10.1090/surv/220/03

CHAPTER 2

Complexity of pairs and conditional complexity

2.1. Complexity of pairs

As we have discussed, we can define complexity of any constructive object using (computable) encodings by strings. In this section we deal with pairs of strings. A pair x, y can be encoded, e.g., by a string $[x, y] = \overline{x}01y$; here $\overline{x}$ stands for x with doubled bits. Any other computable encoding $x, y \mapsto [x, y]$ could be used (of course, we need that $[x, y] \neq [x', y']$ if $x \neq x'$ or $y \neq y'$). Any two encodings of this type are equivalent (there are translation algorithms in both directions), so Theorem 3 (p. 5) guarantees that complexities of the different encodings of the same pair differ by $O(1)$.

Let us fix some encoding $[x, y]$. The *Kolmogorov complexity of a pair x, y* is defined as $C([x, y])$ and is denoted by $C(x, y)$. Here are some evident properties:

- $C(x, x) = C(x) + O(1)$;
- $C(x, y) = C(y, x) + O(1)$;
- $C(x) \leqslant C(x, y) + O(1)$; $C(y) \leqslant C(x, y) + O(1)$.

The following theorem gives an upper bound for the complexity of a pair in terms of complexities of its components:

THEOREM 16.

$$C(x, y) \leqslant C(x) + 2\log C(x) + C(y) + O(1);$$
$$C(x, y) \leqslant C(x) + \log C(x) + 2\log\log C(x) + C(y) + O(1);$$
$$C(x, y) \leqslant C(x) + \log C(x) + \log\log C(x) + 2\log\log\log C(x) + C(y) + O(1);$$
$$\ldots .$$

(We can continue this sequence of inequalities indefinitely. Also, one can exchange x and y.)

PROOF. This proof (for the first inequality) was already explained in the introduction (Theorem 4, p. 6). The only difference is that we considered the concatenation xy instead of a pair. Let us repeat the argument for pairs.

A computable mapping $x \mapsto \hat{x}$ (here x and $\hat{x}$ are binary strings) is called a *prefix-free encoding*, if for any two different strings x and y the string $\hat{x}$ is not a prefix of the string $\hat{y}$. (In particular, $\hat{x} \neq \hat{y}$ if $x \neq y$.) This guarantees that both u and v can be uniquely reconstructed from $\hat{u}v$.

An example of a prefix-free encoding is $x \mapsto \overline{x}01$, where $\overline{x}$ stands for x with doubled bits. Here the block 01 is used as a delimiter. However, this encoding is not the most space-efficient one, since it doubles the length. A better prefix-free encoding is

$$x \mapsto \hat{x} = \overline{\mathrm{bin}(l(x))}\,01x,$$

where $(\mathrm{bin}(l(x))$ is the binary representation of the length $l(x)$ of the string x). Now

$$l(\hat{x}) = l(x) + 2\log l(x) + O(1).$$

This trick can be *iterated*: for any prefix-free encoding $x \mapsto \hat{x}$, we can construct another prefix-free encoding

$$x \mapsto \widehat{\mathrm{bin}(l(x))}x.$$

Indeed, if $\widehat{\mathrm{bin}(l(x))}x$ is a prefix of $\widehat{\mathrm{bin}(l(y))}y$, then one of the strings $\widehat{\mathrm{bin}(l(x))}$ and $\widehat{\mathrm{bin}(l(y))}$ is a prefix of the other one, and therefore $\mathrm{bin}(l(x)) = \mathrm{bin}(l(y))$. Therefore x is a prefix of y, and $l(x) = l(y)$, so $x = y$. (In other words, we uniquely determine the length of the string, since a prefix-free code is used for it, and we then get the string itself knowing where it ends.)

In this way we get a prefix-free encoding such that

$$l(\hat{x}) = l(x) + \log l(x) + 2\log\log l(x) + O(1),$$

then (one more iteration)

$$l(\hat{x}) = l(x) + \log l(x) + \log\log l(x) + 2\log\log\log l(x) + O(1),$$

etc.

Now we return to the proof. Let D be the optimal decompressor used in the definition of Kolmogorov complexity. Consider a decompressor D' defined as

$$D'(\hat{p}q) = [D(p), D(q)],$$

where $\hat{p}$ is a prefix-free encoding and $[\cdot,\cdot]$ is the encoding of pairs (used in the definition of pairs complexity). Since $\hat{p}$ is a prefix-free encoding, D' is well defined (we can uniquely extract $\hat{p}$ out of $\hat{p}q$).

Let p and q be the shortest descriptions of x and y. Then $\hat{p}q$ is a description of $[x, y]$, and its length is exactly as we need in our theorem. (The more iterations we use for the prefix-free encoding, the better bound we get.) $\square$

Theorem 16 implies that

$$C(x, y) \leqslant C(x) + C(y) + O(\log n)$$

for strings x and y of length at most n: one may say that the complexity of a pair does not exceed the sum of the complexities of its component with logarithmic precision.

$\boxed{\mathbf{16}}$ Suggest a natural definition for the complexity of a triple. Show that $C(x, y, z) \leqslant C(x) + C(y) + C(z) + O(\log n)$ for every three strings x, y, z of length at most n.

A natural question arises: is it true that $C(x, y) \leqslant C(x) + C(y) + O(1)$?

A simple argument shows that this is not the case. Indeed, this inequality would imply $C(x, y) \leqslant l(x) + l(y) + O(1)$. Consider some N. For each $n = 0, 1, 2, \dots, N$, we have 2^n strings x of length n and 2^{N-n} strings y of length $N - n$. Combining them, we (for a given n) obtain 2^N different pairs $\langle x, y\rangle$. The total number of pairs (all $n = 0, 1, \dots, N$ give different pairs) is $(N + 1)2^N$.

Assume that indeed $C(x, y) \leqslant l(x) + l(y) + O(1) = N + O(1)$ for all these pairs. Then we get $(N + 1)2^N$ different strings $[x, y]$ of complexity at most $N + O(1)$, but this is impossible (Theorem 7, p. 17, gives the upper bound $O(2^N)$).

17 Prove that there is no constant c such that

$$C(x,y) \leqslant C(x) + \log C(x) + C(y) + c$$

for all x and y.

(*Hint*: Replace C in the right-hand side by l and count the number of corresponding pairs.)

18 (a) Prove that

$$\sum_{x\in\Xi} 2^{-l(\hat{x})} \leqslant 1$$

for any prefix-free encoding $x \mapsto \hat{x}$ (here Ξ is the set of all binary strings).

(b) Prove that if a prefix-free encoding increases the length of an n-bit string at most by $f(n)$, i.e., if $l(\hat{x}) \leqslant l(x) + f(l(x))$, then $\sum_n 2^{-f(n)} < \infty$.

This problem explains why a coefficient 2 appears in Theorem 16 (p. 31): the series

$$\sum \frac{1}{n^2}, \quad \sum \frac{1}{n(\log n)^2}, \quad \sum \frac{1}{n \log n (\log\log n)^2}, \dots$$

converge, while the series

$$\sum \frac{1}{n}, \quad \sum \frac{1}{n \log n}, \quad \sum \frac{1}{n \log n \log\log n}, \dots$$

diverge.

The following problem describes functions that can be used for bounds similar to Theorem 16.

19 Let $f\colon \mathbb{N} \to \mathbb{N}$ be a non-decreasing total computable function. Prove that the following three properties are equivalent:

(a) $C(x,y) \leqslant C(x) + C(y) + f(C(x)) + O(1)$;

(b) $C(x,y) \leqslant l(x) + l(y) + f(l(x)) + O(1)$;

(c) $\sum_n 2^{-f(n)} < \infty$.

(*Hint*: (a) obviously implies (b); to get the reverse implication, consider the shortest descriptions. To derive (a) from (c), one can count pairs with $l(x) + f(l(x)) + l(y) < n$; one can also use results about prefix complexity (see Chapter 4, Problem 107). Finally, to derive (c) from (b), note that the right-hand side in (b) is at most $n + O(1)$ if $l(x) = k$ and $l(y) = n - k - f(k)$, for $k + f(k) \leqslant n$. So the number of such pairs is at least $\sum 2^k 2^{n-k-f(k)} = 2^n \sum_k 2^{-f(k)}$ where the sum is taken over all k such that $k + f(k) \leqslant n$.)

20 Prove that all the inequalities of Theorem 16 become false if the coefficient 2 is replaced by 1 but remain true with the coefficient $1 + \varepsilon$ for any $\varepsilon > 0$.

(*Hint*: See the preceding problem.)

21 Prove that

$$C(x,y) \leqslant C(x) + \log C(x) + C(y) + \log C(y) + O(1).$$

22 (Continued) Prove a stronger inequality:

$$C(x,y) \leqslant C(x) + C(y) + \log(C(x) + C(y)) + O(1).$$

(Note that $C(x) + C(y)$ can be replaced by $\max(C(x), C(y))$. This gives a factor at most 2, which makes $O(1)$ after taking logarithms.)

$\boxed{\mathbf{23}}$ Prove that $C(x, C(x)) = C(x) + O(1)$.

(*Hint*: $C(x, C(x)) \geqslant C(x) + O(1)$ for evident reasons. On the other hand, the shortest description of x determines both x and $C(x)$.)

$\boxed{\mathbf{24}}$ Prove that if $C(x) \leqslant n$ and $C(y) \leqslant n$, then $C(x, y) \leqslant 2n + O(1)$.

2.2. Conditional complexity

When transmitting a file, one could try to save communication charges by compressing that file. The transmission could be made even more effective if an old version of the same file already exists at the other side. In this case we need only describe the changes made. This could be considered as a kind of motivation for the definition of conditional complexity of a given string x relative to a (known) string y.

A *conditional decompressor* is a computable function D of two arguments, the *description* and the *condition* (both arguments and the value of D are binary strings). If $D(y, z) = x$, we say that y is a (conditional) *description of x when z is known* (or *relative to z*). The complexity $C_D(x \,|\, z)$ is then defined as the length of the shortest conditional description:

$$C_D(x \,|\, z) = \min\{l(y) \mid D(y, z) = x\}.$$

We say that (conditional) decompressor D_1 is *not worse* than D_2 if

$$C_{D_1}(x \,|\, z) \leqslant C_{D_2}(x \,|\, z) + c$$

for some constant c and for all x and z. A conditional decompressor is *optimal* if it is not worse than any other conditional decompressor.

THEOREM 17. *There exist optimal conditional decompressors.*

PROOF. This conditional version of the Solomonoff– Kolmogorov theorem can be proved in the same way as the unconditional one (Theorem 1, p. 3).

Fix some programming language where one can write programs for computable functions of two arguments, and let

$$D(\hat{p}y, z) = p(y, z),$$

where $p(y, z)$ is the output of program p on inputs y and z, and $\hat{p}$ is the prefix-free encoding of p.

It is easy to see now that if D' is a conditional decompressor and p is a program for D', then

$$C_D(x \,|\, z) \leqslant C_{D'}(x \,|\, z) + l(\hat{p}).$$

The theorem is proved. □

Again, we fix some optimal conditional decompressor D and omit index D in the notation.

Let us start with some simple properties of conditional complexity.

THEOREM 18.

$$\begin{aligned} C(x \,|\, y) &\leqslant C(x) + O(1); \\ C(x \,|\, x) &= O(1); \\ C(f(x, y) \,|\, y) &\leqslant C(x \,|\, y) + O(1); \\ C(x \,|\, y) &\leqslant C(x \,|\, g(y)) + O(1). \end{aligned}$$

Here g and f are arbitrary computable functions (of one and two arguments, respectively) and the inequalities are valid if $f(x,y)$ and $g(y)$ are defined.

Proof. First inequality: Any unconditional decompressor can be considered as a conditional one that ignores the second argument.

Second inequality: Consider D such that $D(p,z)=z$.

Third inequality: Let D be the optimal conditional decompressor used to define complexity. Consider another decompressor D' such that

$$D'(p,y)=f(D(p,y),y),$$

and apply the optimality property.

A similar argument works for the last inequality, but D' should be defined in a different way:

$$D'(p,y)=D(p,g(y)).$$

The theorem is proven. □

$\boxed{\mathbf{25}}$ Prove that conditional complexity is "continuous as a function of its second argument": $C(x\,|\,y0)=C(x\,|\,y)+O(1)$; $C(x\,|\,y1)=C(x\,|\,y)+O(1)$. Using this property, show that for every string x and for every non-negative integer $l\leqslant C(x)$ there exists a string y such that $C(x\,|\,y)=l+O(1)$.

A similar argument based on two-dimensional topology is used in [**156**].

$\boxed{\mathbf{26}}$ Prove that for any fixed y the function $x\mapsto C(x\,|\,y)$ differs from C at most by $2C(y)+O(1)$.

$\boxed{\mathbf{27}}$ Prove that $C([x,z]\,|\,[y,z])\leqslant C(x\,|\,y)+O(1)$ for any strings x,y,z (here $[\cdot,\cdot]$ stands for the computable encoding of pairs).

$\boxed{\mathbf{28}}$ Fix some "reasonable" programming language. (Formally, we require the corresponding universal function to be a Gödel one. This means that a translation algorithm exists for any other programming language; see, e.g., [**184**].) Show that the conditional complexity $C(x\,|\,y)$ is equal (up to an $O(1)$ additive term) to the minimal complexity of a program that produces output x on input y.

(*Hint*: Let D be an optimal conditional decompressor. If we fix its first argument p, we get a program of complexity at most $l(p)+O(1)$. On the other hand, if program p maps y to x, then $C(x\,|\,y)=C(p(y)\,|\,y)\leqslant C(p)+O(1)$.)

This interpretation of conditional complexity as a minimal complexity of a program with some property will be considered in Chapter 13.

If we restrict ourselves to *total* programs (that terminate on all inputs), we get an essentially different notion of conditional complexity that can be called *total conditional complexity.*

$\boxed{\mathbf{29}}$ Show that the notion of total conditional complexity $CT(x\,|\,y)$, the minimal (plain) complexity of a total program that maps y to x, is well defined (i.e., it changes at most by $O(1)$ when we change the programming language in a reasonable way). Prove that

$$C(x\,|\,y)\leqslant CT(x\,|\,y)\leqslant C(x)$$

with $O(1)$-precision.

$\boxed{\mathbf{30}}$ Show that the total complexity sometimes exceeds significantly the usual conditional complexity: for every n there exist two n-bit strings x and y such that

$$C(x\,|\,y)=O(1)\text{ while }CT(x\,|\,y)\geqslant n.$$

(*Hint*: Let us enumerate all programs of complexity less than n defined on all n-bit strings, and maintain two n-bit strings x and y with the following property: none of the programs found maps y to x. When a new program is found that maps y to x, we choose a fresh value of y and then choose an appropriate x. This process is effective if n (=length of y) is given, it defines a partial function $y \mapsto x$, so $C(x \,|\, y) = O(1)$ for every pair selected.)

$\boxed{\mathbf{31}}$ Let x and y be bit strings such that $CT(x \,|\, y) \leqslant n$ and $CT(y \,|\, x) \leqslant n$. Prove that there exists a program of a computable permutation of the set of bit strings that maps x to y and has complexity at most $2n + O(1)$.

(*Hint*: It is easy to construct a string v of length $2n + O(1)$ that encodes a pair of total programs f that maps x to y and g that maps y to x. We may assume without loss of generality that x and y have 0 as their first bits. Consider a binary relation R on the set of strings that have first bit 0, defined as $R(u, v)$: $(f(u) = v)$ and $(g(v) = u)$). This is a decidable one-to-one correspondence between decidable sets of strings with infinite complements, and it can be easily extended to a computable permutation.)

$\boxed{\mathbf{32}}$ Show that the upper bound in the preceding problem cannot be improved significantly: for every k there are two strings x and y of length $n = 2k + O(1)$ such that $C(x), C(y) \leqslant k + O(1)$ (and therefore $CT(x \,|\, y), CT(y \,|\, x) \leqslant k + O(1)$), but every permutation of n-bit strings that maps x to y has complexity at least $2k$.

(*Hint*: Let us first select (arbitrarily) 2^k strings y and pair them with some string x. Let us enumerate computable permutations of n-bit strings that have complexity less than $2k$. If and when all selected pairs are served by some of these permutations, choose a new string x that is connected (by existing permutation) with at most half of the selected y-strings. After that $\Omega(2^k)$ new permutations are needed to connect new x to all y-strings. Therefore at most $2^{2k}/\Omega(2^k) = O(2^k)$ x-strings will be used, so the final x and y have complexity at most $k + O(1)$. The selection of x connected with at most half of selected y-strings is always possible, since each of the y-strings is connected with a small fraction of x-strings, and we can change the order of summation in the double sum. Note that this argument may be used to guarantee that one of the strings x and y belongs to a given set of 2^k strings.)

See [**136**] for the detailed proofs of these results about total conditional complexity.

Many properties of unconditional complexity have conditional counterparts with essentially the same proofs. Here are some of these counterparts.

- Function $C(\cdot \,|\, \cdot)$ is upper semicomputable (this means that the set of triples $\langle x, y, n \rangle$ such that $C(x \,|\, y) < n$ is enumerable).
- For any y and n the set of all strings x such that $C(x \,|\, y) < n$ has cardinality less then 2^n.
- For any y and n there exists a string x of length n such that $C(x \,|\, y) \geqslant n$.

$\boxed{\mathbf{33}}$ Prove that for any strings y and z and for any number n there exists a string x of length n such that $C(x \,|\, y) \geqslant n - 1$ and $C(x \,|\, z) \geqslant n - 1$.

(*Hint*: Both requirements are violated by a minority of strings.)

THEOREM 19. *Let $\langle x, y\rangle \mapsto k(x, y)$ be an upper semicomputable function such that the set $\{x \mid k(x, y) < n\}$ has cardinality less than 2^n for any string y and integer n. Then $C(x\,|\,y) \leqslant k(x, y) + c$ for some c and for all x and y.*

The proof repeats the proof of Theorem 8.

Using conditional complexity, we get a stronger inequality for the complexity of pairs (compared with Theorem 16, p. 31):

THEOREM 20.

$$C(x, y) \leqslant C(x) + 2\log C(x) + C(y\,|\,x) + O(1).$$

PROOF. Let D_1 be an optimal unconditional decompressor, and let D_2 be an optimal conditional decompressor. Construct a new unconditional decompressor D' as follows:

$$D'(\hat{p}q) = [D_1(p), D_2(q, D_1(p))].$$

Here $\hat{p}$ stands for the prefix-free encoding of p, and $[\cdot,\cdot]$ is a computable encoding of pairs used in the definition of the complexity of pairs. Let p be the shortest D_1-description of x, and let q be the shortest D_2-description of y conditional to x. Then the string $\hat{p}q$ is a D_3-description of $[x, y]$. Therefore,

$$C(x, y) \leqslant C_{D'}(x, y) + O(1) \leqslant l(\hat{p}) + l(q) + O(1).$$

As we have seen, one can choose a prefix-free encoding in such a way that $l(\hat{p})$ is bounded by $l(p) + 2\log l(p) + O(1)$ (see the proof of Theorem 16, p. 31), and we get a desired inequality. $\square$

As before, we may replace $2\log C(x)$ by $\log C(x) + 2\log\log C(x)$, etc., getting a better bound. We also can use conditional complexity in the logarithmic term and write

$$C(x, y) \leqslant C(x) + C(y\,|\,x) + 2\log C(y\,|\,x) + O(1).$$

(In the proof we should then replace $D'(\hat{p}q)$ by $D'(\hat{q}p)$.)

$\boxed{\mathbf{34}}$ Prove that

$$C(x\,|\,z) \leqslant C(x\,|\,y) + 2\log C(x\,|\,y) + C(y\,|\,z) + O(1)$$

for all x, y, z (a sort of triangle inequality).

If we are not interested in the exact form of the additional logarithmic term, the statement of Theorem 20 can be reformulated as

$$C(x, y) \leqslant C(x) + C(y\,|\,x) + O(\log n)$$

for all strings x, y of length at most n.

It turns out[1] that this inequality is in fact an equality.

THEOREM 21 (Kolmogorov–Levin).

$$C(x, y) = C(x) + C(y\,|\,x) + O(\log n)$$

for all strings x, y of length at most n.

[1]This is the first non-trivial statement in this chapter, and probably the first non-trivial result about Kolmogorov complexity; it was proven independently by Kolmogorov and Levin and published in [**79, 225**].

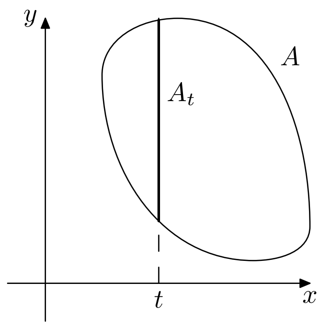

FIGURE 2. The section A_t of the set A of all simple pairs

PROOF. Since we already have one inequality, we need to prove only that

$$C(x, y) \geqslant C(x) + C(y\,|\,x) + O(\log n)$$

for all x and y of length at most n.

Let x and y be some strings of length at most n. Let a be the complexity $C(x, y)$ of the pair $\langle x, y\rangle$. Consider the set A of all pairs whose complexity does not exceed a. Then A is a set of cardinality $O(2^a)$ (in fact, at most 2^{a+1}) and $\langle x, y\rangle$ is one of its elements.

For each string t consider the "vertical section" A_t of A:

$$A_t = \{u \mid \langle t, u\rangle \in A\}$$

(see Figure 2). The sum of the cardinalities of all A_t (over all strings t) is the cardinality of A and does not exceed $O(2^a)$. Therefore there are few "large" sections A_t, and this is the basic argument we need for the proof.

Let m be equal to $\lfloor \log_2 |A_x| \rfloor$ where x is the first component of the pair $\langle x, y\rangle$ we started with. In other words, assume that cardinality of A_x is between 2^m and 2^{m+1}. Let us prove that

(1) $C(y\,|\,x)$ does not exceed m significantly;

(2) $C(x)$ does not exceed $a - m$ significantly.

We start with (1). Knowing a, we can enumerate the set A. If we know also x, we can select only pairs whose first component equals x. In this way we get an enumeration of A_x. To specify y, it is enough to determine the ordinal number of y in this enumeration (of A_x). This ordinal number takes $m+O(1)$ bits, and together with a we get $m + O(\log n)$ bits for the conditional description of y given x. Note that $a = C(x, y)$ does not exceed $O(n)$ for strings x and y of length n. Therefore, we need only $O(\log n)$ to specify a and n, and

$$C(y\,|\,x) \leqslant m + O(\log n).$$

Now let us prove (2). Consider the set B of all strings t such that the cardinality of A_t is at least 2^m. The cardinality of B does not exceed $2^{a+1}/2^m$; otherwise, the sum $|A| = \sum |A_t|$ would be greater than 2^{a+1}. We can enumerate B if we know a and m. Indeed, we should enumerate A and group together the pairs with the same first coordinate. If we find 2^m pairs with the same value of the first coordinate, we put this value into B. Therefore, the string x (as well as every element of B) can be specified by $(a - m) + O(\log n)$ bits: $a - m + 1$ bits are needed for the ordinal number of x in the enumeration of B, and $O(\log n)$ is used to specify a and m. So we get

$$C(x) \leqslant (a - m) + O(\log n),$$

and it remains to add this inequality and the preceding one. □

This theorem can be considered as the complexity counterpart of the following combinatorial statement. Let A be a finite set of pairs. Its cardinality is (obviously) bounded by the product of the cardinality of A's projection onto the first coordinate and the maximal cardinality of the sections A_x. This corresponds to the inequality $C(x, y) \leqslant C(x) + C(y\,|\,x) + O(\log n)$. The reverse inequality needs a more subtle interpretation. Let A be a set of pairs, and let p and q be some numbers such that the cardinality of A does not exceed pq. Then we can split A into parts P and Q with the following properties: the projection of P onto the first coordinate has at most p elements, while all the sections Q_x of Q (for element in Q_x the first coordinate equals x) have at most q elements. (Indeed, let P be the union of all sections that have more than q elements. The number of such sections do not exceed p. The remaining elements form Q.) We return to this combinatorial translation in Chapter 10.

Note that in fact we have not used the lengths of x and y, only their complexities. So we have proved the following statement:

Theorem 22 (Kolmogorov–Levin, complexity version).

$$C(x, y) = C(x) + C(y\,|\,x) + O(\log C(x, y))$$

for all strings x and y.

35 Give a more detailed analysis of the additive terms in the proof, and show that

$$C(x) + C(y\,|\,x) \leqslant C(x, y) + 3\log C(x, y) + O(\log\log C(x, y)).$$

36 Show that if $C(x, y\,|\,k, l) < k+l$, then $C(x\,|\,k, l) < k+O(1)$ or $C(y\,|\,x, k, l) < l + O(1)$.

(*Hint*: This is what we actually proved in the proof of Theorem 22.)

37 Show that $O(\log n)$ terms are unavoidable in the Kolmogorov–Levin theorem in both directions: for each n there exist strings x and y of length at most n such that

$$C(x, y) \geqslant C(x) + C(y\,|\,x) + \log n - O(1)$$

as well as strings x and y of length at most n such that

$$C(x, y) \leqslant C(x) + C(y\,|\,x) - \log n + O(1).$$

(*Hint*: For the first inequality we can refer to the remark after Theorem 16 (p. 31). For the second note that $C(x, l(x)) = C(x)$ for every x, while $C(x|l(x))$ can be equal to $l(x) + O(1)$ and $C(x) + O(1)$. Then we can take a random length between $n/2$ and n and a random string of this length.)

38 Prove that changing one bit in a string of length n changes its complexity at most by $\log n + O(\log\log n)$. Prove the same for the conditional complexity $C(x\,|\,n)$.

As we have seen in Problem 7 (p. 21), for every n-bit string x there exists another string x' of the same length that differs from x in one position only such that $C(x') < n - \log n + O(1)$ (and therefore $C(x'\,|\,n) < n - \log n + O(1)$). In particular, if x is incompressible (given n), one can change one bit in x and decrease $C(x\,|\,n)$.

One can also move in the other direction: if $C(x\,|\,n)$ is small enough (this means that $C(x\,|\,n) \leqslant \alpha n$ for some positive constant α), we can increase this complexity by changing one bit in n: there exists some $\alpha > 0$ such that for each n-bit string x with $C(x\,|\,n) \leqslant \alpha n$ one can change one bit in x and get another n-bit string x' such that $C(x'\,|\,n) > C(x\,|\,n)$. (The proof of this statement requires a more involved combinatorial argument [**24**] than the decrease in complexity.)

$\boxed{\mathbf{39}}$ Fix some unconditional decompressor D. Prove that for some constant c and for all integers n and k the following statement is true: if some string x has at least 2^k descriptions of length at most n, then $C(x\,|\,k) \leqslant n - k + c$.

(*Hint*: Fix some k. For each n consider all strings x that have at least 2^k descriptions of length at most n. The number of these strings does not exceed 2^{n-k}, and we can apply Theorem 19, p. 36.)

Using this problem, we can prove the following statement about unconditional complexity (see [**103**, Exercises 4.3.9, 4.3.10]):

$\boxed{\mathbf{40}}$ Let D be some optimal unconditional decompressor. Then there exists some constant c such that for any string x the number of shortest D-descriptions of x does not exceed c.

(*Hint*: The previous problems show that $C(x) \leqslant n - k + 2\log k + O(1)$, so for $C(x) = n$, we get an upper bound for k.)

$\boxed{\mathbf{41}}$ Prove that there exists a constant c with the following property: if for some x and n the probability of the event $C(x\,|\,y) \leqslant k$ (all strings y of lengths n are considered as equiprobable here) is at least 2^{-l}, then $C(x\,|\,n, l) \leqslant k + l + c$.

(*Hint*: Connect each string y of length n to all strings x such that $C(x\,|\,y) \leqslant k$. We get a bipartite graph that has $O(2^{n+k})$ edges. In this graph the number of vertices x that have degree at least 2^{n-l} does not exceed $O(2^{k+l})$. Note that $C(x\,|\,n, l)$ does not include k—this is not a typo!)

This problem could help us in finding the average value of $C(x\,|\,y)$ for given x and all strings y of some length n. It is evident that $C(x\,|\,y) \leqslant C(x\,|\,n) + O(1)$ since $n = l(y)$ is determined by y. It turns out that for most strings y (of given length) this inequality is close to an equality:

$\boxed{\mathbf{42}}$ Prove that there exists some constant c such that for each string x and for all natural numbers n and d the fraction of strings y such that $C(x\,|\,y) < C(x\,|\,n) - d$ (among all strings of length n) does not exceed $cd^2/2^d$. Using this statement, prove that the average value of $C(x\,|\,y)$ taken over all strings y of a given length n equals $C(x\,|\,n) + O(1)$ (the constant in $O(1)$ does not depend on x and n).

$\boxed{\mathbf{43}}$ Prove that $C(x\,|\,k) \leqslant k$ implies $C(x) \leqslant k + O(1)$.

(*Hint*: See Theorem 7. One can also note that if a conditional description of x given k has length k, then k is known anyway, and if this description is shorter, we have enough space to specify the difference between k and the description length.)

A similar (though not identical) statement:

$\boxed{\mathbf{44}}$ Prove that $C(x) = C(x\,|\,C(x)) + O(1)$.

(*Hint*: Assume that x has a conditional description q with condition $C(x)$ that is shorter than $C(x)$. Then one can specify x by providing q and the difference

$C(x) - l(q)$, and we get a description of x that is shorter than $C(x)$—a contradiction.)

$\boxed{\mathbf{45}}$ Prove that for every n there exists an n-bit string x such that

$$C(C(x)\,|\,x) = \log n - O(1).$$

(This is a maximal possible value, since $C(x) \leqslant n$ for n-bit string x.)

This result (in a bit weaker form) was proven long ago by P. Gács [**55**]. Recently E. Kalinina and B. Bauwens suggested a simple game-theoretic proof of this statement. Here is a sketch of their argument (see [**6**] for details). Consider a rectangular game board of width 2^n and height n. Two players, White and Black, make alternating moves and place pawns of their respective colors into the board cells. Unlike chess, each cell may contain both white and black pawns (at most one of each color). At each move a player may place several pawns into different cells (or no pawns at all); after a pawn is placed, it cannot be moved or removed. Also Black can irreversibly mark some cells. The players should obey the following restrictions:

(a) each of the players can place at most 2^i pawns at row i (the bottom row has number 0, the upper row has number $n-1$);

(b) Black can mark at most half of the cells in each column.

A white pawn is declared *killed* if its cell is marked or if there is a black pawn below it (in the same column). The game does not end formally (though it is essentially a finite game); White wins if in the limit there is at least one non-killed white pawn.

White has a winning strategy in this game: place a pawn in the top row and wait until it is killed. If it is killed by the black pawn below, switch to the next column (for example, White can go from left to right starting with the leftmost column). If the pawn was killed by marking its cell, White places another pawn just below the first one, etc. (We may assume that Black makes only the move needed to kill White's pawn; since only the limit position matters in the game, all of Black's other moves can be postponed.) Recall that Black can mark at most half of the column, so Black is forced to put some pawn in the column at some point. It cannot be done in all columns, since the sum of 2^i for all rows is less (by 1) than the width of the table. Note also that White will not violate restrictions on the number of pawns in each row, since in all the columns (except the currently active one) under each white pawn (in row i) there is a black pawn (in some row $j < i$), and the sum of 2^j for all $j < i$ is less that 2^i and there is a space for one more white pawn.

After a winning strategy for White is described, consider the following "universal" strategy for Black: the cell (x, i) is marked as soon as we find that $C(i\,|\,x) < \log n - 1$; a black pawn in placed at (x, i) when a conditional description of x (given n) of length i is found. It is easy to check that Black obeys the game rules. White wins, and a live white pawn at the cell (x, i) means that $C(x\,|\,n) \geqslant i$ and $C(i\,|\,x) \geqslant \log n - 1$. Since the actions of White (playing against the computable strategy of Black) are computable, we conclude that $C(x\,|\,n) \leqslant i + O(1)$: the set of white pawns in row i is enumerable and it contains at most 2^i elements.

This argument shows that $C(C(x\,|\,n)\,|\,x) \geqslant \log n - 1$ (not exactly what we wanted). To get the desired result, we should change the game and consider in parallel boards of all sizes.

$\boxed{\mathbf{46}}$ Prove that for some constant c for any string x and for every number n, there exists a string y of length n such that

$$C(xy) \geqslant C(x\,|\,n) + n - c.$$

(*Hint*: For a given n the number of strings x, such that $C(xy) < k$ for any y of length n, does not exceed $2^k/2^n$, and this property is enumerable. So we can apply Theorem 19 (p. 36).)

$\boxed{\mathbf{47}}$ Let f be a function with natural arguments and values. Assume that

$$f(n) + \varepsilon h \leqslant f(n+h) \leqslant f(n) + (1-\varepsilon)h$$

for some $\varepsilon > 0$ and for all n and h. Prove that there exist an infinite bit sequence ω whose n-bit prefix has complexity $f(n) + O(1)$ for every n.

(*Hint*: Let us add blocks of length h where h is large enough. Each new block being added to an n-bit prefix increases complexity by more than $f(n+h) - f(n)$ or by less than $f(n+h) - f(n)$, depending on the current situation (whether we are below or above the boundary). To find a block with a big complexity increase, we may use the previous problem; for a block with a small increase, we can use a block of zeros. Note that (large enough) h is fixed, so it is enough to control the complexity on the blocks' boundaries.)

$\boxed{\mathbf{48}}$ Prove that an infinite sequence $x_0x_1x_2\cdots$ of zeros and ones is computable if and only if the values $C(x_0\cdots x_{n-1}\,|\,n)$ (the complexities of its prefixes conditional to their lengths) remain bounded by a constant.

(*Hint*: Consider an infinite binary tree. Let S be the enumerable set of vertices (binary strings) that have conditional complexity (w.r.t. their length) less than some constant c. The horizontal sections of S have cardinality $O(1)$. We need to derive from this that each infinite path that lies entirely inside S is computable. We may assume that S is a subtree (only the strings whose prefixes are in S remain in S).

Let ω be an infinite path that goes through S only. At each level n we count vertices in S on the left of ω (l_n vertices) and on the right of ω (r_n vertices). Let $L = \limsup l_n$ and $R = \limsup r_n$. Let N be the level such that L and R are never exceeded after this level. Knowing L, R, and N, we can compute arbitrarily large prefixes of ω. We should look for a path π in a tree such that at some level above N there are at least L elements of S on the left of π and at some (possibly other) level above N there are at least R elements on the right of π. When such a path π is found, we can be sure that its initial segment (up to the first of those two levels) coincides with ω. This result was published in [**108**] (attributed to A.R. Meyer).)

$\boxed{\mathbf{49}}$ Prove that in the previous problem a weaker assumption is sufficient: instead of $C(x_0\cdots x_{n-1}\,|\,n) = O(1)$, we can require that $C(x_0\cdots x_{n-1}) \leqslant \log n + c$ for some c and for all n.

(*Hint*: In this case we get an enumerable set S of strings (=tree vertices) with the following property: the number of vertices below level N is $O(N)$. This means that the average number of vertices per level is bounded by a constant. To use the previous problem, we need a bound for all levels and not for the average value. We can achieve this if we consider only vertices $x \in S$ that have an extension of length $2l(x)$ that goes entirely inside S. This result was published in [**33**].)

Following Problem 48, we can suggest different definitions of the complexity notion for computable bit sequences:

- A minimal complexity of a program that, given n, computes $x_0 \cdots x_{n-1}$. We can also consider a program that computes x_n for input n, which gives the same (up to $O(1)$) complexity. We denote this complexity by $C(x)$.
- A minimal complexity of a program that, given n, computes $x_0 \cdots x_n$ for all *sufficiently large* n. For other n (finitely many of them) this program may provide a wrong answer or never terminate. Complexity defined in this way is denoted by $C_\infty(x)$.
- $\max\{C(x_0 \cdots x_{n-1} \,|\, n) \mid n = 0, 1, \ldots\}$, denoted by $M(x)$.
- $\limsup_{n\to\infty} C(x_0 \cdots x_{n-1} \,|\, n)$, denoted by $M_\infty(x)$.

There are evident relations between the notions

$$M_\infty(x) \leqslant M(x) \leqslant C(x)$$

(up to $O(1)$ additive term) and

$$M_\infty(x) \leqslant C_\infty(x) \leqslant C(x)$$

(with the same precision).

$\boxed{\mathbf{50}}$ Prove that there exists a computable bit sequence x such that $C_\infty(x)$ is much less than $M(x)$ (and, therefore, much less than $C(x)$). More precisely, there exists a sequence x^m of computable sequences such that $C_\infty(x^m) = O(\log m)$ and $M(x^m) \geqslant m$.

(*Hint*: Consider the sequence $x^m = y_m 000\cdots$, where y_m is the lexicographically first string of length m that has conditional complexity (given m) at least m.)

$\boxed{\mathbf{51}}$ Prove that for some computable sequence x the value of $M(x)$ is much less than $C(x)$. More precisely, there exist a sequence x^m of computable sequences such that $M(x^m) = O(\log m)$ and $C(x^m) \geqslant m$.

(*Hint*: Consider the sequence $x^m = (1^{BB(m)} 000\cdots)$, where the number of ones before trailing zeros equals $BB(m)$, defined on p. 24.)

$\boxed{\mathbf{52}}$ Prove that $C_\infty(x) \leqslant 2M_\infty(x) + O(1)$.

(*Hint*: Use the same argument as in Problem 48.)

In fact, the constant 2 in the preceding problem is optimal, as shown in [**52**].

$\boxed{\mathbf{53}}$ Consider strings of length n that have complexity at least n (*incompressible* strings).

(a) Prove that the number of incompressible strings of length n is between 2^{n-c} and $2^n - 2^{n-c}$ (for some c and for all n).

(b) Prove that the cardinality of the set of incompressible strings of length n has complexity $n - O(1)$ (note that this implies the statement (a)).

(c) Prove that if a string x of length $2n$ is incompressible, then its halves x_1 and x_2 (of length n) have complexity $n - O(1)$.

(d) Prove that if a string x of length n is incompressible, then each of its substrings of length k has complexity at least $k - O(\log n)$.

(e) Prove that for any constant $c < 1$ all incompressible strings of sufficiently large length n contain a substring of $\lfloor c \log_2 n \rfloor$ zeros.

(*Hints*: (a) There is at most $2^n - 1$ descriptions of length less than n, and part of them is used for shorter strings: Any string of length $n - d$ (for some d) has complexity less than n. This gives a lower bound for the number of incompressible strings. To prove the upper bound, note that strings of length n that have a prefix of k zeros could be described by $2\log k + (n-k)$ bits.

(b) Let t be the shortest description of the number of incompressible strings. If t has $n-k$ bits, then knowing t and $\log k$ additional bits, we can reconstruct first n and then the list of all incompressible strings of length n, so the first incompressible string has complexity less than n, a contradiction.

(c) If one part of the string is has a short description, the entire string has a short description that starts with prefix-free encoding of the difference between the length and complexity of the compressible part.

(d) If a string has a simple substring, then the entire string can be compressed (we need to specify the substring, its position, and the rest of the string).

(e) Let us count the number of strings of length n that do not contain k zeros in a row; a recurrent relation shows that this number grows like a geometric sequence whose base is the maximal real root of the equation $x = 2 - (1/x^k)$, and we can get a bound for complexity of strings that do not have k zeros in a row.)

$\boxed{\mathbf{54}}$ Prove that (for some constant c) for every infinite sequence $x_0x_1x_2\cdots$ of zeros and ones there exist infinitely many n such that

$$C(x_0x_1\cdots x_{n-1}) \leqslant n - \log n + c.$$

Prove that there is a constant c and the sequence $x_0x_1x_2\cdots$ such that

$$C(x_0x_1\cdots x_{n-1}) \geqslant n - 2\log n - c$$

for all n.

(*Hint*: The series $\sum 1/n$ diverges while the series $\sum(1/n^2)$ converges. For details see Theorem 95 and 99.)

This result was published by Martin-Löf [**117**] for conditional complexity (and a reference to an earlier unpublished work in Russian was given for unconditional complexity; see also [**225**, Theorem 2.6]).

$\boxed{\mathbf{55}}$ For a string x of length n let us define $d(x)$ and $d_c(x)$ as follows:

$$d(x) = n - C(x) \qquad \text{and} \qquad d_c(x) = n - C(x\,|\,n).$$

Show that they are rather close to each other:

$$d_c(x) - 2\log d_c(x) - O(1) \leqslant d(x) \leqslant d_c(x) + O(1).$$

(*Hint*: We need to show that $C(x\,|\,n) = n - d$ implies $C(x) \leqslant n - d + 2\log d + O(1)$. Indeed, let us take the conditional description of x of length $n-d$ and put it after the self-delimiting description of d that has size $2\log d + O(1)$. Knowing this string, we can reconstruct d, then n, and finally x.])

$\boxed{\mathbf{56}}$ Prove that $d(xy) = d(x) + d(y\,|\,x) + O(\log d(xy))$ for every two n-bit strings x and y. (Here $d(u) = l(u) - C(u)$.)

(*Hint*: Use Problem 36.)

The intuitive meaning of the difference between length and complexity as a kind of "randomness deficiency" is discussed (for different complexity versions) in Chapter 5 and Chapter 14.

$\boxed{\mathbf{57}}$ Prove that for sufficiently large values of a constant c the enumerable set of pairs (x, y) such that $C(x\,|\,y) < c$ is Turing complete (one can solve the halting problem using an oracle for such a set).

(*Hint*: Use Problem 15 and the fact that the output of a program has $O(1)$ conditional complexity given the program.)

2.3. Complexity as the amount of information

As we know (Theorem 18), the conditional complexity $C(y|x)$ does not exceed the unconditional one $C(y)$ (up to a constant). The difference $C(y) - C(y|x)$ tells us how the knowledge of y makes x easier to describe. So this difference can be called the *amount of information in x about y*. We use the notation $I(x\!:\!y)$.

Theorem 18 says that $I(x\!:\!y)$ is non-negative (up to a constant): there exists some c such that $I(x\!:\!y) \geqslant c$ for all x and y.

$\boxed{\mathbf{58}}$ Let f be a computable function. Prove that $I(f(x)\!:\!y) \leqslant I(x\!:\!y) + c$ for some c and for all x, y such that $f(x)$ is defined.

A generalization of this statement to probabilistic algorithms is possible.

$\boxed{\mathbf{59}}$ Let $f(x, r)$ be a computable function of two arguments, and let r be chosen at random uniformly among n-bit strings for some n. Then for each l the probability of the event

$$I(f(x,r)\!:\!y) > I(x\!:\!y) + l$$

does not exceed $2^{-l+O(C(n)+C(l))}$.

(*Hint*: Use the conditional version of Problem 41.)

These properties of information can be described as *conservation laws* for information (about something) in algorithmic or random processes. As Levin once put it, "by torturing an uninformed person you do not get any evidence about the crime." He discusses this property (for different notions of information) in [**100**].

Recall that

$$C(x, y) = C(x) + C(y\,|\,x) + O(\log C(x, y))$$

(Theorem 22, p. 39). This allows us to express conditional complexity in terms of an unconditional one: $C(y\,|\,x) = C(x, y) - C(x) + O(\log C(x, y))$. Then we get the following expression for the information:

$$I(x\!:\!y) = C(y) - C(y\,|\,x) = C(x) + C(y) - C(x, y) + O(\log C(x, y)).$$

This expression immediately implies the following theorem:

THEOREM 23 (Information symmetry).

$$I(x\!:\!y) = I(y\!:\!x) + O(\log C(x, y)).$$

So the difference between $I(x\!:\!y)$ and $I(y\!:\!x)$ is logarithmically small compared to $C(x, y)$. The following problem shows that at the same time this difference could be comparable with the values $I(x\!:\!y)$ and $I(y\!:\!x)$ if they are much less than $C(x, y)$.

$\boxed{\mathbf{60}}$ Let x be a string of length n such that $C(x\,|\,n) \geqslant n$. Show that

$$I(x\!:\!n) = C(n) + O(1) \text{ and } I(n\!:\!x) = O(1).$$

The property of information symmetry (up to a logarithmic term) explains why $I(x\!:\!y)$ (or $I(y\!:\!x)$) is sometimes called *mutual information* in two strings x and y. The connection between mutual information, conditional and unconditional complexities, and pair complexity can be illustrated by a (rather symbolic) picture (see Figure 3).

It shows that strings x and y have $I(x\!:\!y) \approx I(y\!:\!x)$ bits of mutual information. Adding $C(x\,|\,y)$ bits (information that is present in x but not in y, the left part), we obtain

$$I(y\!:\!x) + C(x\,|\,y) \approx (C(x) - C(x\,|\,y)) + C(x\,|\,y) = C(x)$$

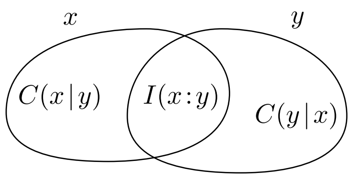

FIGURE 3. Mutual information and conditional complexity

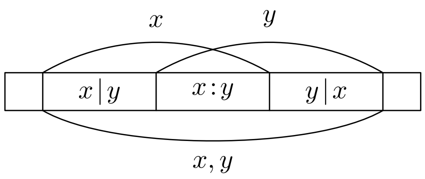

FIGURE 4. Common information in overlapping substrings

bits (matching the complexity of x). Similarly, the central part together with $C(y\,|\,x)$ (the right part) give $C(y)$. Finally, all three parts together give us

$$C(x\,|\,y) + I(x\!:\!y) + C(y\,|\,x) = C(x) + C(y\,|\,x) = C(x\,|\,y) + C(y) = C(x,y)$$

bits (all equalities are true up to $O(\log n)$ for strings x and y of length at most n).

In some cases this picture can be understood quite literally. Consider, for instance, an incompressible string $r = r_1 \cdots r_n$ of length n, so $C(r_1 \cdots r_n) \geqslant n$. Then any substring u of x has complexity $l(u)$ up to $O(\log n)$ terms. Indeed, since u is a substring of r, we have $r = tuv$ for some strings t, v. Then

$$l(r) = C(r) \leqslant C(t) + C(u) + C(v) \leqslant l(t) + l(u) + l(v) = l(r)$$

(up to a logarithmic error), and therefore all the inequalities are equalities (with the same logarithmic precision).

Now take two overlapping substrings x and y (Figure 4). Then $C(x)$ is the length of x and $C(y)$ is the length of y (up to $O(\log n)$).

The complexity $C(x,y)$ is equal to the length of the union of segments (since the pair $\langle x, y\rangle$ is equivalent to this union plus information about lengths requiring $O(\log n)$ bits).

Therefore, conditional complexities $C(x\,|\,y)$, $C(y\,|\,x)$ and the mutual information $I(x\!:\!y)$ are equal to the lengths of the corresponding segments (up to $O(\log n)$).

However, the mutual information cannot always be extracted in the form of some string (like it happened in our example, where this common information is just the intersection of strings x and y). As we will see in Chapter 11, there exist two strings x and y that have large mutual information $I(x\!:\!y)$ but there is no string z that represents (*materializes*) this information in the following sense: $C(z\,|\,x) \approx 0$, $C(z\,|\,y) \approx 0$ (all information that is present in z is also present both in x and in y), and $C(z) \approx I(x\!:\!y)$ (all mutual information is extracted). In our last example we can take the intersection substring for z, but in general this is not possible.

$\boxed{\mathbf{61}}$ Prove that for any string x of length at most n the expected value of the mutual information $I(x\!:\!y)$ in x and the random string y of length n is $O(\log n)$.

Now we move to triples of strings instead of pairs. Here we have an important tool that can be called *relativization*: most of the results proved for unconditional complexities remain valid for conditional complexities (and proofs remain valid with minimal changes). Let us give some example of this type.

A theorem about the complexity of pairs (p. 31) says that

$$C(x,y) \leqslant C(x) + 2\log C(x) + C(y) + O(1).$$

Replacing all complexities by conditional ones (with the same condition z in all cases), we get the following inequality:

$$C(x,y\,|\,z) \leqslant C(x\,|\,z) + 2\log C(x\,|\,z) + C(y\,|\,z) + O(1).$$

By conditional complexity of a pair x, y relative to z we mean, as one can expect, the conditional complexity of its encoding: $C(x,y\,|\,z) = C([x,y]\,|\,z)$. As for unconditional complexity, the choice of encoding is not important (the complexity changes by $O(1)$).

The proof of this relativized inequality repeats the proof of the unrelativized one: we combine the description p for x (with condition z) and the description q for y (with condition z) into a string $\hat{p}q$ that is a description of $[p,q]$ (with condition z) relative to some suitable conditional decompressor.

This is nothing really new. However, we may express all the conditional complexities in terms of unconditional ones: recall that $C(x,y\,|\,z) = C(x,y,z) - C(z)$ and $C(x\,|\,z) = C(x,z) - C(z)$, $C(y\,|\,z) = C(y,z) - C(z)$ (with logarithmic precision). Then we get the following theorem:

Theorem 24.

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

for all strings x, y, z of complexity at most n.

Sometimes this inequality is called the *basic* inequality for complexities.

The same relativization can be applied to Theorem 21 (p. 37) that relates the complexity of a pair and conditional complexity. Then we get the following statement:

Theorem 25.

$$C(x,y\,|\,z) = C(x\,|\,z) + C(y\,|\,x,z) + O(\log n)$$

for all strings x, y, z of complexity at most n.

Proof. We can follow the proof of Theorem 21, replacing unconditional descriptions by conditional ones (with z as the condition). Doing this, we replace $C(y\,|\,x)$ by $C(y\,|\,x,z)$. One can say that now we work in three-dimensional space with coordinates x, y, z and apply the same arguments simultaneously in all planes parallel to the xy plane.

If this argument does not look convincing, there is a more formal one. Express all the conditional complexities in terms of unconditional ones:

$$C(x,y\,|\,z) = C(x,y,z) - C(z),$$

and for the right-hand side

$$C(x\,|\,z) + C(y\,|\,x,z) = C(x,z) - C(z) + C(y,x,z) - C(x,z).$$

We see that both sides coincide (up to $O(\log n)$). (A careful reader may note that this simplified argument gives larger hidden constants in $O(\log n)$-notation.) □

$\boxed{\mathbf{62}}$ Prove that in Theorem 25 the weaker assumption "$C(x\,|\,z)$ and $C(y\,|\,x,z)$ do not exceed n" is sufficient.

We also relativize the definition of mutual information and let $I(x\!:\!y\,|\,z)$ be the difference $C(y\,|\,z) - C(y\,|\,x,z)$. As for the case of (unconditional) information, this quantity is non-negative (with $O(1)$ precision). Replacing conditional complexities by the expressions involving unconditional ones (with logarithmic precision), we can rewrite the inequality $I(x\!:\!y\,|\,z) \geqslant 0$ as

$$C(y\,|\,z) - C(y\,|\,x,z) = C(y,z) - C(z) - C(y,x,z) + C(x,z) \geqslant 0.$$

So we get the basic inequality of Theorem 24 again.

In fact, almost all known equalities and inequalities that involve complexities (unconditional and conditional) and information (and have logarithmic precision) are immediate consequences of Theorems 21 and 24. The first examples of linear inequalities for complexities that do *not* follow from basic inequalities were found fairly recently (see [**222, 223**]) and they are rather complicated and not very intuitive. We discuss them in Section 10.13; we conclude our discussion here with two simple corollaries of basic inequalities.

Independent strings. We say that strings x and y are "independent" if $I(x\!:\!y) \approx 0$. We need to specify what we mean by "$\approx$", but we always ignore the terms of order $O(\log n)$ where n is the maximal length (or complexity) of the strings involved.

Independent strings could be considered as some counterpart of the notion of independent random variables, which is central in probability theory. There is a simple observation: if a random variable ξ is independent with the pair of random variables $\langle \alpha, \beta \rangle$, then ξ is independent with α and with β (separately).

The Kolmogorov complexity counterpart of this statement (if a string x is independent with a pair $\langle y, z\rangle$, then x is independent with y and x is independent with z) can be expressed as the inequality

$$I(x\!:\!\langle y,z\rangle) \geqslant I(x\!:\!y)$$

(and the similar inequality for z instead of y). This inequality is indeed true (with logarithmic precision), and it is easy to see if we rewrite it in terms of unconditional complexities,

$$C(x) + C(y,z) - C(x,y,z) \geqslant C(x) + C(y) - C(x,y),$$

which after cancellation of similar terms gives a basic inequality (Theorem 24). (In classical probability theory one may also apply a similar inequality for Shannon entropies.)

Complexity of pairs and triples. On the other hand, to prove the following theorem (which we have already mentioned on p. 12), it is convenient to replace unconditional complexities by conditional ones:

THEOREM 26.

$$2C(x,y,z) \leqslant C(x,y) + C(x,z) + C(y,z) + O(\log n)$$

for all strings x, y, z of complexity at most n.

PROOF. Moving $C(x,y)$ and $C(x,z)$ to the left-hand side and replacing the differences $C(x,y,z) - C(x,y)$ and $C(x,y,z) - C(x,z)$ by conditional complexities

$C(z\,|\,x,y)$ and $C(y\,|\,x,z)$, we get the inequality

$$C(z\,|\,x,y) + C(y\,|\,x,z) \leqslant C(y,z) + O(\log n).$$

It remains to rewrite the right-hand side of this inequality as $C(y) + C(z\,|\,y)$, and note that $C(z\,|\,x,y) \leqslant C(z\,|\,y)$ and $C(y\,|\,x,z) \leqslant C(y)$. □

Instead we could just add two inequalities (the basic one and the inequality for the complexity of a pair):

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

and then cancel $C(y)$ in both sides. (This proof, as well as the previous one, has an important aesthetic problem: both treat x, y, z in a non-symmetric way while the statement of the theorem is symmetric.)

We return to the inequality of Theorem 26 and refer to its geometric consequences in Chapter 10.

We can provide a more systematic treatment of the different complexity quantities related to three strings as follows. There are seven basic quantities: three of them are complexities of individual strings, another three are complexities of pairs, and one more is the complexity of the entire triple. Other quantities such as conditional complexity and mutual information can be expressed in terms of these seven complexities. To understand better what requirements these seven quantities should satisfy, let us make a linear transformation in the seven-dimensional space and switch to new coordinates. Consider seven variables $a_1, a_2, \ldots, a_7$ that correspond to the seven regions shown in Figure 5.

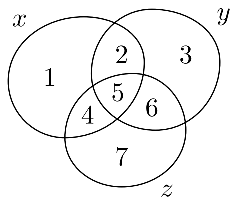

FIGURE 5. New coordinates $a_1, a_2, \ldots, a_7$

Formally, the coordinate transformation is given by the following equations:

$$\begin{aligned}
C(x) &= a_1 + a_2 + a_4 + a_5,\\
C(y) &= a_2 + a_3 + a_5 + a_6,\\
C(z) &= a_4 + a_5 + a_6 + a_7,\\
C(x,y) &= a_1 + a_2 + a_3 + a_4 + a_5 + a_6,\\
C(x,z) &= a_1 + a_2 + a_4 + a_5 + a_6 + a_7,\\
C(y,z) &= a_2 + a_3 + a_4 + a_5 + a_6 + a_7,\\
C(x,y,z) &= a_1 + a_2 + a_3 + a_4 + a_5 + a_6 + a_7.
\end{aligned}$$

Indeed, it is easy to see that these equations determine an invertible linear transformation of $\mathbb{R}^7$: each 7-tuple of complexities corresponds to unique value of variables $a_1, \ldots, a_7$.

Conditional complexities and expressions for mutual information are combinations of complexities and therefore could be rewritten in new coordinates. For example,

$$I(x:y) = C(x) + C(y) - C(x,y) = a_2 + a_5 \text{ and } C(x\,|\,y) = C(x,y) - C(y) = a_1 + a_4.$$

What is the intuitive meaning of these new coordinates? It is easy to see that $a_1 = C(x\,|\,y,z)$ (with logarithmic precision). The meaning of a_3 (and a_7) is similar. The coordinate a_2 (with the same precision) is $I(x:y\,|\,z)$; coordinates a_4 and a_6 have similar meaning (see Figure 6). In particular, we conclude that for any strings x, y, z the corresponding values of coordinates $a_1, a_2, a_3, a_4, a_6, a_7$ are non-negative (up to $O(\log n)$ for strings x, y, z of complexity at most n).

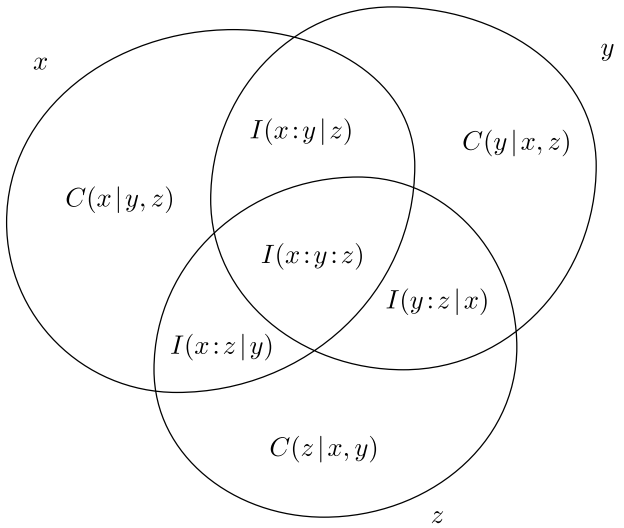

FIGURE 6. The complexity interpretation of new coordinates

The coordinate a_5 is more delicate. Informally, we would like to understand it as the "amount of mutual information in three strings x, y, z". Sometimes the notation $I(x:y:z)$ is used. However, the meaning of this expression is not quite clear, especially if we take into account that a_5 can be negative.

Consider the following example where $a_5 < 0$. Let x and y be two halves of an incompressible string of length $2n$. Then $C(x) = n$, $C(y) = n$, $C(x,y) = 2n$, and $I(x:y) = 0$ (up to $O(\log n)$). Consider a string z of length n which is a bitwise sum modulo 2 of x and y (XOR-operation). Then each of the strings x, y, z can be reconstructed if two others are known; therefore, the complexities of all pairs $C(x,y), C(y,z)$, $C(x,z)$ are equal to $2n$ (again up to $O(\log n)$), and the complexity $C(x,y,z)$ is also $2n$. The complexity of z is equal to n (it cannot be larger, since the length is n; on the other hand, it cannot be smaller, since z and y form a pair of complexity $2n$).

The values of $a_1, \ldots, a_7$ for this example are shown in Figure 7.

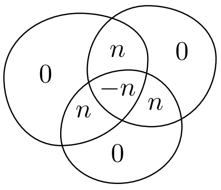

FIGURE 7. Two independent incompressible strings of length n and their XOR

Note that even if a_5 is negative, the sums $a_5 + a_2$, $a_5 + a_4$ and $a_5 + a_6$, being mutual information expressions for pairs, are non-negative. (In our examples these sums are equal to 0.)

This example corresponds to the simple case of secret sharing of secret z between two people: if one of them knows x and the other one knows y, then neither of them has any information about z in isolation (since $I(x:z) \approx 0$ and $I(y:z) \approx 0$)), but together they can reconstruct z as a bitwise sum of x and y.

One can check that we have already given a full list of inequalities that are true for complexities of three strings and their combinations (all a_i, except for a_5, are non-negative, as well as the three sums mentioned above). We return to this question in Chapter 10.

Our diagram is a good mnemonic tool. For example, consider again the inequality

$$C(x, y, z) \leqslant C(x, y) + C(x, z) + C(y, z).$$

In our new variables it can be rewritten as $a_2 + a_4 + a_5 + a_6 \geqslant 0$ (you can easily check it by counting the multiplicity of each a_i in both sides of the inequality). It remains to note that $a_2 + a_5 \geqslant 0$, $a_4 \geqslant 0$, and $a_6 \geqslant 0$. (Alas, the symmetry is broken again!)

$\boxed{\mathbf{63}}$ Prove that $I(xy:z) = I(x:z) + I(y:z\,|\,x) + O(\log n)$ for strings x, y, z of complexity at most n.

(*Hint*: Use the diagram.)

This problem shows that information in xy about z can somehow be split into two parts: information in x about z and information in y about z (when x is known). This is somehow similar to the equality $C(x, y) = C(x) + C(y\,|\,x)$, but now complexity is replaced by the quantity of information about z. As a corollary we immediately get that if xy is independent with z, then x is independent with z and, at the same time, y is independent with z when x is known. (Here independence means that mutual information is negligible.) A symmetric argument shows that y is independent with z and x is independent with z when y is known.

$\boxed{\mathbf{64}}$ Show that properties "x is independent with y" and "x is independent with y when z is known" are quite different: each of them can be true when the other is false.

$\boxed{\mathbf{65}}$ We say that strings x, y, z, t form a *Markov chain* (a well-known notion in probability theory now transferred to algorithmic information theory) if $I(x:z\,|\,y)$ and $I(\langle x, y\rangle:t\,|\,z)$ are negligible. (Of course, we need to specify what is "negligible" to get a formal definition.) Show that the reversed sequence of strings also forms a Markov chain, i.e., that $I(t:y\,|\,z)$ and $I(\langle t, z\rangle:x\,|\,y)$ are negligible.

(*Hint*: Since $I(\langle x, y\rangle : t \,|\, z) = I(y : t \,|\, z) + I(x : t \,|\, y, z)$, the left-hand side in this equality is zero if and only if both terms in the right-hand side are zero; and the second term in the right-hand side does not change when the order of x, y, z, t is reversed.)