| + |y|)$.

[5] A similar argument for standard Turing machines would give additional $O(\log s)$ overhead.

Longpré [6, Theorem 3.13, p. 35] proved essentially[6] the same result with $2s + O(\log s)$ instead of $s + O(\log s)$; in his paper he uses $3s$, but his argument gives $2s + O(\log s)$ without changes. We improve this bound using Sipser's technique from [12].

Proof sketch. To prove this result, we need to look at the standard argument and see what changes are necessary for the space-bounded version. This was done in [6]. The standard argument goes as follows. Assume that $\mathrm{C}(x,y) = m$. Then we consider the set of all pairs $\langle x', y' \rangle$ such that $\mathrm{C}(x', y') \leqslant m$. They can be enumerated, and there are $O(2^m)$ of them. Our pair $\langle x, y \rangle$ is an element of this set. Count the pairs $\langle x, y' \rangle$ in this set that have the same first coordinate x. Assume that we have about 2^k of them for some k, choosing k in such a way that the number of pairs is between 2^k and 2^{k+1}. Note that y can be reconstructed from x if we know the ordinal number of y in the enumeration of pairs $\langle x, y' \rangle$ in this set (this requires $k + O(1)$ bits) and know m (so we can start the enumeration). In total we get $O(\log m) + k$ bits (including the separation overhead), so $\mathrm{C}(y|x) \leqslant k + O(\log m)$. On the other hand, we can enumerate all x' such that there are at least 2^k different y' such that $\mathrm{C}(x', y') \leqslant m$; there are at most $O(2^{m-k})$ of them, and x appears in this enumeration. So we can specify x by the ordinal number in the enumeration ($m - k + O(1)$ bits), and also the m and k needed for the enumeration. The total number of bits is $O(\log m) + m - k$, therefore $\mathrm{C}(x) \leqslant m - k + O(\log m)$. (Note that $k \leqslant m$, so k also has a prefix-free encoding of size $O(\log m)$.) Combining the bounds for $\mathrm{C}(x)$ and $\mathrm{C}(y|x)$, we get the desired result.

The enumeration used in this argument needs a lot of space, since the lists of enumerated objects are exponential in m. However, another approach is possible: knowing the space bound s, we can compute the value of $\mathrm{C}^s(u)$ by trying all programs of size at most $|u| + O(1)$ and choosing the minimal one that produces u. We need to keep the current program we are trying, and for that we need $O(|x| + |y|)$ space. Also for each program we have to check whether it produces u with space bound s. What space do we need to perform this check? We need $O(\log s)$ bits to remember the space bound s. Then we simulate the program, but should prevent it from going into an infinite loop. The simplest way to avoid loops is to keep a step counter: if we made more than $2^{s+O(\log s)}$ steps while using only space s, then some configuration has appeared twice, so we are in a loop and the current program can be rejected. To keep the counter, we need additional space of size $s + O(\log s)$, so in total we need $2s + O(\log s) + O(|x| + |y|)$ space.

Then, instead of enumerating all pairs $\langle x, y \rangle$ with $\mathrm{C}(x,y) \leqslant m$ (in some order), we just consider them in some standard ordering. We consider (in this ordering) all pairs of strings $\langle x', y' \rangle$ with $|x'|, |y'| \leqslant n$, where n is an upper bound for the lengths of x and y. For each pair we may check in space $2s + O(\log s) + O(n)$ if $\mathrm{C}^s(x,y) \leqslant m$. Also, for each x' of length n we may enumerate all y' such that $\mathrm{C}^s(x', y') \leqslant m$ in the standard ordering, and count them, so we know whether their number exceeds the threshold 2^k. We made a lot of checks, but we never keep the results of these tests, performing them again when needed, and reuse the space. In this way we prove the inequality (1) for $s' = 2s + O(\log s) + O(|x| + |y|)$, and with an additional term $O(\log s)$ on the right-hand side (as now s is needed to start the enumeration process).

Now we improve the argument, getting rid of the factor 2 and the $O(\log s)$ additional bits in the description. First let us explain how the factor 2 can be avoided using the following result that goes back to [12]:

Proposition 4 (Sipser)**.** *Let M be a machine. Then there is a machine $\overline{M}$ that decides, given a string x and number s, whether M terminates on input x in space s. Machine $\overline{M}$ uses at most*

[6] His setting is a bit different: s is not a numerical parameter, but a function of the input size, so the exact comparison is difficult.

$s + O(\log s) + O(|x|)$ *space working on pair x, s.*

Note that since the allowed space overhead is $O(\log s) + O(|x|)$, there is no problem of keeping x and the binary representation of s in the memory. Note also that after we checked the termination, we can restart the computation of M and get the output of M on x within the same space bound.

Proof sketch. It is convenient to use the model with two stacks (though the result is valid for ordinary multitape machines since it allows $O(\log s)$ overhead). We may assume without loss of generality that machine M clears its stacks when terminates (the old final state is now a cleaning state that pops elements from the stacks until both are empty). Let us consider all configurations of M that use space at most s. We need to check whether a (unique) path starting from the initial configuration gets into the final one. We can do it backwards. Consider the tree whose root is the (unique) final configuration and the children of vertex S are configurations that are transformed to S in one step. Since M is deterministic, each vertex has only one parent. We get a tree of all configurations leading to termination (configurations belonging to loops are not in the tree). The termination question can be now reformulated as follows: is the initial configuration in the tree? To answer this question, one can traverse the tree in a one of the usual ways: depth-first search. Note that the standard (non-recursive) algorithm for this (see, e.g., the textbook [10, Chapter 3]) does not use any additional memory, and the basic operation (going to the parent, going to the oldest child, going to the next sibling, etc.) can be performed with $O(1)$ space. In our case we also need to keep track of the configuration size (since we do not consider configurations that require more than s space), but this can be done in $O(\log s)$ memory, and the comparison with the initial configuration requires $O(|x|)$ memory. □

Sipser's trick allows us to check whether a given pair $\langle x', y' \rangle$ has s-bounded complexity at most m in space $s + O(\log s) + O(|x'| + |y'|) + O(m)$ (by trying all programs up to size m and checking whether they produce this pair in space s). Then, using the argument described above, we get (1) for $s' = s + O(\log s) + O(|x| + |y|)$ and with an additional term $O(\log s)$ in the right hand side. Note that we may assume that $m = \mathrm{C}(x, y) = O(|x| + |y|)$, otherwise (1) is true for trivial reasons.

To get rid of the $O(\log s)$ term we need to change the enumeration order. Instead of enumerating for some fixed s all pairs $\langle x, y' \rangle$ such that $\mathrm{C}^s(x, y') \leqslant m$, we do it sequentially for $s = 1, 2, 3, \ldots$. So all pairs with $\mathrm{C}(x, y') \leqslant m$ will be enumerated at some stage (but to choose the value of k we still count the pairs such that $\mathrm{C}^s(x, y') \leqslant m$ for the *given* value of s). For each s we skip the pairs $\langle x, y' \rangle$ that were enumerated with space $s - 1$, so every pair is enumerated only once. To do this, during the enumeration process we perform the space-bound complexity check twice: for s and $s - 1$. Note that the space used for the checks can be reused, so we do not need more space for this. Now the enumeration process potentially requires unbounded space, but since we wait for the pair $\langle x, y \rangle$ (knowing its ordinal number and waiting until the element with this ordinal number appears), the actual space used in the process will be still $s + O(\log s) + (|x| + |y|)$. Indeed, this pair will appear at the stage while current space bound reaches s (or earlier).

The same reasoning works for $\mathrm{C}(x)$: we enumerate elements x' that have large sections (have many y' such that $K^s(x', y') \leqslant m$), and do it sequentially for $s = 0, 1, 2, \ldots$, omitting elements that appeared already for space bound $s - 1$. Again we enumerate all x' that have large sections for unbounded complexity, using more and more space, and again x appears at the stage when the current space bound is s (or less), and at this stage the space used by the computation is $s + O(\log s) + O(|x| + |y|)$.

Theorem 1 is proven. □

3 Basic inequalities: space-bounded version

We have defined space-bounded complexity for pairs. In the same way (and with the same precision) one can define the complexity of triples, and, in general, m-tuples for every fixed m. In the section we prove space-bounded versions of the so-called *basic inequalities* for Kolmogorov complexity.

The basic inequality involves complexities of triples and says (in the unbounded version) that

$$\mathrm{C}(x)+\mathrm{C}(x,y,z)\leqslant \mathrm{C}(x,y)+\mathrm{C}(x,z)+O(\log n)$$

if x,y,z are strings of length at most n. Usually it is proved by considering conditional complexities:

$$\begin{aligned}\mathrm{C}(x,y)&=\mathrm{C}(x)+\mathrm{C}(y|x)+O(\log n),\\ \mathrm{C}(x,z)&=\mathrm{C}(x)+\mathrm{C}(y|x)+O(\log n),\\ \mathrm{C}(x,y,z)&=\mathrm{C}(x)+\mathrm{C}(y,z|x)+O(\log n).\end{aligned}$$

Then the inequality can be rewritten as

$$\mathrm{C}(y,z|x)\leqslant \mathrm{C}(y|x)+\mathrm{C}(z|x)+O(\log n),$$

and this is a relativized version of the inequality for the complexity of pairs. Let us do this is more detail to see how the space-bounded version can be proven. We have

$$\begin{aligned}\mathrm{C}^{s'}(x)+\mathrm{C}^{s'}(y|x)&\leqslant \mathrm{C}^{s}(x,y),\\ \mathrm{C}^{s'}(x)+\mathrm{C}^{s'}(z|x)&\leqslant \mathrm{C}^{s}(x,z),\end{aligned}$$

for some s' slightly larger than s (we omit the logarithmic terms $O(\log n)$ in the inequalities), so

$$2\,\mathrm{C}^{s'}(x)+\mathrm{C}^{s'}(y|x)+\mathrm{C}^{s'}(z|x)\leqslant \mathrm{C}^{s}(x,y)+\mathrm{C}^{s}(x,z).$$

From this we conclude that

$$2\,\mathrm{C}^{s'}(x)+\mathrm{C}^{s''}(y,z|x)\leqslant \mathrm{C}^{s}(x,y)+\mathrm{C}^{s}(x,z),$$

for some s'' slightly exceeding s', and then

$$\mathrm{C}^{s'}(x)+\mathrm{C}^{s'''}(x,y,z)\leqslant \mathrm{C}^{s}(x,y)+\mathrm{C}^{s}(x,z).$$

For uniformity we can replace s' by s''' on the left-hand side. Here s''' is the third iteration of adding overhead, so still

$$s'''=s+O(\log s)+O(n),$$

and we get the following space-bounded version of basic inequality:

Theorem 2 (Space-bounded basic inequality)**.**

$$\mathrm{C}^{s'}(x)+\mathrm{C}^{s'}(x,y,z)\leqslant \mathrm{C}^{s}(x,y)+\mathrm{C}^{s}(x,z)+O(\log|x|+\log|y|+\log|z|)$$

for all strings x,y, for all numbers s, and for

$$s'=s+O(\log s)+O(|x|+|y|+|z|).$$

(*The constants in the O-notation do not depend on x,y,s.*)

More general inequalities (called also basic) are obtained if we replace x,y,z by tuples of strings; they are easy corollaries of Theorem 2 (converting the tuples into their string encoding and vice versa can be done in $O(n)$ space for strings of size at most n).

4 Shannon inequalities: iterations

Fix some integer $k \geqslant 1$; let $x_1, \ldots, x_k$ be some strings. For each $I \subset \{1, \ldots, k\}$ we consider the tuple x_I made of strings x_i with $i \in I$. In this notation, the basic inequalities mentioned above can be written as

$$\mathrm{C}^{s'}(x_{I\cap J}) + \mathrm{C}^{s'}(x_{I\cup J}) \leqslant \mathrm{C}^{s}(x_I) + \mathrm{C}^{s}(x_J) + O(\log n),$$

if all $x_1, \ldots, x_k$ are strings of length at most n and $s' = s + O(\log s) + O(n)$. (The constants in the O-notation may depend on k, I, J but not on n, $x_1, \ldots, x_k$, s.)

Taking the sum of several basic inequalities (for the same k, but for different I and J), we may get other linear inequalities for the complexities of tuples, i.e., inequalities of the type

$$\sum_{I\subset\{1,\ldots,k\}} \lambda_I \mathrm{C}(x_I) \geqslant 0$$

where λ_I are some real coefficients. This is a well known procedure for unbounded Kolmogorov complexity [11, Chapter 10]; the resulting linear inequalities are called *Shannon inequailites*. Not all linear inequalities that are true with logarithmic precision are Shannon inequalities (an important discovery made in [15]).

In this section we show that every Shannon inequality has a space-bounded version. To formulate this version, let us separate the positive and negative coefficients in the linear inequality (the corresponding groups are $L, R \subset \{1, \ldots, k\}$; we assume that $L \cap R = \varnothing$). Now the general form of a linear inequality for complexities of tuples is

$$\sum_{I\in L} \lambda_I \mathrm{C}(x_I) \leqslant \sum_{J\in R} \mu_J \mathrm{C}(x_J) \tag{2}$$

where all λ_I and μ_J are non-negative. The following theorem says that each Shannon inequality has a space-bounded counterpart of the same form as for the basic inequalities (but with slightly weaker space bound).

Theorem 3. *Consider a linear inequality of the form* (2) *that is a linear combination of basic inequalities* (*is a Shannon inequality*)*. Then the following space-bounded version of this inequality is true:*

$$\sum_{I\in L} \lambda_I \mathrm{C}^{s'}(x_I) \leqslant \sum_{J\in R} \mu_J \mathrm{C}^{s}(x_J) + O(\log n), \tag{3}$$

if $x_1, \ldots, x_k$ are strings of length at most n, and $s' = s + O(n^2 \log n) + O(n \log s)$.

Here the constants in the O-notation depend on the inequality (and k), but neither on n nor on $x_1, \ldots, x_k$.

Proof sketch. One could just add up the space-bounded versions of the basic inequalities to get the desired inequality, but the problem is that in the resulting inequality of type (3), L and R are not disjoint: the same complexity may appear on both sides. For the unbounded complexities, these terms just cancel each other, and we get the desired Shannon inequality. However, by adding the space-bounded versions of the same basic inequality, we get an inequality where the complexity of the same tuple may appear with the same coefficient on both sides, but with different space bounds:

$$\sum_{I\in L} \lambda_I \mathrm{C}^{s'}(x_I) + \sum_{K\in C} \sigma_K \mathrm{C}^{s'}(x_K) \leqslant \sum_{J\in R} \mu_J \mathrm{C}^{s}(x_J) + \sum_{K\in C} \sigma_K \mathrm{C}^{s}(x_K) + O(\log n),$$

Here x_K are tuples that appear on both sides with the same coefficients σ_K (the terms that are canceled in the unbounded version), but now they have bound s' on the left-hand side and s on the right-hand side, and cannot be canceled anymore.

The following trick helps. Let $f(s) = s + O(\log s) + O(\log n)$ be the function that transforms the right-hand side bound s to the left-hand side bound s' (here s is a variable, while n and the constants in the $O(\cdot)$-notation are fixed). Consider the sequence of space bounds

$$u_0 = s, u_1 = f(u_0), \ldots, u_N = f(u_{N-1})$$

for some large N. All tuple complexities can only decrease if we increase the space bound from u_k to u_{k+1}. Therefore, for a large enough N, namely, $N = O(n)$ with a large enough constant, we guarantee the existence of k such that all complexities of tuples are the same with bounds u_k and u_{k+1}. Then we can add the space-bounded inequalities and cancel the common terms as we did for the unbounded version. More precisely, we get

$$\sum_{I\in L} \lambda_I \mathrm{C}^{u_{k+1}}(x_I) + \sum_{K\in C} \sigma_K \mathrm{C}^{u_{k+1}}(x_K) \leqslant \sum_{J\in R} \mu_J \mathrm{C}^{u_k}(x_J) + \sum_{K\in C} \sigma_K \mathrm{C}^{u_k}(x_K) + O(\log n),$$

and on both sides u_k can be replaced by u_{k+1} due to our assumption. So we can cancel the common terms. We do not know k for which there is no change in the complexities, but we can replace the bound on the left-hand side by u_N, and on the right-hand side by $s = u_0$.

It remains to show that for $N = O(n)$ the nth iteration of function f started with s gives us at most $s + O(n^2 \log n) + O(n^2 \log s)$.

The required bound is an easy corollary of the following lemma:

Lemma 1. *Let $f(s) = s + \log s + k$, and $f^{(n)}(s)$ be the nth iteration of f, i.e., $f^{(2)}(s) = f(f(s))$, etc. Then*

$$f^{(n)}(s) \leqslant s + n\log s + O(kn\log n) + O(n\log n) + O(1)$$

for all integers $n \geqslant 1$, for all $s \geqslant 1$ and $k \geqslant 0$.

Proof sketch. This is a simple calculus exercise; we will prove the inequality

$$f^{(n)}(s) \leqslant s + n\log s + c_1(k+1)(n+c_2)\ln(n+c_2)$$

for some $c_1, c_2 > 0$ and for all $n, s \geqslant 1, k \geqslant 0$. Denote the right hand side by $F(n)$; to perform the induction over n, it is enough to show that $F(n+1) \geqslant F(n) + \log F(n) + k$; note that both f and F are monotone. And for that it is enough to choose c_1 and c_2 in such a way that $F'(n) \geqslant \log F(n) + k$ for all n (now n is not necessarily an integer, and we take derivative over n).

Let us see what is needed for that. We need (recall that $(x\ln x)' = 1 + \ln x$):

$$F'(n) = \log s + c_1(k+1) + c_1(k+1)\ln(n+c_2) \geqslant \\ \geqslant k + \log\big(s + n\log s + c_1(k+1)(n+c_2)\ln(n+c_2)\big).$$

Moving the term $\log s$ on the right-hand side and putting it inside the logarithm, we need to prove that

$$c_1(k+1) + c_1(k+1)\ln(n+c_2) \geqslant k + \log\left(1 + \frac{n}{s}\log s + \frac{c_1(k+1)(n+c_2)}{s}\ln(n+c_2)\right).$$

Note that $\log(a+b+c) \leqslant \log(3\max(a,b,c)) = \log 3 + \max(\log a, \log b, \log c)$. So we may show separately that logarithms of all three terms in the square brackets are small compared to the

left hand side for large c_1, c_2; it is enough since k is also small compared to $c_1(k+1)$ for large c_1. For $\log 1$ and for $\log\left(\frac{n}{s}\log s\right)$ this is obvious, since we have the term $c_1\log(n+c_2)$ in the left-hand side, and c_1 is large. It remains to show that

$$c_1(k+1)+c_1(k+1)\ln(n+c_2) \gg \log\left(c_1(k+1)(n+c_2)\ln(n+c_2)/s\right) = \\ = \log c_1 + \log(k+1) + \log(n+c_2) + \log\ln(n+c_2) - \log s$$

for sufficiently large c_1 and c_2. And this is also simple: each positive term in the right-hand side is much smaller than the left-hand side (and minus $\log s$ can be omitted). □

It remains to note that if $f(s) = s + c\log s + k$ then each iteration of f can be replaced by c iterations of the function $s \mapsto s + \log s + k/c$, so for n iterations we have bound $f^{(n)}(s) \leqslant s + cn\log s + O\big((k/c+1)cn\log cn\big)$. This gives the required bound (we have constant c and $k = O(n)$ in our argument). The Lemma (and Theorem 3) are proven. □

5 General result

In this section we use a similar technique to prove a more general result that covers not only Shannon inequalities but all true linear inequalities for Kolmogorov complexity. Recall that a theorem from Hammer et al. ([3], see [11, Chapter 10] for the detailed exposition) says that the same linear inequalities are true for complexities (with logarithmic precision) and for Shannon entropies. In this section we want to show that all inequalities in this class have space-bounded counterparts. For that, we need to modify the original proof using the tools we developed.

Theorem 4 (General linear inequalities with space bounds)**.** *Assume that a linear inequality for unbounded complexities*

$$\sum_{I\in L} \lambda_I \mathrm{C}(x_I) \leqslant \sum_{J\in R} \mu_J \mathrm{C}(x_J) + O(\log n) \tag{4}$$

is true for all strings $x_1, \ldots, x_k$ of length at most n. Then its space-bounded version

$$\sum_{I\in L} \lambda_I \mathrm{C}^{s'}(x_I) \leqslant \sum_{J\in R} \mu_J \mathrm{C}^{s}(x_J) + O(\log n) \tag{5}$$

holds for all strings $x_1, \ldots, x_k$ of length at most n and for $s' = s + O(n^2\log n) + O(n\log s)$.

Proof sketch. Recall the original argument used in [3]. Given a tuple $x = \langle x_1, \ldots, x_k\rangle$ of strings, we consider the set X of all tuples $\langle x'_1, \ldots, x'_k\rangle$ such that $\mathrm{C}(x'_1, \ldots, x'_k) \leqslant \mathrm{C}(x_1, \ldots, x_k)$, and, moreover,

$$\mathrm{C}(x'_I | x'_J) \leqslant \mathrm{C}(x_I | x_J)$$

for all disjoint subsets $I, J \subset \{1, \ldots, k\}$. The logsize of X is bounded by $\mathrm{C}(x_1, \ldots, x_k) + O(1)$; on the other hand, X cannot be smaller than $\mathrm{C}(x_1, \ldots, x_k) - O(\log n)$, since X can be enumerated if we know the numerical values of the complexities, and x (that is in X by construction) can be described by the ordinal number in this enumeration. Then we show that X is almost uniform (i.e., the average size of its sections is close to the maximal size of the sections of the same type, and the same for projections and their sections). The argument is based on the following observation: the size of the set X is bounded by the size of its x_1-projection, times the size of the maximal x_1-section of the (x_1, x_2)-projection, times the size of the maximal x_3-section of the (x_1, x_2, x_3)-projection, etc. The logarithms of these sizes are bounded by

$$\mathrm{C}(x_1), \mathrm{C}(x_2|x_1), \mathrm{C}(x_3|x_1, x_2), \ldots$$

(respectively), and the sum of these complexities is $O(\log n)$-close to $\mathrm{C}(x_1,\dots,x_k)$. Since we know that the logsize of X also equals $\mathrm{C}(x_1,\dots,x_k)$ with $O(\log n)$-precision, we conclude that all inequalities for sizes in this chain are close to equalities with logarithmic precision. Then we may conclude that for a random k-tuple of variables uniformly distributed in X, the entropy of each of its projections is close to the logsize of the same projection of X and to the complexity of the corresponding tuple x_T, and we can use the inequality for entropies to derive the inequality for complexities. This is a (very rough) sketch of the proof from [3]; see [11] for details.

Now we have to adapt this proof for the space-bounded complexity. We define the set X^u in the same way as X, but with space bound u, i.e. it consists of all tuples $\langle x'_1,\dots,x'_k\rangle$ such that

$$\mathrm{C}(x'_I|x'_J) \leqslant \mathrm{C}^u(x_I|x_J)$$

for all disjoint subsets $I,J\subset\{1,\dots,k\}$. There are two points in the above argument that need to be adapted to this change. First, to show that X^u is large, we need to specify how X^u is enumerated and how much space is needed when we describe x by its number in this enumeration. For the enumeration, we take $s=1,2,\dots$ and for each s enumerate the elements of X^u that satisfy (stronger) conditions

$$\mathrm{C}^s(x'_I|x'_J) \leqslant \mathrm{C}^u(x_I|x_J)$$

but did not appear at the previous stage (for $s-1$), using the same tools as before. The original tuple x appears for $s\leqslant u$, and the space used by enumeration at this moment is $u'=u+O(\log u)+O(n)$, so we get the upper bound for the complexity $\mathrm{C}^{u'}(x)$. To make the argument working we need to have complexities with space bounds u and u' equal.[7]

Also, we used the formulas for the complexities of pairs that now also involve different space bounds u and $u'=u+O(\log u)+O(n)$. To deal with both problems, we use the same iteration trick and note that at some step when u increases, the complexities do not change. For that we need $O(n)$ iterations, since the complexity of all strings and tuples involved can only decrease by $O(n)$ in total. Finally, we get the same bound as in the previous theorem for Shannon inequalities. □

6 Remarks

Space-bounded versions of other results. Our results are part of the space-bounded version of algorithmic information theory. In general, one could take some notion or theorem of algorithmic information theory and look for its space-bounded counterpart. For Muchnik's conditional codes theorem this was done by Musatov (see [8] and references therein).

However, there are many problems in this approach. For example, if we define mutual information with space bound s in a natural way as

$$I^s(a:b)=\mathrm{C}^s(a)-\mathrm{C}^s(a|b),$$

this notion is not monotone; a priori the mutual information can oscillate when s increases. It would be interesting to understand what kinds of oscillations are possible. Is it possible that two strings are mutually independent for some space bound, then dependent for some larger bound, then again independent, and so on? Also the relations between $I^s(a:b)$, $I^s(b:a)$ and the symmetric expression $\mathrm{C}^s(a)+\mathrm{C}^s(b)-\mathrm{C}^s(a,b)$ are unclear.

[7] It is enough to have them close, but this does not improve the resulting bound. Another observation: in this argument we could use instead of X^u its part that consists of tuples that satisfy a stronger conditions $\mathrm{C}^u(x'_I|x'_J)\leqslant \mathrm{C}^u(x_I|x_J)$ for all I,J.

Inequalities for a common space bound. Our proof of Theorem 4 gives a bit more: it guarantees for every s there is a bigger (but not much bigger) s' for which the inequality in question is true when all complexities use space bound s'.

More formally, assume that a linear inequality for unbounded complexities

$$\sum_{I\in L} \lambda_I \mathrm{C}(x_I) \leqslant O(\log n)$$

is true for all n and for all strings $x_1,\dots,x_k$ of length at most n. Let $s \geqslant n$ be some space bound. Then the space-bounded version

$$\sum_{I\in L} \lambda_I \mathrm{C}^{s'}(x_I) \leqslant O(\log n)$$

holds for all strings $x_1,\dots,x_k$ of length at most n and for some $s' = s + O(n^2 \log n) + O(n \log s)$, depending on n and $x_1,\dots,x_k$.

Time-bounded versions. We can try a similar approach for time bounds (instead of space bounds). It also works, but the natural bound in the formula for complexity of pairs multiplies the time complexity by $2^{O(n)}$; also the simulation would increase time significantly (for a one-tape machine the simulation of t steps needs more than t^2 time). When we iterate these bounds $O(n)$ times, we get ridiculously high time bounds — so it is a good luck that Sipser's trick for space bounds allows us to get some reasonable space bounds.

References

[1] Terence H. Chan, A combinatorial approach to information inequalities, *Communications in Informations and Systems*, **1**(3), 241–254 (September 2001, preliminary version in 1999)

[2] Terence H. Chan, Raymond W. Yeung, On a relation between information inequalities and group theory, *IEEE Transactions on Information Theory*, **IT-48**(7), 1992–1995 (July 2002, preliminary version in 1999)

[3] Daniel Hammer, Andrei Romashchenko, Alexander Shen, Nikolai Vereshchagin, Inequalities for Shannon Entropies and Kolmogorov Complexities. *Proceedings 12th IEEE conference on Computational Complexity*, Ulm, 1997, 13–23. Final version: *Journal of Computer and System Sciences*, **60**, 442–464.

[4] A.N. Kolmogorov, Three approaches to the quantitative definition of information, *Problems of Information Transmission*, **1**, 1–7 (1965)

[5] A.N. Kolmogorov, Logical basis for information theory and probability theory, *IEEE Transaction on Information Theory*, **14**, 662–664 (1968).

[6] Luc Longpré, *Resource bounded Kolmogorov complexity, a link between computational complexity and information theory*, Ph. D. thesis, TR 86-776 (1986), 115 pp. Dept. of Computer Science, Cornell University, Ithaca, NY 14583, `https://ecommons.cornell.edu/handle/1813/6616`

[7] Luc Longpré, Sarah Mocas, Symmetry of information and one-way functions, *Information processing letters*, **46**(2), 95–100 (1993)

[8] Daniil Musatov, *Improving the space-bounded version of Muchnik's conditional complexity theory via naive derandomization*, *Theory of Computing Systems*, **55**, 299–312 (2014), see also `https://arxiv.org/abs/1009.5108`

[9] Andrei Romashchenko, Alexander Shen, Nikolai Vereshchagin, Combinatorial interpretation of Kolmogorov complexity, *Proceedings 15th Annual IEEE Conference on Computational Complexity*, Florence, Italy, 2000, 131–137`https://doi.org/10.1109/CCC.2000.856743`. Final version: *Theoretical Computer Science*, **271**(1–2), 111-123 (2002).

[10] Alexander Shen, *Algorithms and programming: problems and solutions*, 2nd ed., Springer, 2010.

[11] Alexander Shen, Vladimir A. Uspensky, Nikolai Vereshchagin, *Kolmogorov complexity and algorithmic randomness*, AMS, 2018.

[12] Michael Sipser, Halting space-bounded computations, *Theoretical Computer Science*, **10** (1980), 335–338.

[13] Nikolai Vereshchagin, Alexander Shen, Algoritmic statistics revisited, in *Measures of Complexity. Festschrift for Alexey Chervonenkis.* Springer, 2015, 235–252.

[14] Nikolai Vereshchagin, Alexander Shen, Algoritmic statistics: forty year later. *Computability and Complexity. Essays Dedicated to Rodney G. Downey on the Occasion of His 60th Birthday*. Springer, 2017, 669–737.

[15] Zhen Zhang, Raymond W. Yeung, On characterization of entropy function via information inequalities, *IEEE Transactions in Information Theory*, **IT-44**(4), 1440–1452 (1998)