Algorithmic statistics: forty years later

N. Vereshchagin, A. Shen

Abstract

Algorithmic statistics has two different (and almost orthogonal) motivations. From the philosophical point of view, it tries to formalize how the statistics works and why some statistical models are better than others. After this notion of a "good model" is introduced, a natural question arises: it is possible that for some piece of data there is no good model? If yes, how often these bad (*non-stochastic*) data appear "in real life"?

Another, more technical motivation comes from algorithmic information theory. In this theory a notion of complexity of a finite object (=amount of information in this object) is introduced; it assigns to every object some number, called its *algorithmic complexity* (or *Kolmogorov complexity*). Algorithmic statistic provides a more fine-grained classification: for each finite object some curve is defined that characterizes its behavior. It turns out that several different definitions give (approximately) the same curve.

In this survey we try to provide an exposition of the main results in the field (including full proofs for the most important ones), as well as some historical comments. We assume that the reader is familiar with the main notions of algorithmic information (Kolmogorov complexity) theory. An exposition can be found in [41, chapters 1, 3, 4] or [21, chapters 2, 3], see also the survey [34].

A short survey of main result of algorithmic statistics was given in [40] (without proofs); see also the last chapter of the book [41].

Contents

1 Statistical models

Let us start with a (very rough) scheme. Imagine an experiment that produces some bit string x. We know nothing about the device that produced this data, and cannot repeat the experiment. Still we want to suggest some statistical model that fits the data

("explains" x in a plausible way). This model is a probability distribution on some finite set of binary strings containing x. What do we expect from a reasonable model?

There are, informally speaking, two main properties of a good model. First, the model should be "simple". If a model contains so many parameters that it is more complicated than the data itself, we would not consider it seriously. To make this requirement more formal, one can use the notion of Kolmogorov complexity.[1] Let us assume that measure P (used as a model) has finite support and rational values. Then P can be considered as a finite (constructive) object, so we can speak about Kolmogorov complexity of P. The requirement then says that complexity of P should be much smaller than the complexity of the data string x itself.

For example, if a data string x contains n bits, we may consider a model that corresponds to n independent fair coin tosses, i.e., the uniform distribution P on the set of all n-bit strings. Such a distribution is a constructive object that is completely determined by the value of n, so its complexity is $O(\log n)$, while the complexity of most n-bit strings is close to n (and therefore is much larger than the complexity of P, if n is large enough).

Still this simple model looks unacceptable if, for example, the sequence x consists of n zeros, or, more generally, if the frequency of ones in x deviates significantly from 1/2, or if zeros and ones alternate. This feeling was one of the motivations for the development of algorithmic randomness notions: why some bit sequences of length n look plausible as outcomes of n fair coin tosses while other do not, while all the n-bit sequences have the same probability 2^{-n} according to the model? This question does not have a clear answer in the classical probability theory, but the algorithmic approach to randomness says that plausible strings should be incompressible: the complexity of such a string (the minimal length of a program producing it) should be close to its length.

This answer works for a uniform distribution on n-bit strings; for arbitrary P it should be modified. It turns out that for arbitrary P we should compare the complexity of x not with its length but with the value $(-\log P(x))$ (all logarithms are binary); if P is the uniform distribution on n-bit strings, the value of $(-\log P(x))$ is n for all n-bit strings x. Namely, we consider the difference between $(-\log P(x))$ and complexity of x as *randomness deficiency* of x with respect to P. We discuss the exact definition in the next section, but let us note here that this approach looks natural: different data strings require different models.

Disclaimer. The scheme above is oversimplified in many aspects. First, it rarely happens that we have no a priori information about the experiment that produced the data. Second, in many cases the experiment can be repeated (the same experimental device can be used again, or a similar device can be constructed). Also we often deal with a data

[1] We assume that the reader is familiar with basic notions of algorithmic information theory (complexity, a priory probability). See [34] for a concise introduction, and [21, 41] for more details.

stream: we are more interested, say, in a good prediction of oil prices for the next month than in a construction of model that fits well the prices in the past. All these aspects are ignored in our simplistic model; still it may serve as an example for more complicated cases. One should stress also that algorithmic statistics is more theoretical than practical: one of the reasons is that complexity is a non-computable function and is defined only asymptotically, up to a bounded additive term. Still the notions and results from this theory can be useful not only as philosophical foundations of statistics but as a guidelines when comparing statistical models in practice.

More practical approach to the same question is provided by machine learning that deals with the same problem (finding a good model for some data set) in the "real world". Unfortunately, currently there is a big gap between the algorithmic statistics and machine learning: the first one provides nice results about mathematical models that are quite far from practice (see the discussion about "standard models" below), while machine learning is a tool that sometimes works well without any theoretical reasons. There are some attempts to close this gap (by considering models from some class or resource-bounded versions of the notions), but much more remains to be done.

A historical remark. The principles of algorithmic statistics are often traced back to Occam's razor principle often stated as "Don't multiply postulations beyond necessity" or in a similar way. Poincare writes in his *Science and Method'* (Chapter 1, *The choice of facts*) that "this economy of thought, this economy of effort, which is, according to Mach, the constant tendency of science, is at the same time a source of beauty and a practical advantage". Still the mathematical analysis of these ideas became possible only after a definition of algorithmic complexity was given in 1960s (by Solomonoff, Kolmogorov and then Chaitin): after that the connection between randomness and incompressibility (high complexity) became clear. The formal definition of (α, β)-stochasticity (see the next section) was given by Kolmogorov (the authors learned it from his talk given in 1981 [16], but most probably it was formulated earlier in 1970s; the definition appeared in print in [32]). For the other related approaches (the notions of logical depth and sophistication, minimal description length principle) see the discussion in the corresponding sections.

2 (α, β)-stochasticity

2.1 Prefix complexity, a priori probability and randomness deficiency

Preparing for the precise definition of (α, β)-stochasticity, we need to fix the version of complexity used in this definition. There are several versions (plain and prefix complexities, different types of conditions), see [41, Chapter 6]. For most of the results the choice between these versions is not important, since the difference between the different ver-

sions is small (at most $O(\log n)$ for strings of length n), and we usually allow errors of logarithmic size in the statements.

We will use the notion of *conditional prefix complexity*, usually denoted by $\mathrm{K}(x|c)$. Here x and c are finite objects; we measure the complexity of x when c is given. This complexity is defined as the length of the minimal prefix-free program that, given c, computes x.[2] The advantage of this definition is that it has an equivalent formulation in terms of a priori probability [41, Chapter 4]: if $\mathbf{m}(x|c)$ is the conditional a priori probability, i.e., the maximal lower semicomputable function of two arguments x and c such that $\sum_x \mathbf{m}(x|c) \le 1$ for every c, then

$$\mathrm{K}(x|c) = -\log \mathbf{m}(x|c) + O(1).$$

In particular, if a probability distribution P with finite support and rational values (we consider only distributions of this type) is considered as a condition, we may compare $\mathbf{m}$ with function $(x, P) \mapsto P(x)$ and conclude that $\mathbf{m}(x|P) \ge P(x)$ up to an $O(1)$-factor, so $\mathrm{K}(x|P) \le -\log P(x)$. So if we define the randomness deficiency as

$$d(x|P) = -\log P(x) - \mathrm{K}(x|P),$$

we get a non-negative (up to $O(1)$ additive term) function. One may also explain in a different way why $\mathrm{K}(x|P) \le -\log P(x)$: this inequality is a reformulation of a standard result from information theory (Shannon–Fano code, Kraft inequality).

Why do we define the deficiency in this way? The following proposition provides some additional motivation.

Proposition 1. The function $d(x|P)$ is (up to $O(1)$-additive term) the maximal lower semicomputable function of two arguments x and P such that

$$\sum_x 2^{d(x|P)} \cdot P(x) \le 1 \qquad (*)$$

for every P.

Here x is a binary string, and P is a probability distribution on binary strings with finite support and rational values. By lower semicomputable functions we mean functions that can be approximated from below by some algorithm (given x and P, the algorithm produces an increasing sequence of rational numbers that converges to $d(x|P)$; no bounds

[2] We do not go into details here, but let us mention one common misunderstanding: the set of programs should be prefix-free for each c, but these sets may differ for different c and the union is not required to be prefix-free.

for the convergence speed are required). Then, for a given P, the function $x \mapsto 2^{d(x|P)}$ can be considered as a random variable on the probability space with distribution P. The requirement $(*)$ says that its expectation is at most 1. In this way we guarantee (by Markov inequality) that only a P-small fraction of strings have large deficiency: the P-probability of the event $d(x|P) > c$ is at most 2^{-c}. It turns out that there exists a maximal function d satisfying $(*)$ up to $O(1)$ additive term, and our formula gives the expression for this function in terms of prefix complexity.

Proof. The proof uses standard arguments from Kolmogorov complexity theory. The function $\mathrm{K}(x|P)$ is upper semicomputable, so $d(x|P)$ is lower semicomputable. We can also note that

$$\sum_x 2^{d(x|P)} \cdot P(x) = \sum_x \frac{\mathbf{m}(x|P)}{P(x)} \cdot P(x) = \sum_x \mathbf{m}(x|P) \le 1,$$

so the deficiency function satisfies $(*)$.

To prove the maximality, consider an arbitrary function $d'(x|P)$ that is lower semicomputable and satisfies $(*)$. Then consider a function $m(x|P) = 2^{d'(x|P)} \cdot P(x)$ (the function equals 0 if x is not in the support of P). Then m is lower semicomputable, $\sum_x m(x|P) \le 1$ for every P, so $m(x|P) \le \mathbf{m}(x|P)$ up to $O(1)$-factor; this implies that $d'(x|P) \le d(x|P) + O(1)$. □

For the case where P is the uniform distribution on n-bit strings, using P as a condition is equivalent to using n as the condition, so

$$d(x|P) = n - \mathrm{K}(x|n)$$

in this case, and small deficiency means that complexity $\mathrm{K}(x|n)$ is close to the length n, so x is incompressible.[3]

2.2 Definition of stochasticity

Definition 1. A string x is called (α, β)-stochastic if there exists some probability distribution P (with rational values and finite support) such that $\mathrm{K}(P) \le \alpha$ and $d(x|P) \le \beta$.

[3]Initially Kolmogorov suggested to consider $n - \mathrm{C}(x)$ as "randomness deficiency" in this case, where C stands for the plain (not prefix) complexity. One may also consider $n - \mathrm{C}(x|n)$. But all three deficiency functions mentioned are close to each other for strings x of length n; one can show that the difference between them is bounded by $O(\log d)$ where d is any of these three functions. The proof works by comparing the expectation and probability-bounded characterizations as explained in [9].

By definition every (α, β)-stochastic string is (α', β')-stochastic for $\alpha' \geq \alpha$, $\beta' \geq \beta$. Sometimes we say informally that a string is "stochastic" meaning that it is (α, β)-stochastic for some reasonably small thresholds α and β (for example, one can consider $\alpha, \beta = O(\log n)$ for n-bit strings).

Let us start with some simple remarks.

- Every simple string is stochastic. Indeed, if P is concentrated on x (singleton support), then $\mathrm{K}(P) \leq \mathrm{K}(x)$ and $d(x|P) = 0$ (in both cases with $O(1)$-precision), so x is always $(\mathrm{K}(x) + O(1), O(1))$-stochastic.

- On the other end of the spectrum: if P is a uniform distribution on n-bit strings, then $\mathrm{K}(P) = O(\log n)$, and most strings of length n have $d(x|P) = O(1)$, so most strings of length n are $(O(\log n), O(1))$-stochastic. The same distribution also witnesses that every n-bit string is $(O(\log n), n + O(1))$-stochastic.

- It is easy to construct stochastic strings that are between these two extreme cases. Let x be an incompressible string of length n. Consider the string $x0^n$ (the first half is x, the second half is zero string). It is $(O(\log n), O(1))$-stochastic: let P be the uniform distribution on all the strings of length $2n$ whose second half contains only zeros.

- For every distribution P (with finite support and rational values, as usual) a random sampling according to P gives us a $(\mathrm{K}(P), c)$-stochastic string with probability at least $1 - 2^{-c}$. Indeed, the probability to get a string with deficiency greater than c is at most 2^{-c} (Markov inequality, see above).

After these observations one may ask whether non-stochastic strings exists at all — and how they can be constructed? A non-stochastic string should have non-negligible complexity (our first observation), but a standard way to get strings of high complexity, by coin tossing or other random experiment, can give only stochastic strings (our last observation).

We will see that non-stochastic strings do exist in the mathematical sense; however, the question whether they appear in the "real world", is philosophical. We will discuss both questions soon, but let us start with some mathematical results.

First of all let us note that with logarithmic precision we may restrict ourselves to uniform distributions on finite sets.

Proposition 2. Let x be an (α, β)-stochastic string of length n. Then there exist a finite set A containing x such that $\mathrm{K}(A) \leq \alpha + O(\log n)$ and $d(x|U_A) \leq \beta + O(\log n)$, where U_A is the uniform distribution on A.

Since $\mathrm{K}(A) = \mathrm{K}(U_A)$ (with $O(1)$-precision, as usual), this proposition means that we may consider only uniform distributions in the definition of stochasticity, and get an equivalent (up to logarithmic change in the parameters) definition. According to this modified definition, a string x in (α, β)-stochastic if there exists a finite set A such that $\mathrm{K}(A) \le \alpha$ and $d(x|A) \le \beta$, where $d(x|A)$ is now defined as $\log \#A - \mathrm{K}(x|A)$. Kolmogorov originally proposed the definition in this form (but used plain complexity).

Proof. Let P be the (finite) distribution that exists due to the definition of (α, β)-stochasticity of x. We may assume without loss of generality that $\beta \le n$ (as we have seen, all strings of length n are $(O(\log n), n + O(1))$-stochastic, so for $\beta > n$ the statement is trivial). Consider the set A formed by all strings that have sufficiently large P-probability. Namely, let us choose minimal k such that $2^{-k} \le P(x)$ and consider the set A of all strings such that $P(x) \ge 2^{-k}$. By construction A contains x. The size of A is at most 2^k, and $-\log P(x) = \log \#A = k$ with $O(1)$-precision. According to our assumption, $d(x|P) = k - \mathrm{K}(x|P) \le n$, so $k = d(x|P) + \mathrm{K}(x|P) \le O(n)$. Then

$$\mathrm{K}(x|A) \ge \mathrm{K}(x|P, k) \ge \mathrm{K}(x|P) - O(\log n),$$

since A is determined by P, k, and the additional information in k is $O(\log k) = O(\log n)$ since $k = O(n)$ by our assumption. So the deficiency may increase only by $O(\log n)$ when we replace P by U_A, and for the same reasons

$$\mathrm{K}(A) \le \mathrm{K}(P, k) \le \mathrm{K}(P) + O(\log n).$$

□

Remark 1. Similar argument can be applied if P is a computable distribution (may be, with infinite support) computed by some program p, and we require $\mathrm{K}(p) \le \alpha$ and $-\log P(x) - \mathrm{K}(x|p) \le \beta$. So in this way we also get the same notion (with logarithmic precision). It is important, however, that program p *computes* the distribution P (given some point x and some precision $\varepsilon > 0$, it computes the probability of x with error at most ε). It is *not* enough for P to be an output distribution for a randomized algorithm p (in this case P is called the semimeasure lower *semicomputed* by p; note that the sum of probabilities may be strictly less than 1 since the computation may diverge with positive probability). Similarly, it is very important in the version with finite sets A (and uniform distributions on them) that the set A is considered as a finite object: A is simple if there is a short program that prints the list of all elements of A. If we allowed the set A to be presented by an algorithm that enumerates A (but never says explicitly that no more elements will appear), then situation would change drastically: for every string of complexity k the finite set S_k of strings that have complexity at most k, would be a good explanation for x, so all objects would become stochastic.

2.3 Stochasticity conservation

We have defined stochasticity for binary strings. However, the same definition can be used for arbitrary finite (constructive) objects: pairs of strings, tuples of strings, finite sets of strings, graphs, etc. Indeed, complexity can be defined for all these objects as the complexity of their encodings; note that the difference in complexities for different encodings is at most $O(1)$. The same can be done for finite sets of these objects (or probability distributions), so the definition of (α, β)-stochasticity makes sense.

One can also note that computable bijection preserves stochasticity (up to a constant that depends on the bijection, but not on the object). In fact, a stronger statement is true: every total computable mapping preserves stochasticity. For example, consider a stochastic pair of strings (x, y). Does it imply that x (or y) is stochastic? It is indeed the case: if P is a distribution on pairs that is a reasonable model for (x, y), then its projection (marginal distribution on the first components) should be a reasonable model for x. In fact, projection can be replaced by any *total* computable mapping.

Proposition 3. Let F be a total computable mapping whose arguments and values are strings. If x is (α, β)-stochastic, then $F(x)$ is $(\alpha + O(1), \beta + O(1))$-stochastic. Here the constant in $O(1)$ depends on F but not on x, α, β.

Proof. Let P be the distribution such that $\mathrm{K}(P) \le \alpha$ and $d(x|P) \le \beta$; it exists according to the definition of stochasticity. Let $Q = F(P)$ be the image distribution. In other words, if ξ is a random variable with distribution P, then $F(\xi)$ has distribution Q. It is easy to see that $\mathrm{K}(Q) \le \mathrm{K}(P) + O(1)$, where the constant depends only on F. Indeed, Q is determined by P and F in a computable way. It remains to show that $d(F(x)|Q) \le d(x|P) + O(1)$.

The easiest way to show this is to recall the characterization of deficiency as the maximal lower semicomputable function such that

$$\sum_u 2^{d(u|S)} S(u) \le 1$$

for every distribution S. We may consider another function d' defined as

$$d'(u|S) = d(F(u)|F(S))$$

It is easy to see that

$$\sum_u 2^{d'(u|S)} S(u) = \sum_u 2^{d(F(u)|F(S))} S(u) = \sum_v 2^{d(v|F(S))} \cdot [F(S)](v) \le 1$$

(in the second equality we group all the values of u with the same $v = F(u)$). Therefore the maximality of d guarantees that $d'(u|S) \le d(u|S) + O(1)$, so we get the required inequality.

This proof can be also rephrased using the definition of stochasticity with a priori probability. We need to show that for $y = P(x)$ and $Q = F(P)$ we have

$$\frac{\mathbf{m}(y|Q)}{Q(y)} \le O(1) \cdot \frac{\mathbf{m}(x|P)}{P(x)}$$

or

$$\frac{\mathbf{m}(F(x)|F(P)) \cdot P(x)}{Q(F(x))} \le O(\mathbf{m}(x|P)).$$

It remains to note that the left hand side is a lower semicomputable function of x and P whose sum over all x (for every P) is at most 1. Indeed, if we group all terms with the same $F(x)$, we get the sum $\sum_y \mathbf{m}(y|F(P)) \le 1$, since the sum of $P(x)$ over all x with $F(x) = y$ equals $Q(y)$. □

Remark 2. In this proof it is important that we use the definition with distributions. If we replace is with the definition with finite sets, the results remains true with logarithmic precision, but the argument becomes more complicated, since the image of the uniform distribution may not be a uniform distribution. So if a set A is a good model for x, we should not use $F(A)$ as a model for $F(x)$. Instead, we should look at the maximal k such that $2^k \le \#F^{-1}(y)$, and consider the set of all y' that have at least 2^k preimages in A.

Remark 3. It is important in Proposition 3 that F is a total function. If x is some non-stochastic object and x^* is the shortest program for x, then x^* is incompressible and therefore stochastic. Still the interpreter (decompressor) maps x^* to x. We discuss the case of non-total F below, see Section 5.4.

Remark 4. A similar argument shows that $d(F(x)|F(P)) \le d(x|P) + \mathrm{K}(F) + O(1)$ (for total F), so both $O(1)$-bounds in Proposition 3 may be replaced by $\mathrm{K}(F) + O(1)$ where $O(1)$-constant does not depend on F anymore.

2.4 Non-stochastic objects

Note that up to now we have not shown that non-stochastic objects exist at all. It is easy to show that they exist for rather large values of α and β (linearly growing with n).

Proposition 4 ([32]). For some c and all n:
(1) if $\alpha + \beta < n - c \log n$, then there exist n-bit strings that are not (α, β)-stochastic;
(2) however, if $\alpha + \beta > n + c \log n$, then every n-bit string is (α, β)-stochastic.

Note that the term $c \log n$ allows us to use the definition with finite sets (i.e., uniform distributions on finite sets) instead of arbitrary finite distributions, since both versions are equivalent with $O(\log n)$-precision.

Proof. The second part is obvious (and is added just for comparison): if $\alpha+\beta = n$, then all n-bit strings can be split into 2^α groups of size 2^β each. Then the complexity of each group is $\alpha+O(\log n)$, and the randomness deficiency of every string in the corresponding group is at most $\beta + O(1)$. It is slightly bigger than the bounds we need, but we have reserve $c\log n$, and α and β can be decreased, say, by $(c/2)\log n$ before using this argument.

The first part: Consider all finite sets A of strings that have complexity at most α and size at most $2^{\alpha+\beta}$. Since $\alpha+(\alpha+\beta) < n$, they cannot cover all n-bit strings. Consider then the first (say, in the lexicographical order) n-bit string u not covered by any of these sets. What is the complexity of u? To specify u, it is enough to give n, α, β and the program of size at most α (from the definition of Kolmogorov complexity) that has maximal running time among programs of that size. Then we can wait until this program terminates and look at the outputs of all programs of size at most α after the same number of steps, select sets of strings of size at most $\alpha+\beta$, and take the first u not covered by these sets. So the complexity of u is at most $\alpha + O(\log n)$ (the last term is needed to specify n, α, β). The same is true for conditional complexity with arbitrary condition, since it is bounded by the unconditional complexity. So the randomness deficiency of u in every set A of size $2^{\alpha+\beta}$ is at least $\beta - O(\log n)$. We see that u is not $(\alpha, \beta - O(\log n))$-stochastic. Again the $O(\log n)$-term can be compensated by $O(\log n)$-change in β (we have $c\log n$ reserve for that). □

Remark 5. There is a gap between lower and upper bounds provided by Proposition 4. As we will see later, the upper bound (2) is tight with $O(\log n)$-precision, but we need more advanced technique (properties of two-part descriptions, Section 3) to prove this.

Proposition 4 shows that non-stochastic objects exist for rather large values of α and β (proportional to n). This, of course, is a mathematical existence result; it does not say anything about the possibility to observe non-stochastic objects in the "real world". As we have discussed, random sampling (from a simple distribution) may produce a non-stochastic object only with a negligible probability; *total* algorithmic transformations (defined by programs of small complexity) also cannot not create non-stochastic object from stochastic ones. What about non-total algorithmic transformations? As we have discussed in Remark 3, a non-total computable transformation may transform a stochastic object into a non-stochastic one, but does it happen with non-negligible probability?

Consider a randomized algorithm that outputs some string. It can be considered as a deterministic algorithm applied to random bit sequence (generated by the internal coin of the algorithm). This deterministic algorithm may be non-total, so we cannot apply the previous result. Still, as the following result shows, randomized algorithms also generate non-stochastic objects only with small probability.

To make this statement formal, we consider the sum of $\mathbf{m}(x)$ over all non-stochastic

x of length n. Since the a priori probability $\mathbf{m}(x)$ is the upper bound for the output distribution of any randomized algorithm, this implies the same bound (up to $O(1)$-factor) for every randomized algorithm. The following theorem gives an upper bound for this sum:

Proposition 5 (see [28], Section 10).

$$\sum\{\, \mathbf{m}(x) \mid x \text{ is a } n\text{-bit string that is not } (\alpha,\alpha)\text{-stochastic } \} \le 2^{-\alpha+O(\log n)}$$

for every n and α.

Proof. Consider the sum of $\mathbf{m}(x)$ over *all* strings of length n. This sum is some real number $\omega \le 1$. Let $\tilde{\omega}$ be the number represented by first α bits in the binary representation of ω, minus $2^{-\alpha}$. We may assume that $\alpha \le O(n)$, otherwise all strings of length n are (α,α)-stochastic.

Now construct a probability distribution as follows. All terms in a sum for ω are lower semicomputable, so we can enumerate increasing lower bounds for them. When the sum of these lower bounds exceeds $\tilde{\omega}$, we stop and get some measure P with finite support and rational values. Note that we have a measure, not a distribution, since the sum of $P(x)$ for all x is less than 1 (it does not exceed ω). So we normalize P (by some factor) do get a distribution $\tilde{P}$ proportional to P. The complexity of $\tilde{P}$ is bounded by $\alpha + O(\log n)$ (since $\tilde{P}$ is determined by $\tilde{\omega}$ and n). Note that the difference between P (without normalization factor) and a priori probability $\mathbf{m}$ (the sum of differences over all strings of length n) is bounded by $O(2^{-\alpha})$. It remains to show that for $\mathbf{m}$-most strings the distribution $\tilde{P}$ is a good model.

Let us prove that the sum of a priori probabilities of all n-bit strings x that have $d(x|\tilde{P}) > \alpha + c\log n$ is bounded by $O(2^{-\alpha})$, if c is large enough. Indeed, for those strings we have

$$\log \tilde{P}(x) - \mathrm{K}(x|\tilde{P}) > \alpha + c\log n.$$

The complexity of $\tilde{P}$ is bounded by $\alpha + O(\log n)$ and therefore $\mathrm{K}(x)$ exceeds $\mathrm{K}(x|\tilde{P})$ at most by $\alpha + O(\log n)$, so $-\log \tilde{P}(x) - \mathrm{K}(x) > 1$ (or $\tilde{P}(x) < \mathbf{m}(x)/2$) for those strings, if c is large enough (it should exceed the constants hidden in $O(\log n)$ notation). The difference 1 is enough for the estimate below, but we could have arbitrary constant or even logarithmic difference by choosing larger value of c.

Prefix complexity can be defined in terms of a priori probability, so we get

$$\log(\mathbf{m}(x)/\tilde{P}(x)) > 1$$

for all x that have deficiency exceeding $\alpha + c\log n$ with respect to $\tilde{P}$. The same inequality is true for P instead of $\tilde{P}$, since P is smaller. So for all those x we have $P(x) < \mathbf{m}(x)/2$,

or $(\mathbf{m}(x) - P(x)) > \mathbf{m}(x)/2$. Recalling that the sum of $\mathbf{m}(x) - P(x)$ over all x of length n does not exceed $O(2^{-\alpha})$ by construction of $\tilde{\omega}$, we conclude that the sum of $\mathbf{m}(x)$ over all strings of randomness deficiency (with respect to $\tilde{P}$) exceeding $\alpha + c\log n$ is at most $O(2^{-\alpha})$.

So we have shown that the sum of $\mathbf{m}(x)$ for all x of length n that are not $(\alpha + O(\log n), \alpha + O(\log n))$-stochastic, does not exceed $O(2^{-\alpha})$. This differs from our claim only by $O(\log n)$-change in α. □

This result shows that non-stochastic objects rarely appear as outputs of randomized algorithms. There is an explanation of this phenomenon (that goes back to Levin): non-stochastic objects provide a lot of information about halting problem, and the probability of appearance of an object that has a lot of information about some sequence α, is small (for any fixed α). We discuss this argument below, see Section 4.6.

It is natural to ask the following general question. For a given string x, we may consider the set of all pairs (α, β) such that x is (α, β)-stochastic. By definition, this set is upwards-closed: a point in this set remains in it if we increase α or β, so there is some boundary curve that describes the trade-off between α and β. What curves could appear in this way? To get an answer (to characterizes all these curves with $O(\log n)$-precision), we need some other technique, explained in the next section.

3 Two-part descriptions

Now we switch to another measure of the quality of a statistical model. It is important both for philosophical and technical reasons. The philosophical reason is that it corresponds to the so-called "minimal description length principle". The technical reason is that it is easier to deal with; in particular, we will use it to answer the question asked at the end of the previous section.

3.1 Optimality deficiency

Consider again some statistical model. Let P be a probability distribution (with finite support and rational values) on strings. Then we have

$$\mathrm{K}(x) \le \mathrm{K}(P) + \mathrm{K}(x|P) \le \mathrm{K}(P) + (-\log P(x))$$

for arbitrary string x (with $O(1)$-precision). Here we use that (with $O(1)$-precision):

- $\mathrm{K}(x|P) \le -\log P(x)$, as we have mentioned;

- the complexity of the pair is bounded by the sum of complexities: $\mathrm{K}(u,v) \le \mathrm{K}(u) + \mathrm{K}(v)$;
- $\mathrm{K}(v) \le \mathrm{K}(u,v)$ (in our case, $\mathrm{K}(x) \le KP(x,P)$).

If P is a uniform distribution on some finite set A, this inequality can be explained as follows. We can specify x in two steps:

- first, we specify A;
- then we specify the ordinal number of x in A (in some natural ordering, say, the lexicographic one).

In this way we get $\mathrm{K}(x) \le \mathrm{K}(A) + \log \#A$ for every element x of arbitrary finite set A. This inequality holds with $O(1)$-precision. If we replace the prefix complexity by the plain version, we can say that $\mathrm{C}(x) \le \mathrm{C}(A) + \log \#A$ with precision $O(\log n)$ for every string x of length at most n: we may assume without loss of generality that both terms in the right hand side are at most n, otherwise the inequality is trivial.

The "quality" of a statistical model P for a string x can be measured by the difference between sides of this inequality: for a good model the "two-part description" should be almost minimal. We come to the following definition:

Definition 2. The *optimality deficiency* of a distribution P considered as the model for a string x is the difference

$$\delta(x,P) = (\mathrm{K}(P) + (-\log P(x))) - \mathrm{K}(x).$$

As we have seen, $\delta(x,P) \ge 0$ with $O(1)$-precision.

If P is a uniform distribution on a set A, the optimality deficiency $\delta(x,P)$ will also be denoted by $\delta(x,A)$, and

$$\delta(x,A) = (\mathrm{K}(A) + \log \#A) - \mathrm{K}(x).$$

The following proposition shows that we may restrict our attention to finite sets as models (with $O(\log n)$-precision):

Proposition 6. Let P be a distribution considered as a model for some string x of length n. Then there exists a finite set A such that

$$\mathrm{K}(A) \le \mathrm{K}(P) + O(\log n); \quad \log \#A \le -\log P(x) + O(1) \qquad (*)$$

This proposition will be used in many arguments, since it is often easier to deal with sets as statistical models (instead of distributions). Note that the inequalities $(*)$ evidently imply that

$$\delta(x, A) \le \delta(x, P) + O(\log n),$$

so arbitrary distribution P may be replaced by a uniform one (U_A) with a logarithmic-only change in the optimality deficiency.

Proof. We use the same construction as in Proposition 2. Let 2^{-k} be the maximal power of 2 such that $2^{-k} \le P(x)$, and let $A = \{x \mid P(x) \ge 2^{-k}\}$. Then $k = -\log P(x) + O(1)$. We may assume that $k = O(n)$: if k is much bigger than n, then $\delta(x, P)$ is also bigger than n (since the complexity of x is bounded by $n + O(\log n)$), and in this case the statement is trivial (let A be the set of all n-bit strings).

Now we see that that A is determined by P and k, so $\mathrm{K}(A) \le \mathrm{K}(P) + \mathrm{K}(k) \le \mathrm{K}(P) + O(\log n)$. Note also that $\#A \le 2^k$, so $\log \#A \le -\log P(x) + O(1)$. □

3.2 Optimality and randomness deficiencies

Now we have two "quality measures" for a statistical model P: the randomness deficiency $d(x|P)$ and the optimality deficiency $\delta(x, P)$. They are related:

Proposition 7.

$$d(x|P) \le \delta(x, P)$$

with $O(1)$-precision.

Proof. By definition

$$d(x|P) = -\log P(x) - \mathrm{K}(x|P);$$
$$\delta(x, P) = -\log P(x) + \mathrm{K}(P) - \mathrm{K}(x).$$

It remains to note that $\mathrm{K}(x) \le \mathrm{K}(x, P) \le \mathrm{K}(P) + \mathrm{K}(x|P)$ with $O(1)$-precision. □

Could $\delta(x, P)$ be significantly larger than $d(x|P)$? Look at the proof above: the second inequality $\mathrm{K}(x, P) = \mathrm{K}(P) + \mathrm{K}(x|P)$ is an equality with logarithmic precision. Indeed, the exact formula (Levin–Gács formula for the complexity of a pair with $O(1)$-precision) is

$$\mathrm{K}(x, P) = \mathrm{K}(P) + \mathrm{K}(x|P, \mathrm{K}(P)).$$

Here the term $\mathrm{K}(P)$ in the condition changes the complexity by $O(\log \mathrm{K}(P))$, and we may ignore models P whose complexity is much greater than the complexity of x.

On the other hand, in the first inequality the difference between $\mathrm{K}(x, P)$ and $\mathrm{K}(x)$ may be significant. This difference equals $\mathrm{K}(P|x)$ with logarithmic accuracy and, if it is large, then $\delta(x, P)$ is much bigger than $d(x|P)$. The following example shows that this is possible. In this example we deal with sets as models.

Example 1. Consider an incompressible string x of length n, so $\mathrm{K}(x) = n$ (all equalities with logarithmic precision). A good model for this string is the set A of all n-bit strings. For this model we have $\#A = 2^n$, $\mathrm{K}(A) = 0$ and $\delta(x, A) = n+0-n = 0$ (all equalities have logarithmic precision). So $d(x|P) = 0$, too. Now we can change the model by excluding some other n-bit string. Consider a n-bit string y that is incompressible and independent of x: this means that $\mathrm{K}(x, y) = 2n$. Let A' be $A \setminus \{y\}$.

The set A' contains x (since x and y are independent, y differs from x). Its complexity is n (since it determines y). The optimality deficiency is then $n + n - n = n$, but the randomness deficiency is still small: $d(x|A') = \log \#A' - \mathrm{K}(x|A') = n - n = 0$ (with logarithmic precision). To see why $\mathrm{K}(A'|x) = n$, note that x and y are independent, and the set A' has the same information as (n, y).

One of the main results of this section (Theorem 3) clarifies the situation: it implies that if optimality deficiency of a model is significantly larger than its randomness deficiency, then this model can be improved and another model with better parameters can be found. More specifically, the complexity of the new model is smaller than the complexity of the original one while both the randomness deficiency and optimality deficiency of the new model are not worse than the randomness deficiency of the original one. This is one of the main results of algorithmic statistics, but first let us explore systematically the properties of two-part descriptions.

3.3 Trade-off between complexity and size of a model

It is convenient to consider only models that are sets (=uniform distribution on sets). We will call them *descriptions*. Note that by Propositions 2 and 6 this restriction does not matter much since we ignore logarithmic terms. For a given string x there are many different descriptions: we can have a simple large set containing x, and at the same time some more complicated, but smaller one. In this section we study the trade-off between these two parameters (complexity and size).

Definition 3. A finite set A is an $(i * j)$-description[4] of x if $x \in A$, complexity $\mathrm{K}(A)$ is at most i, and $\log \#A \leq j$. For a given x we consider the set P_x of all pairs (i, j) such that x has some $(i * j)$-description; this set can be called the *profile* of x.

[4]This notation may look strange; however, we speak so often about finite sets of complexity at most i and cardinality at most 2^j that we decided to introduce some short name and notation for them.

Informally speaking, an $(i * j)$-description for x consists of two parts: first we spend i bits to specify some finite set A and then j bits to specify x as an element of A.

What can be said about P_x for a string x of length n and complexity $k = \mathrm{K}(x)$? By definition, P_x is closed upwards and contains the points $(0, n)$ and $(k, 0)$. Here we omit terms $O(\log n)$: more precisely, we have $((\log n) * O(1))$-description that consists of all string of length n, and a $((k + O(1)) * 0)$-description $\{x\}$. Moreover, the following proposition shows that we can move the information from the second part of the description into its first part (leaving the total length almost unchanged). In this way we make the set smaller (the price we pay is that its complexity increases).

Proposition 8 ([14, 12, 33]). Let x be a string and A be a finite set that contains x. Let s be a non-negative integer such that $s \leq \log \#A$. Then there exists a finite set A' containing x such that $\#A' \leq \#A/2^s$ and $\mathrm{K}(A') \leq \mathrm{K}(A) + s + O(\log s)$.

Proof. List all the elements of A in some (say, lexicographic) order. Then we split the list into 2^s parts (first $\#A/2^s$ elements, next $\#A/2^s$ elements etc.; we omit evident precautions for the case when $\#A$ is not a multiple of 2^s). Then let A' be the part that contains x. It has the required size. To specify A', it is enough to specify A and the part number; the latter takes at most s bits. (The logarithmic term is needed to make the encoding of the part number self-delimiting.) □

This statement can be illustrated graphically. As we have said, the set P_x is "closed upwards" and contains with each point (i, j) all points on the right (with bigger i) and on the top (with bigger j). It contains points $(0, n)$ and $(0, \mathrm{K}(x))$; Proposition 8 says that we can also move down-right adding $\langle s, -s\rangle$ (with logarithmic precision). We will see that movement in the opposite direction is not always possible. So, having two-parts descriptions with the same total length, we should prefer the one with bigger set (since it always can be converted into others, but not vice versa).

The boundary of P_x is some curve connecting the points $(0, n)$ and $(k, 0)$. This curve (introduced by Kolmogorov in 1970s, see [15]) never gets into the triangle $i + j < \mathrm{K}(x)$ and always goes down (when moving from left to right) with slope at least -1 or more.

This picture raises a natural question: which boundary curves are possible and which are not? Is it possible, for example, that the boundary goes along the dotted line on Figure 1? The answer is positive: take a random string of desired complexity and add trailing zeros to achieve desired length. Then the point $\langle 0, \mathrm{K}(x)\rangle$ (the left end of the dotted line) corresponds to the set A of all strings of the same length having the same trailing zeros. We know that the boundary curve cannot go down slower than with slope -1 and that it lies above the line $i + j = \mathrm{K}(x)$, therefore it follows the dotted line (with logarithmic precision).

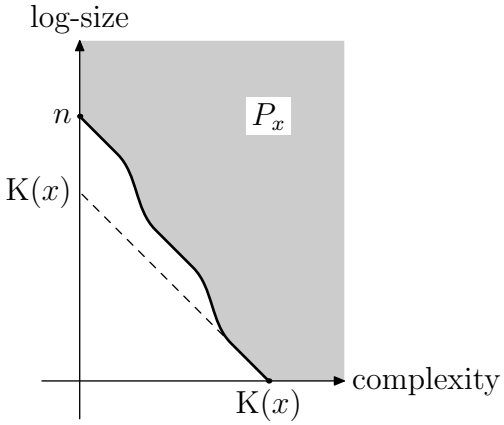

Figure 1: The set P_x and its boundary curve

A more difficult question: is it possible that the boundary curve starts from $\langle 0, n\rangle$, goes with the slope -1 to the very end and then goes down rapidly to $\langle \mathrm{K}(x), 0\rangle$ (Figure 2, the solid line)? Such a string x, informally speaking, would have essentially only two types of statistical explanations: a set of all strings of length n (and its parts obtained by Proposition 8) and the exact description, the singleton $\{x\}$.

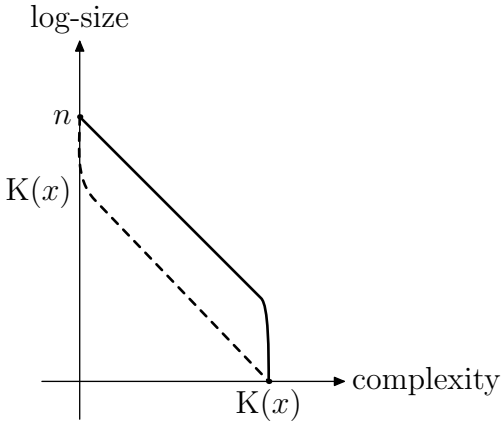

Figure 2: Two opposite possibilities for a boundary curve

It turns out that not only these two opposite cases are possible, but also all intermediate curves (provided they decrease with slope -1 or faster, and are simple enough), at least with logarithmic precision. More precisely, the following statement holds:

Theorem 1 ([42]). *Let $k \le n$ be two integers and let $t_0 > t_1 > \ldots > t_k$ be a strictly decreasing sequence of integers such that $t_0 \le n$ and $t_k = 0$; let m be the complexity of this sequence. Then there exists a string x of complexity $k + O(\log n) + O(m)$ and length $n + O(\log n) + O(m)$ for which the boundary curve of P_x coincides with the line $(0, t_0)$–$(1, t_1)$–…–(k, t_k) with $O(\log n) + O(m)$ precision: the distance between the set P_x and the set $T = \{\langle i, j\rangle \mid (i < k) \Rightarrow (j > t_i)\}$ is bounded by $O(\log n) + O(m)$.*

(We say that the distance between two subsets $P, Q \subset \mathbb{Z}^2$ is at most ε if P is contained in the ε-neighborhood of Q and vice versa.)

Proof. For every i in the range $0 \ldots k$ we list all the sets of complexity at most i and size at most 2^{t_i}. For a given i the union of all these sets is denoted by S_i. It contains at most 2^{i+t_i} elements. (Here and later we omit constant factors and factors polynomial in n when estimating cardinalities, since they correspond to $O(\log n)$ additive terms for lengths and complexities.) Since the sequence t_i strictly decreases (this corresponds to slope -1 in the picture), the sums $i + t_i$ do not increase, therefore each S_i has at most $2^{t_0} \le 2^n$ elements. The union of all S_i therefore also has at most 2^n elements (up to a polynomial factor, see above). Therefore, we can find a string of length n (actually $n + O(\log n)$) that does not belong to any S_i. Let x be a first such string in some order (e.g., in the lexicographic order).

By construction, the set P_x lies above the curve determined by t_i. So we need to estimate the complexity of x and prove that P_x follows the curve (i.e., that T is contained in the neighborhood of P_x).

Let us start with the upper bound for the complexity of x. The list of all objects of complexity at most k plus the full table of their complexities have complexity $k+O(\log k)$, since it is enough to know k and the number of terminating programs of length at most k. Except for this list, to specify x we need to know n and the sequence $t_0, \ldots, t_k$, whose complexity is m.

The lower bound: the complexity of x cannot be less than k since all the singletons of this complexity were excluded (via S_k).

It remains to show that for every $i \le k$ we can put x into a set A of complexity i (or slightly bigger) and size 2^{t_i} (or slightly bigger). For this we enumerate a sequence of sets of correct size and show that one of the sets will have the required properties; if this sequence of sets is not very long, the complexity of its elements is bounded. Here are the details.

We start by taking the first 2^{t_i} strings of length n as our first set A. Then we start enumerating all finite sets of complexity at most j and of size at most 2^{t_j} for all $j = 0, \ldots, k$, and get an enumeration of all sets S_j. Recall that all elements of all S_j should be deleted (and the minimal remaining element should eventually be x). So, when a new set of

complexity at most j and of size at most 2^{t_j} appears, all its elements are included in S_j and deleted. Until all elements of A are deleted, we have nothing to worry about, since A is covering the minimal remaining element. If (and when) all elements of A are deleted, we replace A by a new set that consists of first 2^{t_i} undeleted (yet) strings of length n. Then we wait again until all the elements of this new A are deleted, if (and when) this happens, we take 2^{t_i} first undeleted elements as new A, etc.

The construction guarantees the correct size of the sets and that one of them covers x (the minimal non-deleted element). It remains to estimate the complexity of the sets we construct in this way.

First, to start the process that generates these sets, we need to know the length n (actually something logarithmically close to n) and the sequence $t_0, \ldots, t_k$. In total we need $m + O(\log n)$ bits. To specify each version of A, we need to add its version number. So we need to show that the number of different A's that appear in the process is at most 2^i or slightly bigger.

A new set A is created when all the elements of the old A are deleted. These changes can be split into two groups. Sometimes a new set of complexity j appears with $j \leq i$. This can happen only $O(2^i)$ times since there are at most $O(2^i)$ sets of complexity at most i. So we may consider the other changes (excluding the first changes after each new large set was added). For those changes all the elements of A are gone due to elements of S_j with $j > i$. We have at most 2^{j+t_j} elements in S_j. Since $t_j + j \leq t_i + i$, the total number of deleted elements only slightly exceeds 2^{t_i+i}, and each set A consists of 2^{t_i} elements, so we get about 2^i changes of A. □

Remark 6. It is easy to modify the proof to get a string x of length exactly n. Indeed, we may consider slightly smaller bad sets: decreasing the logarithms of their sizes by $O(\log n)$, we can guarantee that the total number of elements in all bad sets is less than 2^n. Then there exists a string of length n that does not belong to bad sets. In this way the distance between T and P_x may increase by $O(\log n)$, and this is acceptable.

Theorem 1 shows that the value of the complexity of x does not describe the properties of x fully; different strings of the same complexity x can have different boundary curves of P_x. This curve can be considered as an "infinite-dimensional" characterization of x.

Strings x with minimal possible P_x (Figure 2, the upper curve) may be called *antistochastic*. They have quite unexpected properties. For example, if we replace some bits of an antistochastic string x by stars (or some other symbols indicating erasures) leaving only $\mathrm{K}(x)$ non-erased bits, then the string x can be reconstructed from the resulting string x' with logarithmic advice. This and other properties of antistochastic strings were discovered in [23].

3.4 Optimality and randomness deficiency

In this section we establish the connection between optimality and randomness deficiencies. As we have seen, the optimality deficiency can be bigger than the randomness deficiency (for the same description), and the difference is $\delta(x, A) - d(x|A) = \mathrm{K}(A) + \mathrm{K}(x|A) - \mathrm{K}(x)$. The Levin–Gács formula for the complexity of pair ($\mathrm{K}(u, v) = \mathrm{K}(u) + \mathrm{K}(v|u)$ with logarithmic precision, for $O(1)$-precision one needs to add $\mathrm{K}(u)$ in the condition, but we ignore logarithmic size terms anyway) shows that the difference in question can be rewritten as

$$\delta(x, A) - d(x|A) = \mathrm{K}(A, x) - \mathrm{K}(x) = \mathrm{K}(A|x).$$

So if the difference between deficiencies for some $(i * j)$-description A of x is big, then $\mathrm{K}(A|x)$ is big. All the $(i * j)$-descriptions of x can be enumerated if x, i, and j are given. So the large value of $\mathrm{K}(A|x)$ for some $(i * j)$-description A means that there are many $(i * j)$-description of x, otherwise A can be reconstructed from x by specifying i, j (requires $O(\log n)$ bits) and the ordinal number of A in the enumeration. We will prove that if there are many $(i * j)$-descriptions for some x, then there exist a description with better parameters.

Now we explain this in more details. Let us start with the following remark. Consider all strings that have $(i * j)$-descriptions for some fixed i and j. They can be enumerated in the following way: we enumerate all finite sets of complexity at most i, select those sets that have size at most 2^j, and include all elements of these sets into enumeration. In this construction

- the complexity of the enumerating algorithm is logarithmic (it is enough to know i and j);
- we enumerate at most 2^{i+j} elements;
- the enumeration is divided into at most 2^i "portions" of size at most 2^j.

It is easy to see that any other enumeration process with these properties enumerates only objects that have $(i * j)$-descriptions (again with logarithmic precision). Indeed, each portion is a finite set that can be specified by its ordinal number and the enumeration algorithm, the first part requires $i + O(\log i)$ bits, the second is of logarithmic size according to our assumption.

Remark 7. The requirement about the portion size is redundant. Indeed, we can change the algorithm by splitting large portions into pieces of size 2^j (the last piece may be incomplete). This, of course, increases the number of portions, but if the total number of

enumerated elements is at most 2^{i+j}, then this splitting adds at most 2^i pieces. This observation looks (and is) trivial, still it plays an important role in the proof of the following proposition.

Proposition 9. If a string x of length n has at least 2^k different (i,j)-descriptions, then x has some $(i*(j-k))$-description and even some $((i-k)*j)$-description.

Again we omit logarithmic term: in fact one should write $((i+O(\log n))*(j-k+O(\log n)))$, etc. The word "even" in the statement refers to Proposition 8 that shows that indeed the second claim is stronger.

Proof. Consider the enumeration of all objects having $(i*j)$-descriptions in 2^i portions of size 2^j (we ignore logarithmic additive terms and respective polynomial factors) as explained above. After each portion (i.e., new $(i*j)$-description) appears, we count the number of descriptions for each enumerated object and select objects that have at least 2^k descriptions. Consider a new enumeration process that enumerates only these "rich" objects (rich = having many descriptions). We have at most 2^{i+j-k} rich objects (since they appear in the list of size 2^{i+j} with multiplicity 2^k), enumerated in 2^i portions (new portion of rich objects may appear only when a new portion appears in the original enumeration). So we apply the observation above to conclude that all rich objects have $(i*(j-k))$-descriptions.

To get the second (stronger) statement we need to decrease the number of portions (while not increasing too much the number of enumerated objects). This can be done using the following trick: when a new rich object (having 2^k descriptions) appears, we enumerate not only rich objects, but also "half-rich" objects, i.e., objects that currently have at least $2^k/2$ descriptions. In this way we enumerate more objects — but only twice more. At the same time, after we dumped all half-rich objects, we are sure that next $2^k/2$ new $(i*j)$-descriptions will not create new rich objects, so the number of portions is divided by $2^k/2$, as required. □

Let us say more accurately how we deal with logarithmic terms. We may assume that $i,j = O(n)$, otherwise the claim is trivial. Then we allow polynomial (in n) factors and $O(\log n)$ additive terms in all our considerations.

Remark 8. If we unfold this construction, we see that new descriptions (of smaller complexity) are not selected from the original sequence of descriptions but constructed from scratch. In Section 6 we deal with much more complicated case where we restrict ourselves to descriptions from some class (say, Hamming balls). Then the proof given above does not work, since the description we construct is not a ball even if we start with ball descriptions. Still some other (much more ingenious) argument can be used to prove a similar result for the restricted case.

Now we are ready to prove the promised results (see the discussion after Example 1).

Theorem 2. *If a string x of length n is (α,β)-stochastic, then there exists some finite set B containing x such that $\mathrm{K}(B) \le \alpha + O(\log n)$ and $\delta(x,B) \le \beta + O(\log n)$.*

Proof. Since x is (α,β)-stochastic, there exists some finite set A such that $\mathrm{K}(A) \le \alpha$ and $d(x|A) \le \beta$. Let $i = \mathrm{K}(A)$ and $j = \log \#A$, so A is an $(i*j)$-description of x. We may assume without loss of generality that both α and β (and therefore i and j) are $O(n)$, otherwise the statement is trivial. The value $\delta(x,A)$ may exceed $d(x|A)$, as we have discussed at the beginning of this section. So we assume that

$$k = \delta(x,A) - d(x|A) > 0;$$

if not, we can let $B = A$. Then, as we have seen, $\mathrm{K}(A|x) \ge k - O(\log n)$, and there are at least $2^{k-O(\log n)}$ different $(i*j)$-descriptions of x. According to Proposition 9, there exists some finite set B that is an $(i*(j-k+O(\log n)))$-description of x. Its optimality deficiency $\delta(x,B)$ is $(k-O(\log n))$-smaller (compared to A) and therefore $O(\log n)$-close to $d(x|A)$. □

In this argument we used the simple part of Proposition 9. Using the stronger statement about complexity decrease, we get the following result:

Theorem 3 ([42]). *Let A be a finite set containing a string x of length n and let $k = \delta(x,A) - d(x|A)$. Then there is a finite set B containing x such that $\mathrm{K}(B) \le \mathrm{K}(A) - k + O(\log n)$ and $\delta(x,B) \le d(x|A) + O(\log n)$.*

Proof. Indeed, if B is an $((i-k)*j)$-description of x (up to logarithmic terms, as usual), then its optimality deficiency is again $(k - O(\log n))$-smaller (compared to A) and therefore $O(\log n)$-close to $d(x|A)$. □

Note that the statement of the theorem implies that $d(x|B) \le d(x|A) + O(\log n)$.

Theorem 2 and Proposition 7 show that we can replace the randomness deficiency in the definition of (α,β)-stochastic strings by the optimality deficiency (with logarithmic precision). More specifically, for every string x of length n consider the sets

$$Q_x = \{\langle \alpha,\beta\rangle \mid x \text{ is } (\alpha,\beta)\text{-stochastic}\},$$

and

$$\tilde{Q}_x = \{\langle \alpha,\beta\rangle \mid \text{there exists } A \ni x \text{ with } \mathrm{K}(A) \le \alpha,\ \delta(x,A) \le \beta)\}.$$

Then these sets are at most $O(\log n)$ apart (each is contained in the $O(\log n)$-neighborhood of the other one).

This remark, together with the existence of antistochastic strings of given complexity and length, allows us to improve the result about the existence of non-stochastic objects (Proposition 4).

Proposition 10 ([12, Theorem IV.2]). For some c and for all n: if $\alpha+\beta < n - c\log n$, there exist strings of length n that are not (α,β)-stochastic.

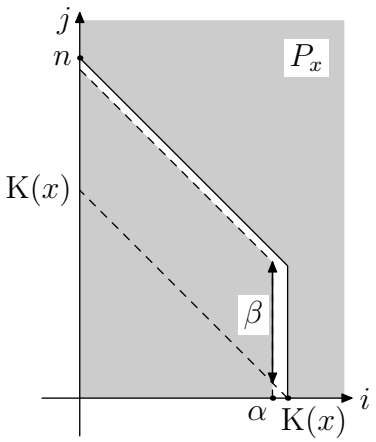

Figure 3: Non-stochastic strings revisited. Left grey area corresponds to descriptions A with $\mathrm{K}(A) \le \alpha$ and $\delta(x, A) \le \beta$.

Proof. Assume that integers n, α, β are given such that $\alpha + \beta < n - c\log n$ (where the constant c will be chosen later). Let x be an antistochastic string of length n that has complexity $\alpha+d$ where d is some positive number (see below about the choice of d). More precisely, for every given d there exists a string x whose complexity is $\alpha + d + O(\log n)$, length is $n + O(\log n)$, and the set P_x is $O(\log n)$-close to the upper gray area (Figure 3).

Assume that x is (α,β)-stochastic. Then (Theorem 2) the string x has an $(i * j)$-description with $i \le \alpha$ and $i + j \le \mathrm{K}(x) + \beta$ (with logarithmic precision). The set of pairs (i, j) satisfying these inequalities is shown as the lower gray area. We have to choose c in such a way that for some d these two gray are disjoint and even separated by a gap of logarithmic size (since they are known only with $O(\log n)$-precision). Note first that for $d = c'\log n$ with large enough c' we guarantee the vertical gap (the vertical segments of the boundaries of two gray areas are far apart). Then we select c large enough to guarantee that the diagonal segments of the boundaries of two gray areas are far apart ($\alpha+\beta < n$ with logarithmic margin). □

The transition from randomness deficiency to optimality deficiency (Theorem 2) has the following geometric interpretation.

Theorem 4. *The sets Q_x and P_x are related to each other via an affine transformation $(\alpha, \beta) \mapsto (\alpha, \mathrm{K}(x) - \alpha + \beta)$, as Figure 4 shows.*[5]

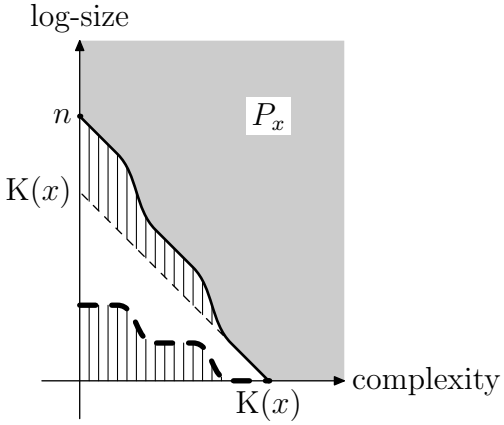

Figure 4: The set P_x and the boundary of the set Q_x (bold dotted line); on every vertical line two intervals have the same length.

As usual, this statement is true with logarithmic accuracy: the distance between the image of the set Q_x under this transformation and the set P_x is claimed to be $O(\log n)$ for string x of length n.

Proof. As we have seen, we may use the optimality deficiency instead of randomness deficiency, i.e., use the set $\tilde{Q}_x$ in place of Q_x. The preimage of the pair (i, j) under our affine transformation is the pair $(i, i + j - \mathrm{K}(x))$. Hence we have to prove that a pair (i, j) is in P_x if and only if the pair $(i, i + j - \mathrm{K}(x))$ is in $\tilde{Q}_x$. Note that $\mathrm{K}(A) = i$ and $\log \#A = j$ is equivalent to $\mathrm{K}(A) = i$ and $\delta(x, A) = i + j - \mathrm{K}(x)$ just by definition of $\delta(x, A)$. (See Figure 4: the optimality deficiency of a description A with $\mathrm{K}(A) = i$ and $\log \#A = j$ is the vertical distance between (i, j) and the dotted line.)

But there is some technical problem: in the definition of P_x we used inequalities $\mathrm{K}(A) \leq i$ and $\log \#A \leq j$, not the equalities $\mathrm{K}(A) = i$ and $\log \#A = j$. The same applies to the definition of $\tilde{Q}_x$. So we have two sets that correspond to each other, but their $\leq$-closures could be different. Obviously, $\mathrm{K}(A) \leq i$ and $\log \#A \leq j$ imply $\mathrm{K}(A) \leq i$ and $\mathrm{K}(A) + \log \#A - \mathrm{K}(x) \leq i + j - \mathrm{K}(x)$, but not vice versa.

In other words, the set of pairs $(\mathrm{K}(A), \log \#A)$ satisfying the latter inequalities (see the right set on Figure 5) is bigger than the set of pairs $(\mathrm{K}(A), \log \#A)$ satisfying the

[5]Technically speaking, this holds only for $\alpha \leq \mathrm{K}(x)$. For $\alpha > \mathrm{K}(x)$ both sets contain all pairs with first component α.

former inequalities (see the left set on Figure 5). Now Proposition 8 helps: we may use it

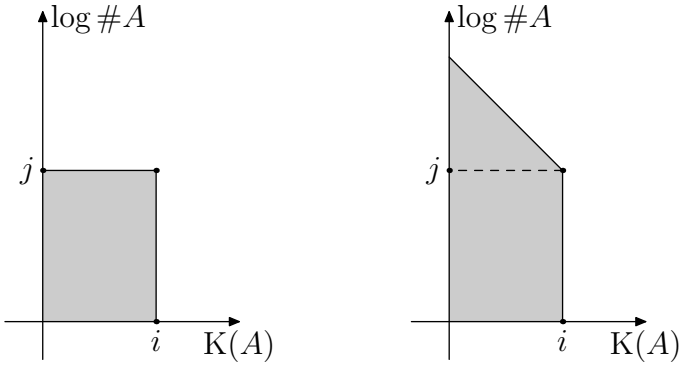

Figure 5: The left picture shows (for given i and j) the set of all pairs $(\mathrm{K}(A), \log \#A)$ such that $\mathrm{K}(A) \le i$ and $\log \#A \le j$; the right picture shows the pairs $(\mathrm{K}(A), \log \#A)$ such that $\mathrm{K}(A) \le i$ and $\delta(x, A) \le i + j - \mathrm{K}(x)$.

to convert any set with parameters from the right region into a set with parameters from the left region. □

Remark 9. Let us stress again that Theorem 2 claims only that the *existence* of a set $A \ni x$ with $\mathrm{K}(A) \le \alpha$ and $d(x|A) \le \beta$ is equivalent to the existence of a set $B \ni x$ with $\mathrm{K}(B) \le \alpha$ and $\delta(x|A) \le \beta$ (with logarithmic accuracy). The theorem does *not* claim that for *every* set $A \ni x$ with complexity at most α the inequalities $d(x|A) \le \beta$ and $\delta(x, A) \le \beta$ are equivalent (with logarithmic accuracy). Indeed, the Example 1 shows that this is not true: the first inequality does not imply the second one in general case. However, Theorems 2 and 3 show that this can happen only for non-minimal descriptions (for which the description with smaller complexity and the same optimality deficiency) exists. Later we will see that all the minimal descriptions of the same (or almost the same) complexity have almost the same information. Moreover, if A and B are minimal descriptions and the complexity of A is less than that of B then $\mathrm{C}(A|B)$ is small.

For the people with taste for philosophical speculations the meaning of Theorems 2 and 3 can be advertised as follows. Imagine several scientists that compete in providing a good explanation for some data x. Each explanation is a finite set A containing x together with a program p that computes A.

How should we compare different explanations? We want the randomness deficiency $d(x|A)$ of x in A to be negligible (no features of x remain unexplained). Among these descriptions we want to find the simplest one (with the shortest p). That is, we look

for a set A corresponding to the point where the bold dotted line on Fig. 4 touches the horizontal axis. (In fact, there is always some trade-off between the parameters, not the specific exact point where the curve touches the horizontal axis, but we want to keep the discussion simple though imprecise.)

However, this approach meets the following obstacle: we are unable to compute randomness deficiency $d(x|A)$. Moreover, the inventor of the model A has no ways to convince us that the deficiency is indeed negligible if it the case (the function $d(x|A)$ is not even upper semicomputable). What could be done? Instead, we may look for an explanation with (almost) minimal sum $\log \#A + |p|$ (minimum description length principle). Note that this quantity is known for competing explanation proposals. Theorems 2 and 3 provide the connection between these two approaches.

Returning to mathematical language, we have seen in this section that two approaches (based on $(i*j)$-descriptions and (α,β)-stochasticity) produce essentially the same curve, though in different coordinates. The other ways to get the same curve will be discussed in Sections 4 and 5.

3.5 Historical remarks

The idea to consider $(i*j)$-descriptions with optimal parameters can be traced back to Kolmogorov. There is a short record for his talk given in 1974 [15]. Here is the (full) translation of this note:

> For every constructive object x we may consider a function $\Phi_x(k)$ of an integer argument $k \geq 0$ defined as a logarithm of the minimal cardinality of a set of complexity at most k containing x. If x itself has a simple definition, then $\Phi_x(1)$ is equal to one [a typo: cardinality equals 1, and logarithm equals 0] already for small k. If such a simple definition does not exist, x is "random" in the negative sense of the word "random". But x is positively "probabilistically random" only if the function Φ has a value Φ_0 for some relatively small k and then decreases approximately as $\Phi(k) = \Phi_0 - (k - k_0)$. [This corresponds to approximate $(k_0, 0)$-stochasticity.]

Kolmogorov also gave a talk in 1974 [14]; the content of this talk was reported by Cover [10, Section 4, page 31]:

> 4. **Kolmogorov's H_k Function**
>
> Consider the function $H_k\colon \{0,1\}^k \to N$, $H_k(x) = \min_{p\colon l(p)\leq k} \log |S|$, where the minimum is taken over all subsets $S \subseteq \{0,1\}^n$, such that $x \in S$, $U(p) = S$, $l(p) \leq k$. This definition was introduces by Kolmogorov in a talk at the

> Information Symposium, Tallinn, Estonia, in 1974. Thus $H_k(x)$ is the log of the size of the smallest set containing x over all sets specifiable by a program of k or fewer bits. Of special interest is the value
>
> $$k^*(x) = \min\{k\colon H_k(x) + k = K(x)\}.$$
>
> Note that $\log|S|$ is the maximal number of bits necessary to describe an arbitrary element $x \in S$. Thus a program for x can be written in two stages: "Use p to print the indicator function for S; the desired sequence is the ith sequence in a lexicographic ordering of the elements of this set". This program has length $l(p) + \log|S|$, and $k^*(x)$ is the length of the shortest program p for which this 2-stage description is as short as the best 1-stage description p^*. We observe that x must be maximally random with respect to S — otherwise the 2-stage description could be improved, contradicting the minimality of $K(x)$. Thus $k^*(x)$ and its associated program p constitute a minimal sufficient description for x. ⟨...⟩
>
> Arguments can be provided to establish that $k^*(x)$ and its associated set S^* describe all of the "structure" of x. The remaining details about x are conditionally maximally complex. Thus pp^{**}, the program for S^*, plays the role of a sufficient statistic.

In both places Kolmogorov speaks about the place when the boundary curve of P_x reaches its lower bound determined by the complexity of x.

Later the same ideas were rediscovered and popularized by many people. Koppel in [17] reformulates the definition using total algorithms. Instead of a finite set A he considered a total program P that terminates on all strings of some length. The two-part description of some x is then formed by this program P and the input D for this program that is mapped to x. In our terminology this corresponds to the set A of all values of P on the strings of the same length as D. He writes then [17, p. 1089]

> **Definition 3**. The c-sophistication of a finite string S [is defined as]
>
> $$\mathrm{SOPH}_c(S) = \min\{|P| \mid \exists D \text{ s. t. } (P, D) \text{ is a } c\text{-minimal description of } \alpha\}.$$

There is a typo in this paper: S should be replaced by α (two times). Before in Definition 1 the description is called c-minimal if $|P| + |D| \le H(\alpha) + c$ (here P and D are the program and and its input, respectively, H stands for complexity).

Though this paper (as well as the subsequent papers [18, 19]) is not technically clear (e.g., it does not say what are the requirements for the algorithm U used in the definition,

and in [18, 19] only universality is required, which is not enough: if U is not optimal, the definition does not make sense), the philosophic motivation for this notion is explained clearly [17, p. 1087]:

> The total complexity of an object is defined as the size of its most concise description. The total complexity of an object can be large while its "meaningful" complexity is low; for example, a random object is by definition maximally complex but completely lacking in structure.
>
> ⟨...⟩ The "static" approach to the formalization of meaningful complexity is "sophistication" defined and discussed by Koppel and Atlan [reference to unpublished paper "Program-length complexity, sophistication, and induction" is given, but later a paper of same authors [19] with a similar title appeared]. Sophistication is a generalization of the "H-function" or "minimal sufficient statistic" by Cover and Kolmogorov ⟨...⟩ The sophistication of an object in the size of that part of that object which describes its structure, i.e. the aggregate of its projectible properties.

One can also mention the formulation of "minimal description length" principle by Rissanen [30]; the abstract of this paper says: "Estimates of both integer-valued structure parameters and real-valued system parameters may be obtained from a model based on the shortest data description principle"; here "integer-valued structure parameters" may correspond to the choice of a statistical hypothesis (description set) while "real-valued system parameters" may correspond to the choice of a specific element in this set. The author then says that "by finding the model which minimizes the description length one obtains estimates of both the integer-valued structure parameters and the real-valued system parameters".

We do not try here to follow the development of these and similar ideas. Let us mention only that the traces of the same ideas (though even more vague) could be found in 1960s in the classical papers of Solomonoff [36, 37] who tried to use shortest descriptions for inductive inference (and, as a side product, gave the definition of complexity later rediscovered by Kolmogorov [13]). One may also mention a "minimum message length principle" that goes back to [48]; the idea of two-part description is explained in [48] as follows:

> If the things are now classified then the measurements can be recorded by listing the following:
>
> 1. The class to which each thing belongs.
> 2. The average properties of each class.

3. The deviations of each thing from the average properties of its parent class.

If the things are found to be concentrated in a small area of the region of each class in the measurement space then the deviations will be small, and with reference to the average class properties most of the information about a thing is given by naming the class to which it belongs. In this case the information may be recorded much more briefly than if a classification had not been used. We suggest that the best classification is that which results in the briefest recording of all the attribute information.

Here the "class to which thing belongs" corresponds to a set (statistical model, description in our terminology); the authors say that if this set is small, then only few bits need to be added to the description of this set to get a full description of the thing in question.

The main technical results of this sections (Theorems 1, 2, and 3) are taken from [42] (where some historical account is provided).

4 Bounded complexity lists

In this section it is more convenient to use plain complexity $C(x)$ instead of the prefix version $K(x)$. As we have mentioned, the difference between them is only logarithmic, and we mainly ignore terms of that size.

4.1 Enumerating strings of complexity at most m

Consider some integer m, and all strings x of (plain) complexity at least m. Let Ω_m be the number of those strings. The following properties of Ω_m are well known and often used (see, e.g., [8]).

Proposition 11.

- $\Omega_m = \Theta(2^m)$ (i.e., $c_1 2^m \le \Omega_m \le c_2 2^m$ for some positive constants c_1, c_2 and for all m;
- $C(\Omega_m) = m + O(1)$.

Proof. The number of strings of complexity at most m is bounded by the total number of programs of length at most m, which is $O(2^m)$. On the other hand, if Ω_m is an $(m-d)$-bit number, we can specify a string of complexity greater than m using $m - d + O(\log d)$ bits: first we specify d in a self-delimiting manner using $O(\log d)$ bits, and then append Ω_m in binary. This information allows us to reconstruct d, then m and Ω_m, then enumerate

strings of complexity at most m until we have Ω_m of them (so all strings of complexity at most m are enumerated), and then take the first string x_m that has not been enumerated. As $m < \mathrm{C}(x_m) \le m - d + O(\log d)$, the value of d is bounded by a constant and hence Ω_m is an $(m - O(1))$-bit number.

In this argument the binary representation of Ω_m can be replaced by its program, so $\mathrm{C}(\Omega_m) \ge m - O(1)$. The upper bound $m + O(1)$ is obvious, since $\Omega_m = O(2^m)$. □

Given m, we can enumerate all strings of complexity at most m. How many steps needs the enumeration algorithm to produce all of them? The answer is provided by the so-called *busy beaver numbers*; let us recall their definition in terms of Kolmogorov complexity (see [41, section 1.2.2] for details).

By definition, the number $B(m)$ is the maximal integer of complexity at most m. It is not hard to see that $\mathrm{C}(B(m)) = m + O(1)$. Indeed, $\mathrm{C}(B(m)) \le m$ by definition. On the other hand, the complexity of the next number $B(m) + 1$ is greater than m and at the same time is bounded by $\mathrm{C}(B(m)) + O(1)$.

Note that $B(m)$ can be undefined for small m (if there are no integers of complexity at most m) and that $B(m + 1) \ge B(m)$ for all m. For some m this inequality may not be strict. This happen, for example, if the optimal algorithm used to define Kolmogorov complexity is defined only on strings of, say, even lengths; this restriction does not prevent it from being optimal, but then $B(2n) = B(2n+1)$ for all n, since there are no objects of complexity exactly $2n + 1$. However, for some constant c we have $B(m + c) > B(m)$ for all m. Indeed, consider a program p of length at most m that prints $B(m)$. Transform it to a program p' that runs p and then adds 1 to the result. This program witnesses that $\mathrm{C}(B(m) + 1) \le m + c$ for some constant c. Hence $B(m + c) \ge B(m) + 1$.

Now we define $B'(m)$ as follows. As we have said, the set of all strings of complexity at most m can be enumerated given m. Fix some enumeration algorithm A (with input m) and some computation model. Then let $B'(m)$ be the number of steps used by this algorithm to enumerate all the strings of complexity at most m.

Proposition 12. The numbers $B(m)$ and $B'(m)$ coincide up to $O(1)$-change in m. More precisely, we have

$$B'(m) \le B(m + c), \qquad B(m) \le B'(m + c)$$

for some c and for all m.

Proof. To find $B'(m)$, it is enough to know m-bit binary string that represents Ω_m (this string also determines m). Therefore $\mathrm{C}(B'(m)) \le m + c$ for some constant c. As $B(m + c)$ is the largest number of complexity $m + c$ or less, we have $B'(m) \le B(m + c)$.

On the other hand, if some integer N exceeding both m and $B'(m)$ is given, we can run the enumeration algorithm A within N steps for each input smaller than N. Consider the first string that has not been enumerated. Its complexity is greater than m, so

$C(N) > m - c$ for some constant c. Thus the complexity of every number N starting from $\max\{m, B'(m)\}$ is greater than $m - c$, which means that $\max\{m, B'(m)\} > B(m - c)$. It remains to note that for all large enough m we have $m \leq B(m - c)$, as the complexity of m is $O(\log m)$. Thus for all large enough m the number $B'(m)$ (and not m) must be bigger than $B(m - c)$. Replacing here m by $m + c$ and increasing the constant c if needed, we conclude that $B'(m + c) > B(m)$ for all m. □

A similar argument shows that $B(n)$ coincides (up to $O(1)$-change in the argument) with the maximal computation time of the universal decompressor (from the definition of plain Kolmogorov complexity) on inputs of size at most m, see [41, section 1.2.2]

The next result says how many strings require long time to be enumerated.

Proposition 13. After $B'(m - s)$ steps of the enumeration algorithm on input m there are $2^{s+O(\log m)}$ strings that are not yet enumerated.

We assume that the algorithm enumerates strings (for every input m) without repetitions. Note also that here B' can be replaced by B, since they differ at most by a constant change in the argument.

Proof. To make the notation simpler we omit $O(1)$- and $O(\log m)$-terms in this argument. Given Ω_{m-s}, we can determine $B'(m-s)$. If we also know how many strings of complexity at most m appear after $B'(m - s)$ steps, we can wait until that many strings appear and then find a string of complexity greater than m. If the number of remaining strings is smaller than $2^{s-O(\log m)}$, we get a prohibitively short description of this high complexity string.

On the other hand, let x be the last element that has been enumerated in $B'(m - s)$ steps. If there are significantly more than 2^s elements after x, say, at least 2^{s+d} for some d, we can split the enumeration in portions of size 2^{s+d} and wait until the portion containing x appears. By assumption this portion is full. The number N of steps needed to finish this portion is at least $B'(m-s)$. This number N and its successor $N+1$ can be reconstructed from the portion number that contains about $m-s-d$ bits. Thus the complexity of $N+1$ is at most $m - s - d + O(\log m)$. Hence we have

$$B(m - s - d + O(\log m)) > N \geq B'(m - s).$$

By Proposition 12 we can replace B' by B here:

$$B(m - s - d + O(\log m)) > B(m - s).$$

(with some other constant in O-notation). Since B is a non-decreasing function, we get $d = O(\log m)$. □

4.2 Ω-like numbers

G. Chaitin introduced the "Chaitin Ω-number" $\Omega = \sum_k \mathbf{m}(k)$; it can also be defined as the probability of termination if the optimal prefix decompressor is applied to a random bit sequence (see [41, section 5.7]).[6] The numbers Ω_n are finite versions of Chaitin's Ω-number. The information contained in Ω_n increases as n increases; moreover, the following proposition is true. In this proposition we consider Ω_n as a bit string (of length $n + O(1)$) identifying the number Ω_n and its binary representation.

Proposition 14. Assume that $k \le m$. Consider the string $(\Omega_m)_k$ consisting of the first k bits of Ω_m. It is $O(\log m)$-equivalent to Ω_k: both conditional complexities $C(\Omega_k|(\Omega_m)_k)$ and $C((\Omega_m)_k|\Omega_k)$ are $O(\log m)$.

Proof. This is essentially the reformulation of the previous statement (Proposition 13).

Run the algorithm that enumerates strings of complexity at most m. Knowing $(\Omega_m)_k$, we can wait until less than 2^{m-k} strings are left in the enumeration of strings of complexity at most m; we know that this happens after more than $B(k)$ steps, and in this time we can enumerate all strings of complexity at most k and compute Ω_k. (In this argument we ignore $O(\log m)$-terms, as usual.)

Now the second inequality follows by the symmetry of information property. Indeed, since $C(\Omega_k) = k+O(1)$ and $C((\Omega_m)_k) \le k+O(1)$, the inequality $C(\Omega_k|(\Omega_m)_k) = O(\log m)$ implies the inequality $C((\Omega_m)_k|\Omega_k) = O(\log m)$.

A direct argument is also easy. Knowing Ω_k and k, we can find the list of all the strings of complexity at most k and the number $B'(k)$. Then we make $B'(k)$ steps in the enumeration of the list of strings of complexity at most m. Proposition 13 then guarantees that at that moment Ω_m is known with error about 2^{m-k}, so the first k bits of Ω_m can be reconstructed with small advice (of logarithmic size; we omit terms of that size in the argument). □

There is a more direct connection with Chaitin's Ω-number: one can show that the number Ω_m is $O(\log m)$-equivalent to the m-bit prefix of Chaitin's Ω-number. Since in this survey we restrict ourselves to finite objects, we do not go into details of the proof here, see [41, section 5.7.7].

4.3 Position in the list is well defined

We discussed how much time is needed to enumerate all strings of complexity at most m and how many strings remain not enumerated before this time. Now we want to study

[6] This number depends on the choice of the prefix decompressor, so it is not a specific number but a class of numbers. The elements of this class can be equivalently characterized as random lower semicomputable reals in $[0, 1]$, see [41, section 5.7].

which strings remain not enumerated.

More precisely, let x be some string of complexity at most m, so x appears in the enumeration of all strings of complexity at most m. How close x is to the end, that is, how many strings are enumerated after x? The answer depends on the enumeration, but only slightly, as the following proposition shows.

Proposition 15. Let A and B be algorithms that both for any given m enumerate (without repetitions) the set of strings of complexity at most m. Let x be some string and let a_x and b_x the number of strings that appear after x in A- and B-enumerations. Then $|\log a_x - \log b_x| = O(\log m)$.

We may also assume that A and B are algorithms without input of complexity $O(\log m)$ that enumerate strings of complexity at most m.

Proof. Assume that a_x is small: $\log a_x \le k$. Why $\log b_x$ cannot be much larger than k? Given the first $m - \log b_x$ bits of Ω_m and B, we can compute a finite set of strings B' that contains x and consists only of string of complexity at most m. Then we can wait until all strings from B' appear in A-enumeration. After then at most 2^k strings are left, and we need k bits to count them. In this way we can describe Ω_m by $m - \log b_x + k + O(\log m)$ bits; however, Proposition 11 says that $\mathrm{C}(\Omega_m) = m + O(1)$. Hence $\log b_x \le k + O(\log m)$.

The other inequality is proven by a symmetric argument. □

In this theorem A and B enumerate exactly the same strings (though in different order). However, the complexity function is essentially defined with $O(1)$-precision only: different optimal programming languages lead to different versions. Let C and $\tilde{\mathrm{C}}$ be two (plain) complexity functions; then $\tilde{\mathrm{C}}(x) \le \mathrm{C}(x) + c$ for some c and for all x. Then the list of all x with $\mathrm{C}(x) \le m$ is contained in the list of all x with $\tilde{\mathrm{C}}(x) \le m + c$. The same argument shows that the number of elements after x in the first list cannot be much larger than the number of elements after x in the second list. The reverse inequality is not guaranteed, however, even for the same version of complexity (small increase in the complexity bound may significantly increase the number of strings after x in the list). We will return to this question in Section 4.4, but let us note first that some increase is guaranteed.

Proposition 16. If for a string x there are at least 2^s elements after x in the enumeration of all strings of complexity at most m, then for every $d \ge 0$ there is at least $2^{s+d-O(\log m)}$ strings after x in the enumeration of all strings of complexity at most $m + d$.

Proof. Essentially the same argument works here: if there are much less than 2^{s+d} strings after x in the bigger list, then this bigger list can be determined by 2^{m-s} bits needed to cover x in the smaller list and less than $s + d$ bits needed to count the elements in the bigger list that follow the last covered element. □

The last proposition can be restated in the following way. Let us fix some complexity function and and some algorithm that, given m, enumerates all strings of complexity at most m. Then, for a given string x, consider the function that maps every $m \geq \mathrm{C}(x)$ to the logarithm of the number of strings after x in the enumeration with input m. Proposition 16 says that d-increase in the argument leads at least to $(d - O(\log m))$-increase of this function (but the latter increase could be much bigger). As we will see, this function is closely related to the set P_x (and therefore Q_x): it is one more representation of the same boundary curve.

4.4 The relation to P_x

To explain the relation, consider the following procedure for a given binary string x. For every $m \geq \mathrm{C}(x)$ draw the line $i + j = m$ on (i, j)-plane. Then draw the point on this line with second coordinate s where s is the logarithm of the number of elements after x in the enumeration of all strings of complexity at most m. Mark also all points on this line on the right of (=below) this point. Doing this for different m, we get a set (Figure 6). Proposition 16 guarantees that this set is upward closed with logarithmic precision: if

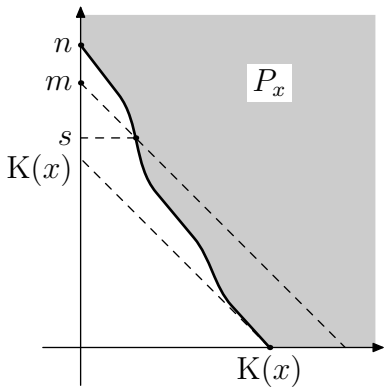

Figure 6: For each m between $\mathrm{K}(x)$ and n (length of x) we count elements after x in the list of strings having complexity at most m; assuming there is about 2^s of them, we draw point $(m - s, s)$ and get a point on some curve. This curve turns out to be the boundary of P_x (with logarithmic precision).

some point (i, j) belongs to this set, then the point $(i, j+d)$ is in $O(\log(i+j))$-neighborhood of this set. This implies that the point $(i + d, j)$ is also in the neighborhood, since our set is closed by construction in the direction $(1, -1)$.

It turns out that this set coincides with P_x (Definition 3) with $O(\log n)$-precision for a string x of length n (this means, as usual, that each of the two sets is contained in the $O(\log n)$-neighborhood of the other one):

Theorem 5. *Let x be a string of length n. If x has a $(i*j)$-description then x is at least $2^{j-O(\log n)}$-far from the end of $(i+j+O(\log n))$-list. Conversely, if there are at least 2^j elements that follow x in the $(i+j)$-list then x has a $((i+O(\log n))*j)$-description.*

Proof. We need to verify two things. First, assuming that x has a $(i*j)$-description, we need to show that it is at least 2^j-far from the end of $(i+j)$-list. (With error terms: in $(i+j+O(\log n))$-list there are at least $2^{j-O(\log n)}$ elements after x.) Indeed, knowing some $(i*j)$-description A for x, we can wait until all the elements of A appear in $(i+j)$-list (as usual, we omit $O(\log n)$-term: all elements of A have complexity at most $i+j+O(\log n)$, so we should consider $(i+j+O(\log n))$-list to be sure that is contains all elements of A). In particular, x has appeared at that moment. If there are (significantly) less than 2^j elements after x, then we can encode the number of remaining elements by (significantly) less than j bits, and together with the description of A we get less than $i+j$ bits to describe Ω_{i+j}, which is impossible.

Second, assume that there are at least 2^j elements that follow x in the $(i+j)$-list. Then, splitting this list into 2^j-portions, we get at most 2^i full portions, and x is covered by one of them. Each portion has complexity at most i and log-size at most j, so we get an $(i*j)$-description for x. (As usual, logarithmic terms are omitted.) □

Now we can reformulate the properties of stochastic and antistochastic objects. Every object of complexity k appears in the list of objects of complexity at most k' for all $k' > k$. Each stochastic object is far from the end of these lists (except, may be, for some k'-lists with k' very close to k). Each antistochastic object of length n is maximally close to the end of all k'-lists with $k' < n$ (there are about $2^{k'-k}$ objects after x), except, may be, for some k'-lists with k' very close to n. When k' becomes greater than n, then even antistochastic strings are far from the end of the k'-list. What we have said is just the description of the corresponding curves (Figure 2) using Theorem 5.

4.5 Standard descriptions

The lists of objects of bounded complexity provide a natural class of descriptions. Consider some m and the number Ω_m of strings of complexity at most m. This number can be represented in binary:

$$\Omega_m = 2^a + 2^b + \dots,$$

where $a > b > \ldots$. The list itself then can be split into pieces of size 2^a, 2^b,..., and these pieces can be considered as description of corresponding objects. In this way for each string x and for each $m \geq \mathrm{C}(x)$ we get some description on x, a piece than contains x. Descriptions obtained in this way will be called *standard* descriptions. Note that for a given x we have many standard descriptions (depending on the choice of m). One should have in mind also that the class of standard descriptions depends on the choice of the complexity function and the enumeration algorithm, and we assume in the sequel that they are fixed.

The following results show that standard descriptions are in a sense universal. First let us note that the standard descriptions have parameters close to the boundary curve of P_x (more precisely, to the boundary curve of the set constructed in the previous section that is close to P_x).[7]

Proposition 17. Consider the standard description A of size 2^j obtained from the list of all strings of complexity at most m. Then $\mathrm{C}(A) = m - j + O(\log m)$, and the number of elements in the list that follow the elements of A is $2^{j+O(\log m)}$.

This statement says that parameters of A are close to the point on the line $i + j = m$ considered in the previous section (Figure 6).

Proof. To specify A, it is enough to know the first $m - j$ bits of Ω_m (and m itself). The complexity of A cannot be much smaller, since knowing A and the j least significant bits of Ω_m we can reconstruct Ω_m.

The number of elements that follow A cannot exceed 2^j (it is a sum of smaller powers of 2); it cannot be significantly less since it determines Ω_m together with the first $m - j$ bits of Ω. (In other words, since Ω_m is an incompressible string of length m, it cannot have more that $O(\log m)$ zeros in a row.) □

This result does *not* imply that every point on the boundary of P_x is close to parameters of some standard description. If some part of the boundary has slope -1, we cannot guarantee that there are standard descriptions along this part. For example, consider the list of strings of complexity at most m; the maximal complexity of strings in this list is $m - c$ for some $c = O(1)$; if we take first string of this complexity, there are $2^{m+O(1)}$ strings after it, so the corresponding point is close to the vertical axis, and due to Proposition 16

[7] In general, if two sets X and Y in $\mathbb{N}^2$ are close to each other (each is contained in the small neighborhood of the other one), this does not imply that their boundaries are close. It may happen that one set has a small "hole" and the other does not, so the boundary of the first set has points that are far from the boundary of the second one. However, in our case both sets are closed by construction in two different directions, and it implies that the boundaries are also close.

all other standard descriptions of x are also close to the vertical axis. However, descriptions with parameters close to arbitrary points on the boundary of P_x can be obtained from standard descriptions by chopping them into smaller parts, as in Proposition 8. In that shopping it is natural to use the order in which the strings were enumerated. In other words, chop the list of strings of complexity at most m into portions of size 2^j. Consider all the full portions (of size exactly 2^j) obtained in this way (they are parts of standard descriptions of bigger size). Descriptions obtained in this way are "universal" in the following sense: if a pair (i, j) is on the boundary of P_x then there is a set $A \ni x$ of this type of complexity $i + O(\log(i + j))$ and log-cardinality $j + O(\log(i + j))$.

The following result says more: for every description A for x there is a "better" standard description that is simple given A (note that $d \geq 0$ in the following proposition and that optimality deficiency of B does not exceed that of A up to logarithmic term).

Proposition 18. Let A be an $(i * j)$-description of a string x of length n. Then there exists a standard description B that has parameters $\mathrm{C}(B) \leq i - d + O(\log n)$ and $\log \#B \leq j + d + O(\log n)$ for some $d \geq 0$, and is simple given A, i.e., $\mathrm{C}(B|A) = O(\log n)$.

Proof. If A has strings of length different from n, remove all those strings. In this way A becomes $(i * j)$-description for x with slightly larger i than before the removal and the same or smaller j. Now all the elements of A have complexity at most $m = i + j + O(\log j) = i + j + O(\log n)$, where the latter inequality holds, as after removal we have $j \leq n$. Consider the list of all strings of complexity at most m and the standard description B of x obtained from this list. As we know from Proposition 17, the sum of the parameters of this description is close to m (and therefore to $i + j$). We need to show that the size of B is large, at least $2^{j-O(\log n)}$ (recall that d in the statement should be positive). Why is this the case? Consider elements that appear after the last element of A in the list. There are at least $2^{j-O(\log n)}$ of them, otherwise the total number of elements in the list could be described in much less than m bits (that number can be specified by m, A and the number of elements after the last element of A). Therefore there are at least $2^{j-O(\log n)}$ elements in the list that appear after x, so B cannot be small.

Why B is simple given A? Denote the size of B by $2^{j'}$. Given A and m, we can find the last element of A, call it x', in the list of strings of complexity at most m. Chop the list into portions of size $2^{j'}$. Then B is the last complete portion. If B contains x', we can find B from m, j', and x' as the complete portion containing x'. Otherwise, x' appears in the list after all the elements from B. In this case we can find B from m and x' as the last complete portion before x'. Thus in any case we are able to find B from m, j', and x' plus one extra bit. □

For the same reason every standard description B of some x is simple given x (and

this is not a surprise, since we know that all optimal descriptions of x are simple given x, see Proposition 9).

Proposition 18 has the following corollary which we formulate in an informal way. Let A be some $(i*j)$-description with parameters on the boundary of P_x. Assume that on the left of this point the boundary curve decreases fast (with slope less than -1). Then in Proposition 18 the value of d is small, otherwise the point $(i-d, j+d)$ would be far from P_x. So the complexities of A and the standard description B are close to each other. We know also that A is simple given B, therefore B is also simple given A, and A and B have the same information (have small conditional complexities in both directions).

If we have two different descriptions A, A' with approximately the same parameters on the boundary of P_x, and the curve decreases fast on the left of the corresponding boundary point, the same argument shows that A and A' have the same information. Note that the condition about the slope is important: if the point is on the segment with slope -1, the situation changes. For example, consider a random n-bit string x and two its descriptions. The first one consists of all n-bit strings that have the same left half as x, the second one consists of all n-bit strings that have the same right half. Both have the same parameters: complexity $n/2$ and log-size $n/2$, so they both correspond to the same point on the boundary of P_x. Still the information in these two descriptions is different (left and right halves of a random string are independent).

These results sound as good news. Let us recall our original goal: to formalize what is a good statistical model. It seems that we are making some progress. Indeed, for a given x we consider the boundary curve P_x and look at the place when it first touches the lower bound $i+j = \mathrm{C}(x)$; after that it stays near this bound. In other terms, we consider models with negligible optimality deficiency, and select among them the model with minimal complexity. Giving a formal definitions, we need to fix some threshold ε. Then we say that a set A is a *ε-sufficient statistic* if $\delta(x, A) < \varepsilon$, and may choose the simplest one among them and call it the *minimal ε-sufficient statistic*. If the curve goes down fast on the left of this point, we see that all the descriptions with parameters corresponding to minimal sufficient statistic are equivalent to each other.

Trying to relate these notion to practice, we may consider the following example. Imagine that we have digitized some very old recording and got some bit string x. There is a lot of dust and scratches on the recording, so the originally recorded signal is distorted by some random noise. Then our string x has a two-part description: the first part specifies the original recording and the noise parameters (intensity, spectrum, etc.) and the second part specifies the noise exactly. May be, the first part is the minimal sufficient statistic — and therefore sound restoration (and lossy compression in general) is a special case of the problem of finding a minimal sufficient statistic? The uniqueness result above (saying that all the minimal sufficient statistics contain the same information under some

conditions) seem to support this view: different good models for the same object contain the same explanation.

Still the following observation (that easily follows from what we know) destroys this impression completely.

Proposition 19. Let B be some standard description of complexity i obtained from the list of all strings of complexity at most m. Then B is $O(\log m)$-equivalent to Ω_i.

This looks like a failure. Imagine that we wanted to understand the nature of some data string x; finally we succeed and find a description for x of reasonable complexity and negligible randomness and optimality deficiencies (and all the good properties we dreamed of). But Proposition 19 says that the information contained in this description is more related to the computability theory than to specific properties of x. Recalling the construction, we see that the corresponding standard description is determined by some prefix of some Ω-number, and is an interval in the enumeration of objects of bounded complexity. So if we start with two old recordings, we may get the same information, which is not what we expect from a restoration procedure.

What could we do with this? First, we could just relax and be satisfied that we now understand much better the situation with possible descriptions for x. We know that every x is characterized by some curve that has several equivalent definitions (in terms of stochasticity, randomness deficiency, position in the enumeration — as well as time-bounded complexity, see Section 5 below). We know that standard descriptions cover the parts of the curve where it goes down fast, and to cover the parts where the slope is -1 one may use standard descriptions and their pieces; all these descriptions are simple given x. When curve goes down fast, the description is essentially unique (all the descriptions with the same parameters contain the same information, equivalent to the corresponding Ω-number); this is not true on parts with slope -1. So, even if this curve is of no philosophical importance, we have a lot of technical information about possible models.

The other approach is to go farther and consider only models from some class (Section 6), or add some additional conditions and look for "strong models" (Section 7).

4.6 Non-stochastic objects revisited

Now we can explain in a different way why the probability of obtaining a non-stochastic object in a random process is negligible (Proposition 5). This explanation uses the notion of mutual information from algorithmic information theory. The mutual information in two strings x and y is defined as

$$I(x : y) = \mathrm{C}(x) - \mathrm{C}(x|y) = \mathrm{C}(y) - \mathrm{C}(y|x) = \mathrm{C}(x) + \mathrm{C}(y) - \mathrm{C}(x, y);$$

all three expressions are $O(\log n)$-close if x and y are strings of length n (see, e.g., [41, Chapter 2]).

Consider an arbitrary string x of length n; let k be the complexity of x. Consider the list of all objects of complexity at most k, and the standard description A for x obtained from this list. If A is large, then x is stochastic; if A is small, then x contains a lot of information about Ω_k and Ω_n.

More precisely, let us assume that A has size 2^{k-s} (i.e., is 2^s times smaller than it could be). Then (recall Proposition 17) the complexity of A is $s + O(\log k)$, since we can construct A knowing k and the first s bits of Ω_k (before the bit that corresponds to A). So we get $(s + O(\log k)) * (k - s)$-description with optimality deficiency $O(\log k)$.

On the other hand, knowing x and k, we can find the ordinal number of x in the enumeration, so we know Ω_k with error at most 2^{k-s}, so $\mathrm{C}(\Omega_k|x) \le k - s + O(\log k)$, and $I(x : \Omega_k) \ge s - O(\log k)$ (recall that $\mathrm{C}(\Omega_k) = k + O(1)$). In the last statement we may replace Ω_k by Ω_n (where n is the length of x): we know from Proposition 14 that Ω_k is simple given Ω_n, so if condition Ω_k decreases complexity of x by almost s bits, the same is true for condition Ω_n.

Comparing arbitrary $i \le n$ with this s (it can be larger than s or smaller than s), we get the following result:

Proposition 20. Let x be a string of length n. For every $i \le n$

- either x is $(i + O(\log n), O(\log n))$-stochastic,
- or $I(x : \Omega_n) \ge i - O(\log n)$.

Now we may use the following (simple and general) observation: for every string u the probability to generate (by a randomized algorithm) an object that contains a lot of information about u is negligible:

Proposition 21. For every string u and for every number d, we have

$$\sum\{\mathbf{m}(x) \mid \mathrm{K}(x) - \mathrm{K}(x|u) \ge d\} \le 2^{-d}.$$

In this proposition the sum is taken over all strings x that have the given property (have a large mutual information with u). Note that we have chosen the representation of mutual information that makes the proposition easy (in particular, we have used prefix complexity). As we mentioned, other definitions differ only by $O(\log n)$ if we consider strings x and u of length at most n, and logarithmic accuracy is enough for our purposes.

Proof. Recall the definition of prefix complexity: $\mathrm{K}(x) = -\log \mathbf{m}(x)$, and $\mathrm{K}(x|u) = -\log \mathbf{m}(x|u)$. So $\mathrm{K}(x) - \mathrm{K}(x|u) \ge d$ implies $\mathbf{m}(x) \le 2^{-d}\, \mathbf{m}(x|u)$, and it remains to note that $\sum_x \mathbf{m}(x|u) \le 1$ for every u. □

Propositions 20 and 21 immediately imply the following improved version of Proposition 5 (page 12):

Proposition 22.

$$\sum\{\,\mathbf{m}(x) \mid x \text{ is a } n\text{-bit string that is not } (\alpha, O(\log n))\text{-stochastic }\} \le 2^{-\alpha+O(\log n)}$$

for every α.

The improvement here is the better upper bound for the randomness deficiency: $O(\log n)$ instead of $\alpha + O(\log n)$.

4.7 Historical comments

The relation between busy beaver numbers and Kolmogorov complexity was pointed out in [11] (see Section 2.1). The enumerations of all objects of bounded complexity and their relation to stochasticity were studied in [12] (see Section III, E).

5 Computational and logical depth

In this section we reformulate the results of the previous one in terms of bounded-time Kolmogorov complexity.

5.1 Bounded-time Kolmogorov complexity

The usual definition of Kolmogorov complexity of x as the minimal length $l(p)$ of a program p that produces x does not take into account the running time of the program p: it may happen that the minimal program for x requires a lot of time to produce x while other programs produce x faster but are longer (for example, program "print x" is rather fast). To analyze this trade-off, the following definition is used.

Definition 4. Let D be some algorithm; its input and output are binary strings. For a string x and integer t, define

$$\mathrm{C}^t_D = \min\{l(p)\colon D \text{ produces } x \text{ on input } p \text{ in at most } t \text{ steps}\},$$

the time-bounded Kolmogorov complexity of x with time bound t with respect to D.

This definition was mentioned already in the first paper by Kolmogorov [13]:

> Our approach has one important drawback: it does not take into account the efforts needed to transform the program p and object x [the description and the condition] to the object y [whose complexity is defined]. With appropriate definitions, one may prove mathematical results that could be interpreted as the existence of an object x that has simple programs (has very small complexity $K(x)$) but all short programs that produce x require an unrealistically long computation. In another paper I plan to study the dependence of the program complexity $K^t(x)$ on the difficulty t of its transformation into x. Then the complexity $K(x)$ (as defined earlier) reappears as the minimum value of $K^t(x)$ if we remove restrictions on t.

Kolmogorov never published a paper he speaks about, and this definition is less studied than the definition without time bounds, for several reasons.

First, the definition is machine-dependent: we need to decide what computation model is used to count the number of steps. For example, we may consider one-tape Turing machines, or multi-tape Turing machine, or some other computational model. The computation time depends on this choice, though not drastically (e.g., a multi-tape machine can be replaces with a one-tape machine with quadratic increase in time, and most popular models are polynomially related — this observation is used when we argue that the class P of polynomial-time computable functions is well defined).

Second, the basic result that makes the Kolmogorov complexity theory possible is the Solomonoff–Kolmogorov theorem saying that there exists an optimal algorithm D that makes the complexity function minimal up to $O(1)$ additive term. Now we need to take into account the time bound, and get the following (not so nice) result.

Proposition 23. There exists an optimal algorithm D for time-bounded complexity in the following sense: for every other algorithm D' there exists a constant c and a polynomial q such that

$$\mathrm{C}^t_{D'}(x) \le \mathrm{C}^{q(t)}_D(x) + c$$

for all strings x and integers t.

In this result, by "algorithm" we may mean a k-tape Turing machine, where k is an arbitrary fixed number. However, the claim remains true even when k is not fixed, i.e., we may allow D' to have more tapes than D has.

The proof remains essentially the same: we choose some simple self-delimiting encoding of binary strings $p \mapsto \hat{p}$ and some universal algorithm $U(\cdot,\cdot)$ and then let

$$D(\hat{p}x) = U(p,x)$$

Then the proof follows the standard scheme; the only thing we need to note is that the decoding of $\hat{p}$ runs in polynomial time (which is true for most natural ways of self-delimiting encoding) and that the universal algorithm simulation overhead is polynomial (which is also true for most natural constructions of universal algorithms).

A similar result is true for conditional decompressors, so the conditional time-bounded complexity can be defined as well.

For Turing machines with fixed number of tapes the statement is true for some linear polynomial $q(n) = O(n)$. For the proof we need to consider a universal machine U that simulates other machines efficiently: it should move the program along the tape, so the overhead is bounded by a factor that depends on the size of the program and not on the size of the input or computation time.[8]

Let $t(n)$ be an arbitrary total computable function with integer arguments and values; then the function

$$x \mapsto \mathrm{C}_D^{t(l(x))}(x)$$

is a computable upper bound for the complexity $\mathrm{C}(x)$ (defined with the same D). Replacing the function $t(\cdot)$ by a bigger function, we get a smaller computable upper bound. An easy observation: in this way we can match every computable upper bound for Kolmogorov complexity.

Proposition 24. Let $\tilde{\mathrm{C}}(x)$ be some total computable upper bound for Kolmogorov complexity function based on the optimal algorithm D from Proposition 23. Then there exists a computable function t such that $\mathrm{C}_D^{t(l(x))}(x) \leq \tilde{\mathrm{C}}(x)$ for every x.

Proof. Given a number n, we wait until every string x of length at most n gets a program that has complexity at most $\tilde{\mathrm{C}}(x)$, and let $t(n)$ be the maximal number of steps used by these programs. □

So the choice of a computable time bound is essentially equivalent to the choice of a computable total upper bound for Kolmogorov complexity.

In the sequel we assume that some optimal (in the sense of Proposition 23) D is fixed and omit the subscript D in $\mathrm{C}_D^t(\cdot)$. Similar notation $\mathrm{C}^t(\cdot|\cdot)$ is used for conditional time-bounded complexity.

[8] This observation motivates Levin's version of complexity (Kt, see [20, Section 1.3, p. 21]) where the program size and logarithm of the computation time are added: linear overhead in computation time matches the constant overhead in the program size. However, this is a different approach and we do not use the Levin's notion of time bounded complexity in this survey.

5.2 Trade-off between time and complexity

We use the extremely fast growing sequence $B(0), B(1), \ldots$ as a scale for measuring time. This sequence grows faster than any computable function (since the complexity of $t(n)$ for any computable t is at most $\log n + O(1)$, we have $B(\log n + O(1)) \geq t(n)$). In this scale it does not matter whether we use time or space as the resource measure: they differ at most by an exponential function, and $2^{B(n)} \leq B(n + O(1))$ (in general, $f(B(n)) \leq B(n + O(1))$ for every computable f). So we are in the realm of general computability theory even if we technically speak about computational complexity, and the problems related to the unsolved P=NP question disappear.

Let x be a string of length n and complexity k. Consider the time-bounded complexity $\mathrm{C}^t(x)$ as a function of t. (The optimal algorithm from Proposition 23 is fixed, so we do not mention it in the notation.) It is a decreasing function of t. For small values of t the complexity $\mathrm{C}^t(x)$ is bounded by $n + O(1)$ where n stands for the length of x. Indeed, the program that prints x has size $n + O(1)$ and works rather fast. Formally speaking, $\mathrm{C}^t(x) \leq n + O(1)$ for $t = B(O(\log n))$. As t increases, the value of $\mathrm{C}^t(x)$ decreases and reaches $k = \mathrm{C}(x)$ as $t \to \infty$. It is guaranteed to happen for $t = B(k + O(1))$, since the computation time for the shortest program for x is determined by this program.

We can draw a curve that reflects this trade-off using B-scale for the time axis. Namely, consider the graph of the function

$$i \mapsto \mathrm{C}^{B(i)}(x) - \mathrm{C}(x)$$

and the set of points above this graph, i.e., the set

$$D_x = \{(i, j) \mid \mathrm{C}^{B(i)}(x) - \mathrm{C}(x) \leq j\}.$$

Theorem 6 ([2]). *The set D_x coincides with the set Q_x with $O(\log n)$-precision for a string x of length n.*

Recall that the set Q_x consists of pairs (α, β) such that x is (α, β)-stochastic (see p. 23).

Proof. As we know from Theorem 4, the sets P_x and Q_x are related by an affine transformation (see Figure 4). Taking this transformation into account, we need to prove two statements:

- if there exists an $(i * j)$-description A for x, then

$$\mathrm{C}^{B(i+O(\log n))}(x) \leq i + j + O(\log n);$$

- if $\mathrm{C}^{B(i)}(x) \leq i + j$, then

there exist an $((i + O(\log n)) * (j + O(\log n)))$-description for x.

Both statements are easy to prove using the tools from the previous section. Indeed, assume that x has an $(i * j)$-description A. All elements of A have complexity at most $i + j + O(\log n)$. Knowing A and this complexity, we can find the minimal t such that $C^t(x') \le i + j + O(\log n)$ for all x' from A. This t can be computed from A, which has complexity i, and an $O(\log n)$-bit advice (the value of complexity). Hence $t \le B(i + O(\log n))$ and $C^t(x) \le i + j + O(\log n)$, as required.

The converse: assume that $C^{B(i)}(x) \le i + j$. Consider all the strings x' that satisfy this inequality. There are at most $O(2^{i+j})$ such strings. Thus we only need to show that given i and j we are able to enumerate all those strings in at most $O(2^i)$ portions.

One can get a list of all those strings x' if $B(i)$ is given, but we cannot compute $B(i)$ given i. Recall that $B(i)$ is the maximal integer that has complexity at most i; new candidates for $B(i)$ may appear at most 2^i times. The candidates increase with time; when this happens, we get a new portion of strings that satisfy the inequality $C^{B(i)}(x) \le i + j$. So we have at most $O(2^{i+j})$ objects including x that are enumerated in at most 2^i portions, and this implies that x has an $((i + O(\log n)) * j)$-description. Indeed, we make all portions of size at most 2^j by splitting larger portions into pieces. The number of portions increases at most by $O(2^i)$, so it remains $O(2^i)$. Each portion (including the one that contains x) has then complexity at most $i + O(\log n)$ since it can be computed with logarithmic advice from its ordinal number. □

This theorem shows that the results about the existence of non-stochastic objects can be considered as the "mathematical results that could be interpreted as the existence of an object x that has simple programs (has very small complexity $K(x)$) but all short programs that produce x require an unrealistically long computation" mentioned by Kolmogorov (see the quotation above), and the algorithmic statistics can be interpreted as an implementation of Kolmogorov's plan "to study the dependence of the program complexity $K^t(x)$ on the difficulty t of its transformation into x", at least for the simple case of (unrealistically) large values of t.

5.3 Historical comments

Section 5 has title "logical and computational depth" but we have not mentioned these notions yet. The name "logical depth" was introduced by C. Bennett in [7]. He explains the motivation as follows:

> Some mathematical and natural objects (a random sequence, a sequence of zeros, a perfect crystal, a gas) are intuitively trivial, while others (e.g., the human body, the digits of π) contain internal evidence of a nontrivial causal history. ⟨...⟩

> We propose depth as a formal measure of value. From the earliest days of information theory it has been appreciated that information per se is not a good measure of message value. For example, a typical sequence of coin tosses has high information content but little value; an ephemeris, giving the positions of the moon and the planets every day for a hundred years, has no more information than the equations of motion and initial conditions from which it was calculated, but saves its owner the effort of recalculating these positions. The value of a message thus appears to reside not in its information (its absolutely unpredictable parts), nor in its obvious redundancy (verbatim repetitions, unequal digit frequencies), but rather is what might be called its buried redundancy — parts predictable only with difficulty, things the receiver could in principle have figured out without being told, but only at considerable cost in money, time, or computation. In other words, the value of a message is the amount of mathematical or other work plausibly done by its originator, which its receiver is saved from having to repeat.

Trying to formalize this intuition, Bennett suggests the following possible definitions:

> **Tentative Definition 0.1**: A string's depth might be defined as the execution time of its minimal program.

This notion is not robust (it depends on the specific choice of the optimal machine used in the definition of complexity). So Bennett considers another version:

> **Tentative Definition 0.2**: A string's depth at significance level s [might] be defined as the time required to compute the string by a program no more than s bits larger than the minimal program.

We see that Definition 0.2 consider the same trade-off as in Theorem 6, but in reversed coordinates (time as a function of difference between time-bounded and limit complexities). Bennett is still not satisfied by this definition, for the following reason:

> This proposed definition solves the stability problem, but is unsatisfactory in the way it treats multiple programs of the same length. Intuitively, 2^k distinct $(n+k)$-bit programs that compute same output ought to be accorded the same weight as one n-bit program $\langle ... \rangle$

In other language, he suggests to consider a priori probability instead of complexity:

> **Tentative Definition 0.3**: A string's depth at significance level s might be defined as the time t required for the string's time-bounded algorithmic probability $P_t(x)$ to rise to within a factor 2^{-s} of its asymptotic time-unbounded value $P(x)$.

Here $P_t(x)$ is understood as a total weight of all self-delimiting programs that produce x in time at most t (each program of length s has weight 2^{-s}). For our case (when we consider busy beaver numbers as time scale) the exponential time increase needed to switch from a priori probability to prefix complexity does not matter. Still Bennett is interested in more reasonable time bounds (recall that in his informal explanation a polynomially computable sequence of π-digits was an example of a deep sequence!), and prefers a priori probability approach. Moreover, he finds a nice reformulation of this definition (almost equivalent one) in terms of complexity:

> Although Definition 0.3 satisfactorily captures the informal notion of depth, we propose a slightly stronger definition for the technical reason that it appears to yield a stronger slow growth property ⟨...⟩
>
> **Definition 1** (Depth of Finite Strings): Let x and w be strings [probably w is a typo: it is not mentioned later] and s a significance parameter. A string's *depth* at significance level s, denoted $D_s(x)$, will be defined as
>
> $$\min\{T(p)\colon (|p| - |p^*| < s) \wedge (U(p) = x)\},$$
>
> the least time required to compute it by a s-incompressible program.

Here p^* is a shortest self-delimiting program for p, so its length $|p^*|$ equals $\mathrm{K}(p)$.

Actually, this *Definition 1* has a different underlying intuition than all the previous ones: a string x is deep if *all programs that compute x in a reasonable time, are compressible*. Note the before we required a different thing: that all programs that compute x in a reasonable time are much longer than the minimal one. This is a weaker requirement: one may imagine a long incompressible program that computes x fast. This intuition is explained in the abstract of the paper [7] as follows:

> [We define] an object's "logical depth" as the time required by a standard universal Turing machine to generate it from an input that is algorithmically random.

Bennett then proves a statement (called Lemma 3 in his paper) that shows that his *Definition 1* is almost equivalent to *Tentative Definition 0.3*: the time remains exactly the same, while s changes at most logarithmically (in fact, at most by $\mathrm{K}(s)$). So if we use Bennett's notion of depth (any of them, except for the first one mentioned) with busy beaver time scale, we get the same curve as in our definition.

A natural question arises: is there a direct proof that the output of an incompressible program with not too large running time is stochastic? In fact, yes, and one can prove

a more general statement: the output of a *stochastic* program with reasonable running time is stochastic (see Section 5.4); note that stochasticity is a weaker condition than incompressibility.

Let us mention also the notion of *computational depth* introduced in [4]. There are several versions mentioned in this paper; the first one exchanges coordinates in the Bennett's tentative definition 0.2 (reproduced in [4] as Definition 2.5). The authors write: "The first notion of computational depth we propose is the difference between a time-bounded Kolmogorov complexity and traditional Kolmogorov complexity" (Definition 3.1, where time bound is some function of input length). The other notions of computation depth are more subtle (they use distinguishing complexity or Levin complexity involving the logarithm of the computation time).

The connections between computational/logical depth and sophistication were anticipated for a long time; for example, Koppel writes in [18]:

> ⟨...⟩ The "dynamic" approach to the formalization of meaningful complexity is "depth" defined and discussed by Bennett [1]. [Reference to an unpublished paper "On the logical 'depth' of sequences and their reducibilities to incompressible sequences".] The depth of an object is the running-time of its most concise description. Since it is reasonable to assume that an object has been generated by its most concise description, the depth of an object can be thought of as a measure of its evolvedness.
>
> Although sophistication is measured in integers [not clear what in meant here: sophistication of S is also a function $c \mapsto SOPH_c(S)$] and depth is measured in functions, it is not difficult to translate to a common range.

Strangely, the direct connection between the most basic versions of these notions (Theorem 6) seems to be noticed only recently in [2].

5.4 Why so many equivalent definitions?

We have shown several equivalent (with logarithmic precision and up to affine transformation) ways to defined the same curve:

- (α, β)-stochasticity (Section 2);
- two-part descriptions and optimality deficiency, the set P_x (Section 3);
- position in the enumeration of objects of bounded complexity (Section 4);
- logical/computational depth (resource-bounded complexity, Section 5).

One can add to this list a characterization in terms of split enumeration (Section 3.4): the existence of $(i*j)$-description for x is equivalent (with logarithmic precision) to the existence of a simple enumeration of at most 2^{i+j} objects in at most 2^i portions (see Remark 7, p. 21, and the discussion before it).

Why do we need so many equivalent definitions of the same curve? First, this shows that this curve is really fundamental — almost as fundamental characterization of an object x as its complexity. As Koppel writes in [17], speaking about (some versions of) sophistication and depth:

> One way of demonstrating the naturalness of a concept is by proving the equivalence of a variety of prime facie different formalizations $\langle\ldots\rangle$. It is hoped that the proof of the equivalence of two approaches to meaningful complexity, one using static resources (program size) and the other using dynamic resources (time), will demonstrate not only the naturalness of the concept but also the correctness of the specifications used in each formalization to ensure robustness and generality.

Another, more technical reason: different results about stochasticity use different equivalent definitions, and a statement that looks quite mysterious for one of them may become almost obvious for another. Let us give two examples of this type (the first one is stochasticity conservation when random noise is added, the second one is a direct proof of Bennett's characterization mentioned above).

Proposition 25. Let x be some binary string, and let y be another string ("noise") that is conditionally random with respect to x, i.e., $C(y|x) \approx l(y)$. Then the pair (x,y) has the same stochasticity profile as x: the sets Q_x and $Q_{(x,y)}$ are logarithmically close to each other.

Before giving a proof sketch, let us mention that an interesting special case of this proposition is obtained if we consider a string u and its description X with small randomness deficiency: $d(u|X) \approx 0$. Let y be the ordinal number of u in X. Then the small randomness deficiency guarantees that y is conditionally random with respect to x. Then the pair (X,y) has the same stochasticity profile as X. Since this pair is mapped to u by a simple total computable function, we conclude (Proposition 3) that the stochasticity profile of X is contained in the stochasticity profile of u (more precisely, in its $O(\log n + d(u|X))$-neighborhood). (More simple and direct proof of this statement goes as follows: if U is a description for X that has small complexity and optimality deficiency, we can take the union of all elements of U that have approximately the same cardinality as X; one can verify easily that this union also has small complexity and optimality deficiency as a description for u.)

The full statement of Proposition 25 would introduce some bound for the difference $l(y) - \mathrm{C}(y|x)$ that is allowed to appear in the final estimate for the distance between sets. Recall also that we can speak about profiles of arbitrary finite objects, in particular, pairs of strings, using some natural encoding (Section 2.3).

Proof sketch. Using the depth characterization of stochasticity profile, we need to show that

$$\mathrm{C}^{B(i)}(x,y) - \mathrm{C}(x,y) \approx \mathrm{C}^{B(i)}(x) - \mathrm{C}(x).$$

Here "approximately" means that these two quantities may differ by a logarithmic term, and also we are allowed to add logarithmic terms to i (see below what does it mean). The natural idea is to rewrite this equality as

$$\mathrm{C}^{B(i)}(x,y) - \mathrm{C}^{B(i)}(x) \approx \mathrm{C}(x,y) - \mathrm{C}(x).$$

The right hand side is equal to $\mathrm{C}(y|x)$ (with logarithmic precision) due to Kolmogorov–Levin formula for the complexity of a pair (see, e.g., [41, Chapter 2]), and $\mathrm{C}(y|x)$ equals $l(y)$, as y is random and independent of x. Thus it suffices to show that the left hand side also equals $l(y)$. To this end we can prove a version of Kolmogorov–Levin formula for bounded complexity and show that the left hand side equals to $\mathrm{C}^{B(i)}(y|x)$. Again, since y is random and independent of x, $\mathrm{C}^{BB(i)}(y|x)$ equals $l(y)$.

This plan needs clarification. First of all, let us explain which version of Kolmogorov–Levin formula for bounded complexity we need. (Essentially it was published by Longpré in [22] though the statement was obscured by considering time bound as a function of the input length.)

The equality $\mathrm{C}(x,y) = \mathrm{C}(x) + \mathrm{C}(y|x)$ should be considered as two inequalities, and each one should be treated separately.

Lemma. 1. There exist some constant c and some polynomial $p(\cdot,\cdot)$ such that

$$\mathrm{C}^{p(n,t)}(x,y) \le \mathrm{C}^{t}(x) + \mathrm{C}^{t}(y|x) + c\log n$$

for all n and t and for all strings x and y of length at most n.

2. There exist some constant c and some polynomial $p(\cdot,\cdot)$ such that

$$\mathrm{C}^{p(2^n,t)}(x) + \mathrm{C}^{p(2^n,t)}(y|x) \le \mathrm{C}^{t}(x,y) + c\log n$$

for all n and t and for all strings x and y of length at most n.

Proof of the lemma. The proof of this time-bounded version is obtained by a straightforward analysis of the time requirements in the standard proof. The first part says that if

there is some program p that produces x in time t, and some program q that produces y from x in time t, then the pair (p,q) can be considered as a program that produces (x,y) in time $\mathrm{poly}(t,n)$ and has length $l(p)+l(q)+O(\log n)$ (we may assume without loss of generality that p and q have length $O(n)$, otherwise we replace them by shorter fast programs).

The other direction is more complicated. Assume that $\mathrm{C}^t(x,y)=m$. We have to count for a given x the number of strings y' such that $\mathrm{C}^t(x,y')\le m$. These strings (y is one of them) can be enumerated in time $\mathrm{poly}(2^n,t)$, so if there are 2^s of them, then $\mathrm{C}^{\mathrm{poly}(2^n,t)}(y|x)\le s+O(\log n)$ (the program witnessing this inequality is the ordinal number of y in the enumeration plus $O(\log n)$ bits of auxiliary information. Note that we do not need to specify t in advance, we enumerate y' in order of increasing time, and y is among first 2^s enumerated strings.

On the other hand, there are at most $2^{m-s+O(1)}$ strings x' for which this number (of different y' such that $\mathrm{C}^t(x',y')\le m$) is at least 2^{s-1}, and these strings also could be enumerated in time $\mathrm{poly}(2^n,t)$, so $\mathrm{C}^{\mathrm{poly}(2^n,t)}(x)\le m-s+O(\log n)$ (again we do not need to specify t, we just increase gradually the time bound). When these two inequalities are added, s disappears and we get the desired inequality. □

Of course, the exponent in the lemma is disappointing (for space bound it is not needed, by the way), but since we measure time in busy beaver units, it is not a problem for us: indeed, $\mathrm{poly}(2^n,B(i))\le B(i+O(\log n))$, and we allow logarithmic change in the argument anyway.

Now we should apply this lemma, but first we need to give a full statement of what we want to prove. There are two parts (as in the lemma):

- for every i there exists $j\le i+O(\log n)$ such that

$$\mathrm{C}^{B(j)}(x,y)-\mathrm{C}(x,y)\le \mathrm{C}^{B(i)}(x)-\mathrm{C}(x)+\varepsilon+O(\log n)$$

  for all strings x and y of length at most n such that $\mathrm{C}(y|x)\le l(y)-\varepsilon$;

- for every i there exists $j\le i+O(\log n)$ such that

$$\mathrm{C}^{B(j)}(x)-\mathrm{C}(x)\le \mathrm{C}^{B(i)}(x,y)-\mathrm{C}(x,y)+O(\log n)$$

  for all strings x and y of length at most n;

Both statements easily follow from the lemma. Let us start with the second statement where the hard direction of the lemma is used. As planned, we rewrite the inequality as

$$\mathrm{C}^{B(j)}(x)+\mathrm{C}(y|x)\le \mathrm{C}^{B(i)}(x,y)+O(\log n)$$

using the unbounded formula. Our lemma guarantees that

$$\mathrm{C}^{B(j)}(x) + \mathrm{C}^{B(j)}(y|x) \le \mathrm{C}^{B(i)}(x,y) + O(\log n)$$

for some $j \le i + O(\log n)$, and it remains to note that $\mathrm{C}(y|x) \le \mathrm{C}^{B(j)}(y|x)$. For the other direction the argument is similar: we rewrite the inequality as

$$\mathrm{C}^{B(j)}(x,y) \le \mathrm{C}(y|x) + \mathrm{C}^{B(i)}(x) + O(\log n)$$

and note that $\mathrm{C}(y|x) \ge l(y) - \varepsilon \ge C^{B(i)}(y|x) - \varepsilon$, assuming that $B(i)$ is greater than the time needed to print y from its literary description (otherwise the statement is trivial). So the lemma again can be used (in the simple direction). □

This proof used the depth representation of the stochasticity curve; in other cases some other representation are more convenient. Our second example is the change in stochasticity profile when a simple algorithmic transformation is applied. We have seen (Section 2.3) that a total mapping with a short program preserves stochasticity, and noted that for non-total mapping it is not the case (Remark 3, p. 10). However, if the time needed to perform the transformation is bounded, we can get some bound (first proven by A. Milovanov in a different way):

Proposition 26. Let F be a computable mapping whose arguments and values are strings. If some n-bit string x is (α,β)-stochastic, and $F(x)$ is computed in time $B(i)$ for some i, then $F(x)$ is $(\max(\alpha,i) + O(\log n), \beta + O(\log n))$-stochastic. (The constant in $O(\log n)$-notation depends on F but not on n, x, α, β.)

Proof sketch. Let us denote $F(x)$ by y. By assumption there exist a $(\alpha * (\mathrm{C}(x) - \alpha + \beta))$-description of x (recall the definition with optimality deficiency; we omit logarithmic terms as usual). So there exists a simple enumeration of at most $2^{\mathrm{C}(x)+\beta}$ objects x' in at most 2^α portions that includes x. Let us count x' in this enumeration such that $F(x') = y$ and the computation uses time at most $B(i)$; assume there are 2^s of them. Then we can enumerate all y's that have at least 2^s preimages in time $B(i)$, in $2^\alpha + 2^i$ portions. Indeed, new portions appear in two cases: (1) a new portion appears in the original enumeration; (2) candidate for $B(i)$ increases. The first event happens at most 2^α times, the second at most 2^i times. The total number of y's enumerated is $2^{\mathrm{C}(x)+\beta-s}$; it remains to note that $\mathrm{C}(x) - s \le \mathrm{C}(y)$. Indeed, $\mathrm{C}(x) \le \mathrm{C}(y) + \mathrm{C}(x|y)$, and $\mathrm{C}(x|y) \le s$, since we can enumerate all the preimages of y in the order of increasing time, and x is determined by s-bit ordinal number of x in this enumeration. □

A special case of this proposition is Bennett's observation: if some d-incompressible program p produces x in time $B(i)$, then p is $(0,d)$-stochastic, and p is mapped to x by the

interpreter (decompressor) in time $B(i)$, so x is $(0 + i, d)$-stochastic. (For simplicity we omit all the logarithmic terms in this argument, as well as in the previous proof sketch.)

Remark 10. One can combine Remark 4 (page 10) with Proposition 26 and show that if a program F of complexity at most j is applied to an (α, β)-stochastic string x of length n and the computation terminates in time $B(i)$, then $F(x)$ is $(\max(i, \alpha) + j + O(\log n), \beta + j + O(\log n))$-stochastic, where the constant in $O(\log n)$ notation is absolute (does not depend on F). To show this, one may consider the pair (x, F); it is easy to show (this can be done in different ways using different characterizations of the stochasticity curve) that this pair is $(\alpha + j + O(\log n), \beta + j + O(\log n))$-stochastic.

Let us note also that there are some results in algorithmic information theory that are true for stochastic objects but are false or unknown without this assumption. We will discuss (without proofs) two examples of this type. The first is Epstein–Levin theorem saying that for a stochastic set A its total a priori probability is close to the maximum a priori probability of A's elements; see [29] for details. Here the result is (obviously) false without stochasticity assumption.

In the next example [27] the stochasticity assumption is used in the proof, and it is not known whether the statement remains true without it: *for every triple of strings* (x, y, z) *of length at most* n *there exists a string* z' *such that*

- $\mathrm{C}(x|z) = \mathrm{C}(x|z') + O(\log n)$,
- $\mathrm{C}(y|z) = \mathrm{C}(y|z') + O(\log n)$,
- $\mathrm{C}(x, y|z) = \mathrm{C}(x, y|z') + O(\log n)$;
- $\mathrm{C}(z') \le I((x, y) : z) + O(\log n)$,

assuming that (x, y) *is* $(O(\log n), O(\log n))$*-stochastic.*

This proposition is related to the following open question on "irrelevant oracles": assume that the mutual information between (x, y) and some z is negligible. Can an oracle z (an "irrelevant oracle") change substantially any natural properties of the pair (x, y) formulated in terms of Kolmogorov complexity? For instance, can such an oracle z allow us to extract some common information of x and y? In [27] a negative answer to the latter question is given, but only for stochastic pairs (x, y).

6 Descriptions of restricted type

6.1 Family of descriptions

In this section we consider the restricted case: the sets (considered as descriptions, or statistical hypotheses) are taken from some family $\mathscr{A}$ that is fixed in advance.[9] (Elements of $\mathscr{A}$ are finite sets of binary strings.) Informally speaking, this means that we have some *a priori* information about the black box that produces a given string: this string is obtained by a random choice in one of the $\mathscr{A}$-sets, but we do not know in which one.

Before we had no restrictions (the family $\mathscr{A}$ was the family of all finite sets). It turns out that the results obtained so far can be extended (sometimes with weaker bounds) to other families that satisfy some natural conditions. Let us formulate these conditions.

(1) The family $\mathscr{A}$ is enumerable. This means that there exists an algorithm that prints elements of $\mathscr{A}$ as lists, with some separators (saying where one element of $\mathscr{A}$ ends and another one begins).

(2) For every n the family $\mathscr{A}$ contains the set $\mathbb{B}^n$ of all n-bit strings.

(3) The exists some polynomial p with the following property: for every $A \in \mathscr{A}$, for every natural n and for every natural $c < \#A$ the set of all n-bit strings in A can be covered by at most $p(n) \cdot \#A/c$ sets of cardinality at most c from $\mathscr{A}$.

The last condition is a replacement for splitting: in general, we cannot split a set $A \in \mathscr{A}$ into pieces from A, but at least we can cover a set $A \in \mathscr{A}$ by smaller elements of $\mathscr{A}$ (of size at most c) with polynomial overhead in the number of pieces, compared to the required minimum $\#A/c$ (more precisely, we have to cover only n-bit elements of A).

We assume that some family $\mathscr{A}$ that has properties (1)–(3) is fixed. For a string x we denote by $P_x^{\mathscr{A}}$ the set of pairs (i, j) such that x has $(i * j)$-description *that belongs to* $\mathscr{A}$. The set $P_x^{\mathscr{A}}$ is a subset of P_x defined earlier; the bigger $\mathscr{A}$ is, the bigger is $P_x^{\mathscr{A}}$. The full set P_x is $P_x^{\mathscr{A}}$ for the family $\mathscr{A}$ that contains all finite sets.

For every string x the set $P_x^{\mathscr{A}}$ has properties close to the properties of P_x proved earlier.

Proposition 27. For every string x of length n the following is true:

1. The set $P_x^{\mathscr{A}}$ contains a pair that is $O(\log n)$-close to $(0, n)$.
2. The set $P_x^{\mathscr{A}}$ contains a pair that is $O(1)$-close to $(\mathrm{C}(x), 0)$.
3. The adaptation of Proposition 8 is true: if $(i, j) \in P_x^{\mathscr{A}}$, then $(i + k + O(\log n), j - k)$ also belongs to $P_x^{\mathscr{A}}$ for every $k \leq j$. (Recall that n is the length of x.)

[9] One can also consider some class of probability distributions, but we restrict our attention to sets (uniform distributions).

Proof. 1. The property (2) guarantees that the family $\mathscr{A}$ contains the set $\mathbb{B}^n$ that is an $(O(\log n) * n)$-description of x.

2. The property (3) applied to $c = 1$ and $A = \mathbb{B}^n$ says that every singleton belongs to A, therefore each string has $((\mathrm{C}(x) + O(1)) * 0)$-description.

3. Assume that x has $(i * j)$-description $A \in \mathscr{A}$. For a given k we enumerate $\mathscr{A}$ until we find a family of $p(n)2^k$ sets of size $2^{-k}\#A$ (or less) in $\mathscr{A}$ that covers all strings of length n in A. Such a family exists due to (3), and p is the polynomial from (3). The complexity of the set that covers x does not exceed $i+k+O(\log n+\log k)$, since this set is determined by A, n, k and the ordinal number of the set in the cover. We may assume without loss of generality that $k \le n$, otherwise $\{x\}$ can be used as $((i+k+O(\log n)) * (j-k))$-description of x. So the term $O(\log k)$ can be omitted. □

For example, we may consider the family that consists of all "cylinders": for every n and for every string u of length at most n we consider the set of all n-bit strings that have prefix u. Obviously the family of all such sets (for all n and u) satisfies the conditions (1)–(3).

We may also fix some bits of a string (not necessarily forming a prefix). That is, for every string z in ternary alphabet $\{0, 1, *\}$ we consider the set of all bit strings that can be obtained from z by replacing stars with some bits. This set contains 2^k strings, if u has k stars. The conditions (1)–(3) are fulfilled for this larger family, too.

A more interesting example is the family $\mathscr{A}$ formed by all balls in Hamming sense, i.e., the sets $B_{y,r} = \{x \mid l(x) = l(y), d(x,y) \le r\}$. Here $l(u)$ is the length of binary string u, and $d(x,y)$ is the Hamming distance between two strings x and y of the same length. The parameter r is called the *radius* of the ball, and y is its *center*. Informally speaking, this means that the experimental data were obtained by changing at most r bits in some string y (and all possible changes are equally probable). This assumption could be reasonable if some string y is sent via an unreliable channel. Both parameters y and r are not known to us in advance.

It turns out that the family of Hamming balls satisfies the conditions (1)–(3). This is not completely obvious. For example, these conditions imply that for every n and for every $r \le n$ the set $\mathbb{B}^n$ of n-bit strings can be covered by $\mathrm{poly}(n)2^n/V$ Hamming balls of radius r, where V stands for the cardinality of such a ball (i.e., $V = \binom{n}{0} + ... + \binom{n}{r}$), and p is some polynomial. This can be shown by a probabilistic argument: take N balls of radius r whose centers are randomly chosen in $\mathbb{B}^n$. For a given $x \in \mathbb{B}^n$ the probability that x is not covered by any of these balls equals $(1 - V/2^n)^N < e^{-VN/2^n}$. For $N = n \ln 2 \cdot 2^n/V$ this upper bound is 2^{-n}, so for this N the probability to leave some x uncovered is less than 1. A similar argument can be used to prove (1)–(3) in the general case.

Proposition 28 ([43]). The family of all Hamming balls satisfies conditions (1)–(3) above.

Proof sketch. Let A be a ball of radius a and let c be a number less than $\#A$. We need to cover A by balls of cardinality c or less, using almost minimal number of balls, close to the lower bound $\#A/c$ up to a polynomial factor. Let us make some observations.

(1) The set of all n-bit strings can be covered by two balls of radius $n/2$. So we can assume without loss of generality that $a \le n/2$, otherwise we can apply the probabilistic argument above.

(2) Clearly the radius of covering balls should be maximal possible (to keep cardinality less than c); for this radius the cardinality of the ball equals c up to polynomial factors, since the size of the ball increases at most by factor $n+1$ when its radius increases by 1.

(3) It is enough to cover spheres instead of balls (since every ball is a union of polynomially many spheres); it is also enough to consider the case when the radius of the sphere that we want to cover (a) is bigger than the radius of the covering ball (b), otherwise one ball is enough.

(4) We will cover a-sphere by randomly chosen b-balls whose centers are uniformly taken at some distance f from the center of a-sphere. (See below about the choice of f.) We use the same probabilistic argument as before (for the set of all strings). It is enough to show that for a b-ball whose center is at that distance, the polynomial fraction of points belong to a-sphere. Instead of b-balls we may consider b-spheres, the cardinality ratio is polynomial.

(5) It remains to choose some f with the following property: if the center of a b-sphere S is at a distance f from the center of a-sphere T, then the polynomial fraction of S-points belong to T. One can compute a suitable f explicitly. In probabilistic terms we just change f/n-fraction of bits and then change random b/n fraction of bits. The expected fraction of twice changed bits is, therefore, about $(f/n)(b/n)$, and the total fraction of changed bits is about $f/n + b/n - 2(f/n)(b/n)$. So we need to write an equation saying that this expression is a/n and the find the solution f. (Then one can perform the required estimate for binomial coefficients.)

However, one can avoid computations with the following probabilistic argument: start with b changed bits, and then change all the bits one by one in a random order. At the end we hat $n-b$ changed bits, and a is somewhere in between, so there is a moment where the number of changed bits is exactly a. And if the union of n events covers the entire probability space, one of these events has probability at least $1/n$. □

When a family $\mathscr{A}$ is fixed, a natural question arises: does the restriction on models (when we consider only models in $\mathscr{A}$) changes the set P_x? Is it possible that a string has good models in general, but not in the restricted class? The answer is positive for the class of Hamming balls, as the following proposition shows.

Proposition 29. Consider the family $\mathscr{A}$ that consists of all Hamming balls. For some positive ε and for all sufficiently large n there exists a string x of length n such that the distance between $P_x^{\mathscr{A}}$ and P_x exceeds εn.

Proof sketch. Fix some α in $(0, 1/2)$ and let V be the cardinality of the Hamming ball of radius αn. Find a set E of cardinality $N = 2^n/V$ such that every Hamming ball of radius αn contains at most n points from E. This property is related to *list decoding* in the coding theory. The existence of such a set can be proved by a probabilistic argument: N randomly chosen n-bit strings have this property with positive probability. Indeed, the probability of a random point to be in E is an inverse of the number of points, so the distribution is close to Poisson distribution with parameter 1, and tails decrease much faster that 2^{-n} needed.

Since E with this property can be found by an exhaustive search, we can assume that $\mathrm{C}(E) = O(\log n)$ and ignore the complexity of E (as well as other $O(\log n)$ terms) in the sequel. Let x be a random element in E, i.e., a string $x \in E$ of complexity about $\log \#E$. The complexity of a ball A of radius αn that contains x is at least $\mathrm{C}(x)$, since knowing such a ball and an ordinal number of x in $A \cap E$, we can find x. Therefore x does not have $(\log \#E, \log V)$-descriptions in $\mathscr{A}$. On the other hand, x does have $(0, \log \#E)$-description if we do not require the description to be in $\mathscr{A}$; the set E is such a description. The point $(\log \#E, \log V)$ is above the line $\mathrm{C}(A) + \log \#A = \log \#E$, so $P_x^{\mathscr{A}}$ is significantly smaller than P_x. □

This construction gives a stochastic x (E is the corresponding model) that becomes maximally non-stochastic if we restrict ourselves to Hamming balls as descriptions (Figure 7).

6.2 Possible shapes of boundary curve

Our next goal is to extend some results proven for non-restricted descriptions to the restricted case. Let $\mathscr{A}$ be a family that has properties (1)–(3). We prove a version of Theorem 1 where the precision (unfortunately) is significantly worse: $O(\sqrt{n \log n})$ instead of $O(\log n)$. Note that with this precision the term $O(m)$ (proportional to the complexity of the curve) that appeared in Theorem 1 is not needed. Indeed, if we draw the curve on the cell paper with cell size $\sqrt{n}$ or larger, then it touches only $O(\sqrt{n})$ cells, so it is determined by $O(\sqrt{n})$ bits with $O(\sqrt{n})$-precision, and we may assume without loss of generality that the complexity of the curve is $O(\sqrt{n})$.

Theorem 7 ([43]). *Let $k \le n$ be two integers and let $t_0 > t_1 > \ldots > t_k$ be a strictly decreasing sequence of integers such that $t_0 \le n$ and $t_k = 0$.. Then there exists a string x of*

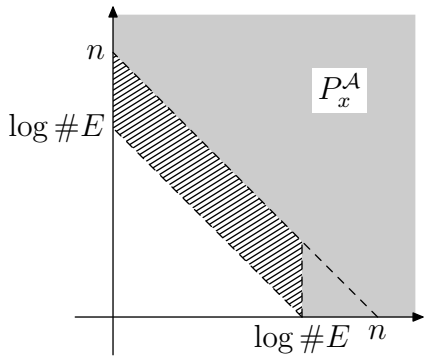

Figure 7: Theorem 8 can be used (together with the argument above) to show that the border of the set $P_x^{\mathcal{A}}$ (shown in gray) consists of a vertical segment $\mathrm{C}(A) = n - \log V$, $\log \#A \le \log V$, and the segment of slope -1 defined by $\mathrm{C}(A) + \log \#A = n$, $\log V \le \log \#A$. The set P_x contains also the hatched part.

complexity $k + O(\sqrt{n \log n})$ *and length* $n + O(\log n)$ *for which the distance between* $P_x^{\mathcal{A}}$ *and* $T = \{(i,j) \mid (i \le k) \Rightarrow (j \ge t_i)\}$ *is at most* $O(\sqrt{n \log n})$.

We will see later (Theorem 8) that for every x the boundary curve of $P_x^{\mathcal{A}}$ goes down at least with slope -1, as for the unrestricted case, so this theorem describes all possible shapes of the boundary curve.

Proof. The proof is similar to the proof of Theorem 1. Let us recall this proof first. We consider the string x that is the lexicographically first string (of suitable length n') that is not covered by any "bad" set, i.e., by any set of complexity at most i and size at most 2^j, where the pair (i,j) is at the boundary of the set T. The length n' is chosen in such a way that the total number of strings in all bad sets is strictly less than $2^{n'}$. On the other hand, we need "good sets" that cover x. For every boundary point (i,j) we construct a set $A_{i,j}$ that contains x, has complexity close to i and size 2^j. The set $A_{i,j}$ is constructed in several attempts. Initially $A_{i,j}$ is the set of lexicographically first 2^j strings of length n'. Then we enumerate bad sets and delete all their elements from $A_{i,j}$. At some step $A_{i,j}$ may become empty; then we refill it with 2^j lexicographically first strings that are not in the bad sets (at the moment). By construction the final $A_{i,j}$ contains the first x that is not in bad sets (since it is the case all the time). And the set $A_{i,j}$ can be described by the number of changes (plus some small information describing the process as a whole and the value of j). So it is crucial to have an upper bound for the number of changes. How do we get this bound? We note that when $A_{i,j}$ becomes empty, it is refilled again, and all the new

elements should be covered by bad sets before the new change could happen. Two types of bad sets may appear: "small" ones (of size less than 2^j) and "large ones" (of size at least 2^j). The slope of the boundary line for T guarantees that the total number of elements in all small bad sets does not exceed 2^{i+j} (up to a $\mathrm{poly}(n)$-factor), so they may make $A_{i,j}$ empty only 2^i times. And the number of large bad sets is $O(2^i)$, since the complexity of each is bounded by i. (More precisely, we count separately the number of changes for $A_{i,j}$ that are first changes after a large bad set appears, and the number of other changes.)

Can we use the same argument in our new situation? We can generate bad sets as before and have the same bounds for their sizes and the total number of their elements. So the length n' of x can be the same (in fact, almost the same, as we will need now that the union of all bad sets is less than half of all strings of length n', see below). Note that we now may enumerate only bad sets in $\mathscr{A}$, since $\mathscr{A}$ is enumerable, but we do not even need this restriction. What we cannot do is to let $A_{i,j}$ to be the set of the first non-deleted elements: we need $A_{i,j}$ to be a set from $\mathscr{A}$.

So we now go in the other direction. Instead of choosing x first and then finding suitable "good" $A_{i,j}$ that contain x, we construct the sets $A_{i,j} \in \mathscr{A}$ that change in time in such a way that (1) their intersection always contains some non-deleted element (an element that is not yet covered by bad sets); (2) each $A_{i,j}$ has not too many versions. The non-deleted element in their intersection (in the final state) is then chosen as x.

Unfortunately, we cannot do this for all points (i,j) along the boundary curve. (This explains the loss of precision in the statement of the theorem.) Instead, we construct "good" sets only for some values of j. These values go down from n to 0 with step $\sqrt{n \log n}$. We select $N = \sqrt{n/\log n}$ points $(i_1,j_1), \ldots, (i_N,j_N)$ on the boundary of T; the first coordinates $i_1, \ldots, i_N$ form a non-decreasing sequence, and the second coordinates $j_1, \ldots, j_N$ split the range $n \ldots 0$ into (almost) equal intervals ($j_1 = n$, $j_N = 0$). Then we construct good sets of sizes at most $2^{j_1}, \ldots, 2^{j_N}$, and denote them by $A_1, \ldots, A_N$. All these sets belong to the family $\mathscr{A}$. We also let A_0 to be the set of all strings of length $n' = n + O(\log n)$; the choice of the constant in $O(\log n)$ will be discussed later.

Let us first describe the construction of $A_1, \ldots, A_N$ assuming that the set of deleted elements is fixed. (Then we discuss what to do when more elements are deleted.) We construct A_s inductively (first A_1, then A_2 etc.). As we have said, $\#A_s \le 2^{j_s}$ (in particular, A_N is a singleton), and we keep track of the ratio

$$(\text{the number of non-deleted strings in } A_0 \cap A_1 \cap \ldots \cap A_s)/2^{j_s}.$$

For $s = 0$ this ratio is at least 1/2; this is obtained by a suitable choice of n' (the union of all bad sets should cover at most half of all n'-bit strings). When constructing the next A_s, we ensure that this ratio decreases only by $\mathrm{poly}(n)$-factor. How? Assume that A_{s-1} is already constructed; its size is at most $2^{j_{s-1}}$. The condition (3) for $\mathscr{A}$ guarantees that

A_{s-1} can be covered by $\mathscr{A}$-sets of size at most 2^{j_s}, and we need about $2^{j_{s-1}-j_s}$ covering sets (up to $\mathrm{poly}(n)$-factor). Now we let A_s be the covering set that contains maximal number of non-deleted elements in $A_0 \cap \ldots \cap A_{s-1}$. The ratio can decrease only by the same $\mathrm{poly}(n)$-factor. In this way we get

$$\text{(the number of non-deleted strings in } A_0 \cap A_1 \cap \ldots \cap A_s) \geq \alpha^{-s} 2^{j_s}/2,$$

where α stands for the $\mathrm{poly}(n)$-factor mentioned above.[10]

Up to now we assumed that the set of deleted elements is fixed. What happens when more strings are deleted? The number of the non-deleted in $A_0 \cap \ldots \cap A_s$ can decrease, and at some point and for some s can become less than the declared threshold $\nu_s = \alpha^{-s} 2^{j_s}/2$. Then we can find minimal s where this happens, and rebuild all the sets $A_s, A_{s+1}, \ldots$ (for A_s the threshold is not crossed due to the minimality of s). In this way we update the sets A_s from time to time, replacing them (and all the consequent ones) by new versions when needed.

The problem with this construction is that the number of updates (different versions of each A_s) can be too big. Imagine that after an update some element is deleted, and the threshold is crossed again. Then a new update is necessary, and after this update next deletion can trigger a new update, etc. To keep the number of updates reasonable, we agree that after the update *for all the new sets* A_l (starting from A_s) *the number of non-deleted elements in* $A_0 \cap \ldots \cap A_l$ *is twice bigger than the threshold* $\nu_l = \alpha^{-l} 2^{j_l}/2$. This can be achieved if we make the factor α twice bigger: since for A_{s-1} we have not crossed the threshold, for A_s we can guarantee the inequality with additional factor 2.

Now let us prove the bound for the number of updates for some A_s. These updates can be of two types: first, when A_s itself starts the update (being the minimal s where the threshold is crossed); second, when the update is induced by one of the previous sets. Let us estimate the number of the updates of the first type. This update happens when the number of non-deleted elements (that was at least $2\nu_s$ immediately after the previous update of any kind) becomes less than ν_s. This means that at least ν_s elements were deleted. How can this happen? One possibility is that a new bad set of complexity at most i_s ("large bad set") appears after the last update. This can happen at most $O(2^{i_s})$ times, since there is at most $O(2^i)$ objects of complexity at most i. The other possibility is the accumulation of elements deleted due to "small" bad sets, of complexity at least i_s and of size at most 2^{j_s}. The total number of such elements is bounded by $nO(2^{i_s+j_s})$, since the sum $i_l + j_l$ may only decrease as l, increases. So the number of updates of A_s not caused

[10]Note that for the values of s close to N the right-hand side can be less than 1; the inequality then claims just the existence of non-deleted elements. The induction step is still possible: non-deleted element is contained in one of the covering sets.

by large bad sets is bounded by

$$nO(2^{i_s+j_s})/\nu_s = \frac{O(n2^{i_s+j_s})}{\alpha^{-s}2^{j_s}} = O(n\alpha^s 2^{i_s}) = 2^{i_s+NO(\log n)} = 2^{i_s+O(\sqrt{n\log n})}$$

(recall that $s \le N$, $\alpha = \mathrm{poly}(n)$, and $N \approx \sqrt{n/\log n}$). This bound remains valid if we take into account the induced updates (when the threshold is crossed for the preceding sets: there are at most $N \le n$ these sets, and additional factor n is absorbed by O-notation).

We conclude that all the versions of A_s have complexity at most $i_s + O(\sqrt{n\log n})$, since each of them can be described by the version number plus the parameters of the generating process (we need to know n and the boundary curve, whose complexity is $O(\sqrt{n})$ according to our assumption, see the discussion before the statement of the theorem). The same is true for the final version. It remains to take x in the intersection of the final sets A_s. (Recall that A_N is a singleton, so final A_N is $\{x\}$.) Indeed, by construction this x has no bad $(i*j)$-descriptions where (i,j) is on the boundary of T. On the other hand, x has good descriptions that are $O(\sqrt{n\log n})$-close to this boundary and whose vertical coordinates are $\sqrt{n\log n}$-apart. (Recall that the slope of the boundary guarantees that horizontal distance is less than the vertical distance.) Therefore the position of the boundary curve for $P_x^{\mathscr{A}}$ is determined with precision $O(\sqrt{n\log n})$, as required.[11] □

Remark 11. In this proof we may use bad sets not only from $\mathscr{A}$. Therefore, the set P_x is also close to T (and the same is true for for every family $\mathscr{B}$ that contains $\mathscr{A}$). It would be interesting to find out what are the possible combinations of P_x and $P_x^{\mathscr{A}}$; as we have seen, it may happen that P_x is maximal and $P_x^{\mathscr{A}}$ is minimal, but this does not say anything about other possible combinations.

For the case of Hamming balls the statement of Theorem 7 has a natural interpretation. To find a simple ball of radius r that contains a given string x is the same as to find a simple string in a radius r ball centered at x. So this theorem show the possible behavior of the "approximation complexity" function

$$r \mapsto \min\{\mathrm{C}(x') \mid d(x,x') \le r\}$$

where d is Hamming distance. One should only rescale the vertical axis replacing the log-sizes of Hamming balls by their radii. The connection is described by the Shannon entropy function: a ball in $\mathbb{B}^n$ of radius r has log-size about $nH(r/n)$ for $r \le n/2$, and

[11] Now we see why N was chosen to be $\sqrt{n/\log n}$: the bigger N is, the more points on the curve we have, but then the number of versions of the good sets and their complexity increases, so we have some trade-off. The chosen value of n balances these two sources of errors.

has almost full size for $r \geq n/2$. For example, error correcting codes (in classical sense, or with list decoding) are example of strings where this function is almost a constant for small values of r: it is almost as easy to approximate a codeword as give it precisely (due to the possibility of error correction).

6.3 Randomness and optimality deficiencies: restricted case

Not all the results proved for unrestricted descriptions have natural counterparts in the restricted case. For example, one hardly can relate the set $P_x^{\mathscr{A}}$ with bounded-time complexity (is completely unclear how $\mathscr{A}$ could enter the picture). Still some results remain valid (but new and much more complicated proofs are needed). This is the case for Proposition 8 and 9.

Let again $\mathscr{A}$ be the class of descriptions that satisfies requirements (1)–(3).

Theorem 8 ([43]).

- *If a string x of length n has an $(i*j)$-description in $\mathscr{A}$, then it has $((i+d+O(\log n))*(j-d+O(\log n)))$-description in $\mathscr{A}$ for every $d \leq j$.*
- *Assume that x is a string of length n that has at least 2^k different $(i*j)$-descriptions in $\mathscr{A}$. Then it has $((i-k+O(\log n))*(j+O(\log n))$-description in $\mathscr{A}$.*

In fact, the second part uses only condition (1); it says that $\mathscr{A}$ is enumerable. The first part uses also (3). It can be combined with the second part to show that x has also $((i+O(\log n))*(j-k+O(\log n))$-description in $\mathscr{A}$.

Proof. The first part is easy: having some $(i*j)$-description for x, we can search for a covering by the sets of right size that exists due to condition (3); since $\mathscr{A}$ is enumerable, we can do it algorithmically until we find this covering. Then we select the first set in the covering that contains x; the bound for the complexity of this set is guaranteed by the size of the covering.

The proof of the second statement is much more interesting. In fact, there are two different proofs: one uses a probabilistic existence argument and the second is more explicit. But both of them start in the same way.

Let us enumerate all $(i*j)$-descriptions from $\mathscr{A}$, i.e., all finite sets that belong to $\mathscr{A}$, have cardinality at most 2^j and complexity at most i. For a fixed n, we start a selection process: some of the generated descriptions are marked (=selected) immediately after their generation. This process should satisfy the following requirements: (1) at any moment every n-bit string x that has at least 2^k descriptions (among enumerated ones) belongs to one of the marked descriptions; (2) the total number of marked sets does not

exceed $2^{i-k}p(n)$ for some polynomial p. Note that for $i \geq n$ or $j \geq n$ the statement is trivial, so we may assume that i, j (and therefore k) do not exceed n; this explains why the polynomial depends only on n.

If we have such a strategy (of logarithmic complexity), then the marked set containing x will be the required description of complexity $i - k + O(\log n)$ and log-size j. Indeed, this marked set can be specified by its ordinal number in the list of marked sets, and this ordinal number has $i - k + O(\log n)$ bits.

So we need to construct a selection strategy of logarithmic complexity. We present two proofs: a probabilistic one and an explicit construction.

Probabilistic proof. First we consider a finite game that corresponds to our situation. Two players alternate, each makes 2^i moves. At each move the first player presents some set of n-bit strings, and the second player replies saying whether it *marks* this set or not. The second player loses if after some moves the number of marked sets exceeds $2^{i-k+1}(n+1)\ln 2$ (this specific value follows from the argument below) or if there exists a string x that belongs to 2^k sets of the first player but does not belong to any marked set.

Since this is a finite game with full information, one of the players has a winning strategy. We claim that the second player can win. If it is not the case, the first player has a winning strategy. We get a contradiction by showing that the second player has a *probabilistic* strategy that wins with positive probability against any strategy of the first player. So we assume that some (deterministic) strategy of the first player is fixed, and consider the following simple probabilistic strategy: every set A presented by the first player is marked with probability $p = 2^{-k}(n+1)\ln 2$.

The expected number of marked sets is $p2^i = 2^{i-k}(n+1)\ln 2$. By Chebyshev's inequality, the number of marked set exceeds the expectation by a factor 2 with probability less than 1/2. So it is enough to show that the second bad case (after some move there exists x that belongs to 2^k sets of the first player but does not belong to any marked set) happens with probability at most 1/2.

For that, it is enough to show that for every fixed x the probability of this bad event is at most $2^{-(n+1)}$, and then use the union bound. The intuitive explanation is simple: if x belongs to 2^k sets, the second player had (at least) 2^k chances to mark a set containing x (when these 2^k sets were presented by the first player), and the probability to miss all these chances is at most $(1-p)^{2^k}$; the choice of p guarantees that this probability is less than $1/2^{-(n+1)}$. Indeed, using the bound $(1-1/x)^x < 1/e$, it is easy to show that $(1-p)^{2^k} < e^{-(n+1)\ln 2} = 2^{-(n+1)}$.

The pedantic reader would say that this argument is not formally correct, since the behavior of the first player (and the moment when next set containing x is produced) depends on the moves of the second player, so we do not have independent events with

probability $1-p$ each (as it is assumed in the computation).[12] The formal argument considers for each t the event R_t: "after some move of the second player the string x belongs to at least t sets provided by the first player, but does not belong to any marked set". Then we prove by induction (over t) that the probability of R_t does not exceed $(1-p)^t$. Indeed, it is easy to see that R_t in a union of several disjoint subsets (depending on the events happening until the first player provides $t+1$ sets containing x), and R_{t+1} is obtained by taking a $(1-p)$-fraction in each of them.

Constructive proof. We consider the same game, but now allow more sets to be marked (replacing the bound $2^{i-k+1}(n+1)\ln 2$ by a bigger bound $2^{i-k}i^2\ln 2$) and also allow the second player to mark sets that were produced earlier (not necessarily at the current move of the first player). The explicit winning strategy for the second player performs in parallel $i-k+\log i$ substrategies (indexed by the numbers $\log(2^k/i), \dots, i$).

The substrategy number s wakes up once in 2^s moves (when the number of moves made by the first player is a multiple of 2^s). It considers a family S that consists of 2^s last sets produced by the first player, and the set T that consists of all strings x covered by at least $2^k/i$ sets from S. Then it selects and marks some elements in S in such a way that all $x\in T$ are covered by one of the selected sets. It is done by a greedy algorithm: first take a set from S that covers maximal part of T, then the set that covers maximal number of non-covered elements, etc. How many steps do we need to cover the entire T? Let us show that

$$(i/2^k)n2^s\ln 2$$

steps are enough. Indeed, every element of T is covered by at least $2^k/i$ sets from S. Therefore, some set from S covers at least $\#T2^k/(i2^s)$ elements, i.e., $2^{k-s}/i$-fraction of T. At the next step the non-covered part is multiplied by $(1-2^{k-s}/i)$ again, and after $in2^{s-k}\ln 2$ steps the number of non-covered elements is bounded by

$$\#T(1-2^{k-s}/i)^{in2^{s-k}\ln 2} < 2^n(1/e)^{n\ln 2} = 1,$$

therefore all elements of T are covered. (Instead of a greedy algorithm one may use a probabilistic argument and show that randomly chosen $in2^{s-k}\ln 2$ sets from S cover T with positive probability; however, our goal is to construct an explicit strategy.)

Anyway, the number of sets selected by a substrategy number s, does not exceed

$$in2^{s-k}(\ln 2)2^{i-s} = in2^{i-k}\ln 2,$$

[12] The same problem appears if we observe a sequence of independent coin tossings with probability of success p, select some trials (before they are actually performed, based on the information obtained so far), and ask for the probability of the event "t first selected trials were all unsuccessful". This probability does not exceed $(1-p)^t$; it can be smaller if the total number of selected trials is less than t with positive probability. This scheme was considered by von Mises when he defined random sequences using selection rules, so it should be familiar to algorithmic randomness people.

and we get at most $i^2 n 2^{i-k} \ln 2$ for all substrategies.

It remains to prove that after each move of the second player every string x that belongs to 2^k or more sets of the first player, also belongs to some selected set. For tth move we consider the binary representation of t:

$$t = 2^{s_1} + 2^{s_2} + \ldots, \text{ where } s_1 > s_2 > \ldots$$

Since x does not belong to the sets selected by substrategies with numbers $s_1, s_2, \ldots$, the multiplicity of x among the first 2^{s_1} sets is less than $2^k/i$, the multiplicity of x among the next 2^{s_2} sets is also less than $2^k/i$, etc. For those j with $2^{s_j} < 2^k/i$ the multiplicity of x among the respective portion of 2^{s_j} sets is obviously less than $2^k/i$. Therefore, we conclude that the total multiplicity of x is less that $i \cdot 2^k/i = 2^k$ sets of the first player and the second player does not need to care about x. This finishes the explicit construction of the winning strategy.

Now we can assume without loss of generality that the winning strategy has complexity at most $O(\log(n + k + i + j))$. (In the probabilistic argument we have proved the existence of a winning strategy, but then we can perform the exhaustive search until we find one; the first strategy found will have small complexity.) Then we use this simple strategy to play with the enumeration of all $\mathscr{A}$-sets of complexity less than i and size 2^j (or less). The selected sets can be described by their ordinal number (among the selected sets), so their complexity is bounded by $i - k$ (with logarithmic precision). Every string that has 2^k different $(i * j)$-descriptions in $\mathscr{A}$, will also have one among the selected sets, and that is what we need. □

As before (for the unrestricted case), this result implies that descriptions with minimal parameters are simple with respect to the data string:

Theorem 9 ([43]). *Let $\mathscr{A}$ be an enumerable family of finite sets. If a string x of length n has $(i * j)$-description $A \in \mathscr{A}$ such that $\mathrm{C}(A|x) \geq k$, then x has a $((i - k + O(\log n)) * (j + O(\log n)))$-description in $\mathscr{A}$. If the family $\mathscr{A}$ satisfies the condition* (3)*, then x has also a $((i + O(\log n)) * (j - k + O(\log n)))$-description in $\mathscr{A}$.*

This gives us the same corollaries as in the unrestricted case:

Corollary. Let $\mathscr{A}$ be a family of finite sets that satisfies the conditions (1)–(3). Then for every string x of length n three statements

- there exists a set $A \in \mathscr{A}$ of complexity at most α with $d(x|A) \leq \beta$;
- there exists a set $A \in \mathscr{A}$ of complexity at most α with $\delta(x, A) \leq \beta$;
- the point $(\alpha, \mathrm{C}(x) - \alpha + \beta)$ belongs to $P_x^{\mathscr{A}}$

are equivalent with logarithmic precision (the constants before the logarithms depend on the choice of the set $\mathscr{A}$).

If we are interested in the uniform statements true for every enumerable family $\mathscr{A}$, the same arguments prove the following result:

Proposition 30. Let $\mathscr{A}$ be an arbitrary family of finite sets enumerated by some program p. Then for every x of length n the statements

- there exists a set $A \in \mathscr{A}$ such that $d(x|A) \le \beta$;
- there exists a set $A \in \mathscr{A}$ such that $\delta(x, A) \le \beta$

are equivalent up to $O(\mathrm{C}(p) + \log \mathrm{C}(A) + \log n + \log\log \#A)$-change in the parameters.

7 Strong models

7.1 Information in minimal descriptions

A possible way to bring the theory in accordance to our intuition is to change the definition of "having the same information". Although we have not given that definition explicitly, we have adopted so far the following viewpoint: x and y have the same (or almost the same) information if both conditional complexities $\mathrm{C}(x|y), \mathrm{C}(y|x)$ are small. If only one complexity, say $\mathrm{C}(x|y)$, is small, we said that all (or almost all) information contained in x is present in y.

Now we will adopt a more restricted viewpoint and say that x and y have the same information if there are short *total* (everywhere defined) programs mapping x to y and vice versa. From this viewpoint we cannot say any more that a string x and its shortest program x^* have the same information: for example, x may be antistochastic while x^* is always stochastic, so there is no short total program that maps x^* to x because of Proposition 3.[13] Let us mention that if x and y have the same information in this new sense, then there exists a simple computable *bijection* that maps x to y (so they have the same properties if the property is defined in the computability language), see [26] for the proof.

Formally, let us define the total conditional complexity with respect to a computable function D of two arguments, as

$$\mathrm{CT}_D(x|y) = \min\{l(p) \mid D(p, y) = x, \text{ and } D(p, y') \text{ is defined for all } y'\}.$$

[13]It is worth to mention that on the other hand, for all strings x there is an almost shortest program that can be obtained from x by a simple total transformation [39, Theorem 17].

(Note that D is not required to be total, but we consider only p such that $D(p, y')$ is defined for all y'.)

There is a computable function D such that CT_D is minimal up an additive constant. Fixing any such D we obtain the *total conditional complexity* $\mathrm{CT}(x|y)$. In other way, we may define $\mathrm{CT}(x|y)$ as the minimal plain complexity of a total program that maps y to x.

We will think that y has all (or almost all) the information from x if $\mathrm{CT}(x|y)$ is negligible. Formally, we write $x \overset{\varepsilon}{\to} y$ if $\mathrm{CT}(y|x) \le \varepsilon$ and we call x and y *ε-equivalent*, $x \overset{\varepsilon}{\leftrightarrow} y$, if both $\mathrm{CT}(y|x)$ and $\mathrm{CT}(x|y)$ are at most ε.

Proposition 31. If $x \overset{\varepsilon}{\leftrightarrow} y$ then the sets P_x and P_y are in $O(\varepsilon)$ neighborhood of each other.

Proof. Indeed, if A is an $(i*j)$-description of x and p is a total program witnessing $x \overset{\varepsilon}{\to} y$ then the set $B = \{D(p, x') \mid x' \in A\}$ is a $((i + O(\varepsilon)) * j)$-description of y. (We need p to be total, as otherwise we cannot produce the list of B from the list of A and p.) □

7.2 An attempt to separate "good" models from "bad" ones

Now we have more fine-grained classification of description and can try to distinguish between description that were equivalent in the former sense. For example, consider a string xy where y is random conditionally to x. Let A be a model consisting of all extension of x (of the same length). This model looks good (in particular, it has negligible optimality deficiency). On the other hand, we can get a standard model B for x of the same (or smaller) complexity that also has negligible optimality deficiency. This model looks bad in general. In this section we are interested in the following question: how can we formally distinguish good models like A from bad models like B? We will see that at least for some strings x the value $\mathrm{CT}(A|x)$ can be used. (Indeed, for A it is small, while for B instead of A it can be large.)

Definition 5. A set $A \ni x$ is an ε-strong model (or statistic) for a string x if $\mathrm{CT}(A|x) \le \varepsilon$.

For instance, the model A discussed above is an $O(\log n)$-strong model for x. On the other hand, we will see later that, if y is chosen appropriately, then no standard description B of the same complexity and log-cardinality as A is an ε-strong model for x, even for $\varepsilon = \Omega(n)$.

Strong models satisfy an analog of Proposition 8:

Proposition 32. Let x be a string and A be an ε-strong model for x. Let i be a non-negative integer such that $i \le \log \#A$. Then there exists a $\varepsilon + O(\log i)$-strong model A' for x such that $\#A' \le \#A/2^i$ and $\mathrm{C}(A') \le \mathrm{C}(A) + i + O(\log i)$.

To take into account the strength of models consider the set

$$P_x(\varepsilon) = \{(i,j) \mid x \text{ has an } \varepsilon\text{-strong } (i*j)\text{-description}\}.$$

Obviously, we have

$$P_x(\varepsilon) \subset P_x = P_x(n + O(1))$$

for all strings x of length n and for all ε.

If the set $P_x(\varepsilon)$ is close to the set P_x for a reasonably small ε, we will say that x is a "normal" string and otherwise we call x "strange". More precisely, a string x is called ε,δ-*normal* if P_x is in δ-neighborhood of $P_x(\varepsilon)$. Otherwise, x is called ε,δ-*strange*.

It turns out that there are $\sqrt{n\log n}, O(\log n)$-normal strings with any given set P_x, satisfying conditions of Theorem 1. On the other hand, there are $\Omega(n),\Omega(n)$-strange strings of length n. We are going to state these facts accurately.

Theorem 10 ([24]). *Let $k \le n$ be two integers and let $t_0 > t_1 > \ldots > t_k$ be a strictly decreasing sequence of integers such that $t_0 \le n$ and $t_k = 0$. Then there exists a string x of complexity $k + O(\sqrt{n\log n})$ and length $n + O(\log n)$ for which the distance between both sets P_x and $P_x(O(\log n))$ and the set $T = \{(i,j) \mid (i \le k) \Rightarrow (j \ge t_i)\}$ is at most $O(\sqrt{n\log n})$.*

Proof. Consider the family $\mathscr{A}$ of all cylinders, i.e., the family in the sets of the form $\{ur \mid l(r) = m\}$ for different strings u and natural numbers m. Sets from this family have the following feature: if $A \ni x$ then A is an $O(\log n)$-strong model for x. Hence for all strings x we have $P_x^{\mathscr{A}} = P_x^{\mathscr{A}}(O(\log n))$.

By Theorem 7 and Remark 11 there is a string x of length $n+O(\log n)$ and complexity $k+O(\sqrt{n\log n})$ such that all sets $P_x, P_x^{\mathscr{A}}, T$ are $O(\sqrt{n\log n})$-close to each other. Hence all the three sets are close to the set $P_x^{\mathscr{A}}(O(\log n))$ as well. As the set $P_x(O(\log n))$ includes the latter set and is included into P_x, all the three sets are close to the set $P_x(O(\log n))$ as well. □

Theorem 11 ([39]). *Assume that natural numbers k, n, ε satisfy the inequalities*

$$O(1) \le \varepsilon \le k \le n.$$

Then there is a string x of length n and complexity $k + O(\log n)$ such that the sets P_x and P_x^{ε} are $O(\log n)$-close to the sets shown on Fig. 8.

Let (say) $\varepsilon = n/3$ and $k = 2n/3$ in Theorem 11. Then the sets P_x and $P_x(n/3)$ are almost $n/3$-apart, since the point $(n/3, n/3)$ is in an $O(\log n)$-neighborhood of P_x while all points from $P_x(n/3)$ are $(n/3 - O(\log n))$-apart from $(n/3, n/3)$ (w.r.t. l_1-norm). Thus the string x is $(n/3, n/3 - O(\log n))$-strange.

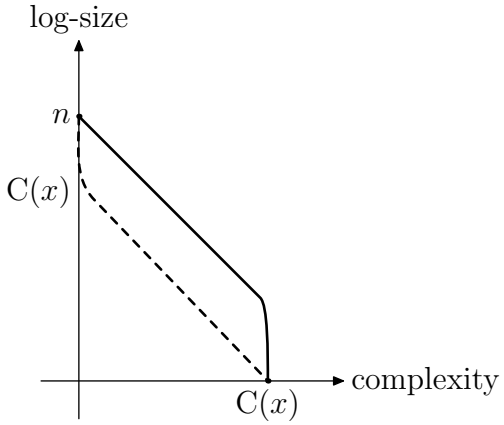

Figure 8: The sets P_x and $P_x(\varepsilon)$ for the strange string from Theorem 11. The set P_x is to the right of the dashed line. The set $P_x(\varepsilon)$ is to the right of the solid line.

Recall that we have introduced the notion of a strong model to separate good models from bad ones. Now we are able to present a theorem justifying this approach. Roughly speaking it states that there is a strong model A for a string x of length n such that parameters (complexity, log-cardinality) of every strong standard model B for x are $\Omega(n)$-far from those of A. The next theorem shows that there are strange strings.

Theorem 12 ([24]). *For all k there is a string x of length $n = 4k$ whose profile P_x is $O(\log n)$-close to the gray set shown on Fig. 9 and such that*

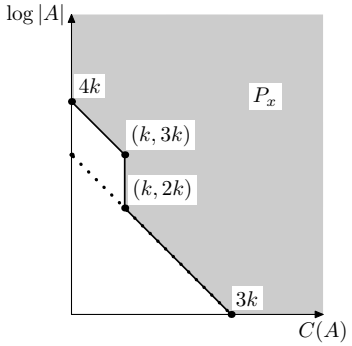

Figure 9: The profile P_x of a string x from Theorem 12.

- *there is an $O(\log n)$-strong model A for x with complexity $k + O(\log n)$ and log-cardinality $2k$ (that model witnesses the point $(k, 2k)$ on the border of P_x), but*

- *for all $m \geq \mathrm{C}(x)$ and for all simple lists of strings of complexity at most m the standard model B for x obtained from that list is either not strong for x or its parameters are far from the point $(k, 2k)$. More specifically, if B is an ε-strong model for x obtained from a list enumerated by a program q then $\mathrm{C}(q) + |\mathrm{C}(x) - k| + |\log|B| - 2k| + \varepsilon \geq \Omega(n)$.*

7.3 Properties of strong models

Once we have decided that non-strong descriptions are bad, it is natural to restrict ourselves to strong descriptions with negligible randomness deficiency (and hence negligible optimality deficiency).

Assume that A is an ε-strong description and the randomness deficiency of x in A is at most ε. Let i be the ordinal number if x in A w.r.t. some fixed order. Then $\mathrm{CT}(x|A,i) = O(1)$ and $\mathrm{CT}(A,i|x) \leq \varepsilon + O(1)$ (the latter holds, since $\mathrm{CT}(A|x) \leq \varepsilon$). As i is random and independent of A, the sets $Q_{A,i}$ and Q_A are $\varepsilon + O(\log n)$-close (Proposition 25). On the other hand, the sets $Q_{A,i}$ and Q_x are $\varepsilon + O(1)$-close by Proposition 31. Thus we obtain the first property of strong models:

Proposition 33. If both $\mathrm{CT}(A|x)$ and $\log|A| - \mathrm{C}(x|A)$ are at most ε then the sets Q_x and Q_A are $O(\varepsilon + \log l(x))$-close.

Assume that A is a strong model for x with negligible randomness deficiency. Assume that A is normal as well. Then the string x is normal as well. Indeed, for every pair $(i,j) \in P_x$ with $i \leq \mathrm{C}(A)$ the pair $(i, j - \log|A|)$ is in $P_{[A]}$ (Proposition 25) and hence there is a strong $(i * (j - \log|A|))$-description $\mathscr{B}$ for $[A]$. Consider the "lifting" of $\mathscr{B}$, that is, the union of all sets from $\mathscr{B}$ that have approximately the same size as A. It is a strong $(i * j)$-description for x. On the other hand, if $(i,j) \in P_x$ and $i \geq \mathrm{C}(A)$, then $i + j \geq \mathrm{C}(A) + \log|A| = \mathrm{C}(x)$. Hence the subset of A consisting of all strings x' whose ordinal number in A has the same $i - \mathrm{C}(A)$ leading bits as the ordinal number of x, is a strong $(i * j)$-description for x.

It turns out that for minimal models the converse is true as well. A model A for x is called $(\varepsilon, \varkappa)$-*minimal* if there is no model B for x with $\mathrm{C}(B) \leq \mathrm{C}(A) - \varepsilon$ and $\delta(x,B) \leq \delta(x,A) + \varkappa$.

Theorem 13 ([24]). *For some value $\varkappa = O(\log n)$ the following holds. Assume that A is an ε-sufficient statistic for an ε, ε-normal string x of length n. Assume also that A is a $(\delta, \varepsilon + \varkappa)$-minimal model for x. Then the code $[A]$ of A is $O((\delta + \varepsilon + \log n)\sqrt{n})$-normal.*

This theorem establishes the second interesting property of strong models. The third one states that conditional complexity of any strong, sufficient and minimal statistic for x conditioned by any other sufficient statistic for x is negligible.

Theorem 14 ([38]). *For some value $\varkappa = O(\log n)$ the following holds. Assume that A, B are ε-sufficient statistics for a string x of length n. Assume also that A is an ε-strong and a $(\delta, \varepsilon + \varkappa)$-minimal statistic for x. Then $\mathrm{CT}(A|B) = O(\varepsilon + \delta + \log n)$.*

This theorem can be interpreted as follows: assume that we have removed some noise from the given data string x by finding its description B with negligible optimality deficiency. Let A be any "ultimately denoised" model for x, i.e. a minimal model for x with negligible optimality deficiency. Then $\mathrm{C}(A|B)$ is negligible, as we have seen before. Hence to obtain the "ultimately denoised" model for x we do not need x: any such model can be obtained from B by a short program. Theorem 14 shows that any such *strong* model A can be obtained from B by a short program *in a reasonable time*.

References

[1] L.M. Adleman, *Time, space and randomness*. MIT report MIT/LCS/TM-131. March 1979.

[2] L. Antunes, B. Bauwens, A. Souto, A. Teixeira, *Sophistication vs. Logical Depth*, Theory of Computing Systems, First Online, 10.1007/s00224-016-9672-6

[3] L. Antunes and L. Fortnow. Sophistication revisited. *Theory of Computing Systems*, **45**(1), 150–161 (June 2009).

[4] L. Antunes, L. Fortnow, and D. van Melkebeek. Computational depth, *Proceedings of the 16th IEEE Conference on Computational Complexity*, 266–273. IEEE, New York, 2001. Journal version: Computational depth: Concept and applications, *Theoretical Computer Science*, **354**(3), 391–404 (2006)

[5] L. Antunes, A. Matos, A. Souto, P. Vitányi, Depth as Randomness Deficiency, *Theory of Computing Systems*, **45**(4), 724–739 (2009)

[6] B. Bauwens. *Computability in statistical hypotheses testing, and characterizations of independence and directed influences in time series using Kolmogorov complexity*, PhD thesis, University of Gent, May 2010.

[7] C.H. Bennett, Logical Depth and Physical Complexity, in *The Universal Turing Machine: a Half-Century Survey*. Edited by Rolf Herken, 227–257, Oxford University Press, 1988.

[8] L. Bienvenu, D. Desfontaines, A. Shen, What Percentage of Programs Halt? *Proceedings of ICALP 2015, 42nd International Colloquium, Kyoto, Japan, July 6-10, 2015*, Lecture notes in computer science, **9134**, 219–230. Extended version: Generic algorithms for halting problem and optimal machines revisited, `arXiv:1505.00731`.

[9] L. Bienvenu, P. Gács, M. Hoyrup, C. Rojas, A. Shen, Algorithmic tests and randomness with respect to a class of measures, *Proceedings of the Steklov Institute of Mathematics*, **274**, 34–89 (2011).

Russian version: Алгоритмические тесты и случайность относительно классов мер, *Труды математического института имени В.А.Стеклова*, **274**. 41–102 (2011).

[10] T. Cover, Kolmogorov complexity, data compression and inference. In: *The Impact of Processing Techniques on Communications*, ed. J.K. Skwirzynski. Martinus Nijhoff Publishers, 1985.

[11] P. Gács, On the relation between descriptional complexity and algorithmic probability, *Theoretical Computer Science*, **22**, 71–93 (1983).

[12] P. Gács, J. Tromp, P.M.B. Vitányi, Algorithmic statistics, *IEEE Transactions on Information Theory*, **47**(6), 2443–2463 (2001).

[13] A.N. Kolmogorov, Three Approaches to the Quantitative Definition of Information [Russian: Три подхода к определению понятия «количество информации»] *Problems of Information Transmission* [Проблемы передачи информации], **1**(1), 4–11 (1965). English translation published in: *International Journal of Computer Mathematics*, **2**, 157–168 (1968).

[14] A.N. Kolmogorov, Talk at the Information Theory Symposium in Tallinn, Estonia (then USSR), 1974. [As reported by Cover in his 1985 paper [10]]

[15] A.N. Kolmogorov, The complexity of algorithms and the objective definition of randomness. Summary of the talk presented April 16, 1974 at Moscow Mathematical Society. *Успехи математических наук* (Uspekhi matematicheskikh nauk, Russian), **29**(4[178]), 155 (1974). See `http://mi.mathnet.ru/rus/umn/v29/i4/p153`. A short note in Russian.

[16] A. Kolmogorov. Talk at the seminar at Moscow State University Mathematics Department (Logic Division), 26 November 1981. [The definition of (α, β)-stochasticity was defined in this talk, and the question about the fraction of non-stochastic objects was posed.]

[17] M. Koppel, Complexity, Depth and Sophistication, *Complex Systems*, **1**, 1087–1091 (1987).

[18] M. Koppel, Structure, in *The Universal Turing Machine: a Half-Century Survey*. Edited by Rolf Herken, 435–452, Oxford University press, 1988.

[19] M. Koppel, H. Atlan, An almost machine-independent theory of program-length complexity, sophistication, and induction, *Information Sciences*. **56**(1–3), 23–33 (1991).

[20] L. Levin, Randomness conservation inequalities; information and independence in mathematical theories, *Information and Control*, **61**(1), 15–37 (1984).

[21] M. Li, P.M.B. Vitányi, *An Introduction to Kolmogorov Complexity and its Applications*, 3rd ed., Springer, New York, 2008.

[22] L. Longpré, *Resource bounded Kolmogorov complexity, a link between computational complexity and information theory*. Ph. D. Thesis, Department of Computer Science, Cornell University, TR 86-776, 1986.

[23] A. Milovanov, Some properties of antistochastic strings. In: *Computer Science – Theory and Applications, 10th International Computer Science Symposium in Russia, Russia, July 13–17, 2015.* (CSR 2015), Lecture Notes in Computer Science, **9139**, 339–349.

[24] A. Milovanov, CSR 2016, to be published

[25] F. Mota, S. Aaronson, L. Antunes, A. Souto, Sophistication as Randomness Deficiency, *Descriptional Complexity of Formal Systems*, Lecture Notes in Computer Science, **8031**, 172–181 (2013)

[26] An.A. Muchnik, I. Mezhirov, A. Shen, N.K. Vereshchagin, *Game interpretation of Kolmogorov complexity*, `https://arxiv.org/abs/1003.4712`

[27] An.A. Muchnik, A. Romashchenko, Stability of properties of Kolmogorov complexity under relativization, *Problems of Information Transmission*, **46**(1), 38–61 (2010).

[28] An.A. Muchnik, A.L. Semenov, V.A. Uspensky, Mathematical metaphysics of randomness, *Theoretical Computer Science*, **207**(2), 263–317 (November 1998).

[29] An.A. Muchnik, A. Shen, M. Vyugin, *Game arguments in computability theory and algorithmic information theory*, `https://arxiv.org/pdf/1204.0198.pdf`.

[30] J. Rissanen, Modeling by shortest data description, *Automatica*, **14**, 465–471 (1978).

[31] C.P. Schnorr. Optimal enumerations and optimal Gödel numberings. *Mathematical Systems Theory*, 8(2):182–191, 1975.

[32] А. Шень. Понятие (α, β)-стохастичности по Колмогорову и его свойства. *Доклады Академии наук СССР*, **271**(6), 1337–1340 (1983).

English translation: A. Shen, The concept of (α, β)-stochasticity in the Kolmogorov sense, and its properties. *Soviet Math. Dokl.*, **28**(1), 295–299 (1983).

[33] A. Shen, Discussion on Kolmogorov complexity and statistical analysis, *The Computer Journal*, 42:4(1999), 340–342.

[34] A. Shen, Around Kolmogorov complexity: Basic Notions and Results, *Measures of Complexity. Festschrift for Alexey Chervonenkis*, Springer, 2015, 75–116. See also: `arXiv:1504.04955`.

[35] M. Sipser, A complexity theoretic approach to randomness, *Proceedings of the fifteenth annual ACM symposium on Theory of computing* (STOC), 1983, 330–335.

[36] R. Solomonoff, A formal theory of inductive inference. Part I, *Information and Control*, 7(1), 1–22 (1964)

[37] R. Solomonoff, A formal theory of inductive inference. Part II. Applications of the Systems to Various Problems in Induction, *Information and Control*, 7(2), 224–254 (1964)

[38] N. Vereshchagin, Algorithmic Minimal Sufficient Statistic Revisited In: *Mathematical Theory and Computational Practice, 5th Conference on Computability in Europe*, CiE 2009, Heidelberg, Germany, July 19–24, 2009. Proceedings. LNCS 5635.

[39] N. Vereshchagin. Algorithmic Minimal Sufficient Statistics: a New Approach. *Theory of Computing Systems*, **58**(3), 463–481 (2016).

[40] N. Vereshchagin, A. Shen, Algorithmic statistics revisited, *Measures of Complexity. Festschrift for Alexey Chervonenkis*, Springer, 2015, 2035–252. See also: `arXiv:1504.04950`.

[41] Н.К. Верещагин, В.А. Успенский, А. Шень. Колмогоровская сложность и алгоритмическая случайность. 576 с. Москва, МЦНМО, 2013. Electronic version: `http://www.lirmm.fr/~ashen/kolmbook.pdf`.

Draft English translation: `http://www.lirmm.fr/~ashen/kolmbook-eng.pdf`.

[42] N.K. Vereshchagin, P.M.B. Vitányi, Kolmogorov's structure functions and model selection, *IEEE Transactions on Information Theory*, **50**(12), 3265–3290 (2004).

[43] N.K. Vereshchagin, P.M.B. Vitányi. Rate Distortion and Denoising of Individual Data Using Kolmogorov Complexity. *IEEE Transactions on Information Theory*, v. 56(7), 2010, p. 3438–3454.

[44] P.M.B. Vitányi, *Meaningful information*, *IEEE Transactions on Information Theory*, **52**(10), 4617–4626 (2006). See also: `arXiv:cs/0111053`.

[45] V.V. V'yugin, On the defect of randomness of a finite object with respect to measures with given complexity bounds, *SIAM Theory Probab. Appl.*, **32**(3), 508–512 (1987).

[46] V.V. V'yugin, Algorithmic complexity and stochastic properties of finite binary sequences, *The Computer Journal*, **42**(4), 294–317 (1999).

[47] V.V. V'yugin. Does snooping help? *Theoretical Computer Science*, **276**(1), 407–415 (2002).

[48] C.S. Wallace, D.M. Boulton, An information measure for classification, *Computer Journal*, **11**(2), 185–194 (1968).