ELSEVIER Theoretical Computer Science 271 (2002) 15–35

Theoretical Computer Science

www.elsevier.com/locate/tcs

Kolmogorov entropy in the context of computability theory

Andrej A. Muchnik*, Semen Ye. Positselsky

Institute of New Technologies of Education (INT), Nizhnyaya Radishchevskaya Street 10, 109004 Moscow, Russia

Abstract

We consider the overgraph of the Kolmogorov entropy function and study whether it is a complete enumerable set with respect to different types of reductions. It turns out that (for any type of entropy) the overgraph of the conditional entropy function is m-complete, but the overgraph of the unconditional entropy function is not m-complete (and also not bT-complete). For tt-completeness, the situation is more subtle: the overgraph of the unconditional prefix entropy may be tt-complete or incomplete depending on the optimal programming system used in the definition of entropy. To prove these results we use the notion of r-separability and its effective version introduced in this article for the first time. © 2002 Elsevier Science B.V. All rights reserved.

Keywords: Kolmogorov entropy; m-completeness; tt-completeness; r-separability

Preface

In the 1960s, the notion of entropy of a finite object was introduced in the works of A. Kolmogorov and R. Solomonoff. An important contribution of this notion is constituted by formal mathematical explication of the concept of randomness (with respect to a probability distribution). In this paper, we do not consider distributions, nor other structures on finite objects (like the pairing function). We are interested in a purely algorithmic characterization of the entropy function. Let us mention that alternative definitions of entropy were given after the first one. They are systematized in [13]. If the opposite is not mentioned, our considerations will be applicable to any of these notions of entropy. To make our considerations complete and self-contained, we describe both new and known results.

An entropy function maps a constructive universe[1] (for example – the set of all finite binary strings) into the set of all positive integers. This function, denoted by $K(\cdot)$, is enumerable from above (or upper computable). This means that the overgraph

* Corresponding author. Fax: +7-095-915-6963.

[1] The reader can learn about constructive objects and constructive universes (aggregates) in [12].

$M=\{(x,n) \mid K(x)<n\}$ is enumerable. In our study of the set M we discovered several algorithmic properties and notions, which are interesting independent of the entropy context.

In Part I, we study these notions. We shall utilize several types of algorithmic reducibility. Among many reducibilities introduced in the theory of algorithms, we have chosen seven. The first is m-reducibility. Its role for computability is similar to that of homomorphism in algebra. We write $A \leqslant_m B$ if A is an inverse image of B under a total computable function. The other six reducibilities are transitive and closed under Boolean operations. (This means that $A \leqslant B$ and $B \leqslant C$ imply $A \leqslant C$, $A \leqslant C$ and $B \leqslant C$ imply $A \cup B \leqslant C$, $A \leqslant B$ implies $\bar{A} \leqslant B$.) Namely, we consider the Turing (or T-) reducibility, the weak truth-table (or w-) reducibility, the truth-table (or tt-) reducibility, and their bounded versions (bT, bw, btt). As usual, the Turing reducibility uses the oracle without restrictions. For w-reducibility, the oracle is used once and is asked several (finite number of) questions. For the truth-table reducibility, the reducing algorithm gives a result for any answers of the oracle, even those not corresponding to the set to which we reduce. The bounded reducibilities assume that the number of questions to the oracle is limited by a constant independent of inputs and of oracle's answers.

Statement. *By a compactness argument, any tt-reducing algorithm can be effectively transformed to an algorithm satisfying both the tt- and w-restrictions. Similarly, a btt-reducing algorithm can be effectively transformed to an algorithm satisfying both the btt- and bw-restrictions. In this transformation, if the original algorithm asks no more than c questions to the oracle, the new algorithm will ask no more than* 2^c *questions.*

Proof of this statement is straightforward.

Further on, discussing the tt- (btt-) reducing algorithms we will assume that they are also restricted by w (bw).

Clearly, the transitivity and Boolean-closure properties for all reducibilities are valid effectively. In the following diagram, the arrows are directed from stronger reducibilities to weaker ones:

$$\begin{tikzcd}
T & w \arrow[l] & tt \arrow[l] & \\
bT \arrow[u] & bw \arrow[l] \arrow[u] & btt \arrow[l] \arrow[u] & \\
 & & & m \arrow[ul]
\end{tikzcd}$$

Let us now outline the further content of the paper.

In Part I, we would like to mention a strengthening of Post's theorem on non-btt-completeness of simple sets. It is proved that simple sets are even non-bT-complete. This follows from the fact that the complete bT-degree contains exactly one btt-degree (not only among the enumerable sets).

Part II needs familiarity with the definitions of unconditional and conditional entropies (the simple and prefix ones). We study the conditional entropy $K(\cdot|\cdot)$ first. It is enumerable from above and we prove that the enumerable set $M=\{(x,y,n)\,|\,K(x|y)<n\}$ is m-complete. By a theorem of J. Myhill, all m-complete enumerable sets are recursively isomorphic. Hence, we get a final algorithmic characterization of conditional entropy. The situation with unconditional entropy is more complicated. Its overgraph is not bT-complete, but it is w-complete. It remains to resolve the problem of its tt-completeness. M. Kummer proved that the overgraph of *simple* entropy is tt-complete. Let us recall here that any entropy consists of a countable family of functions. The difference between any two members of the family is bounded. In the previous consideration all results were equally true for all members of any entropy family. Unexpectedly, it is not so now: there are functions of *prefix* entropy for which the overgraph is tt-complete and such functions for which the overgraph is not tt-complete.

In Part II, we also prove an interesting inequality: [2]

$$\forall d\ \exists x,y\ (KS(x)+d\ <\ KS(y)\wedge KP(y)+d\ <\ KP(x)).$$

So, "up to an additive logarithm" the simple and prefix entropies are the same, but on closer look they are very different.

Main results of this paper were reported at Kolmogorov Seminar of Moscow State University in the fall of 1998. [3]

Part 1

In 1956 Albert Muchnik introduced the following notion [8].

Definition 1.1. An enumerable set A is called r-separable if for any enumerable set B such that $A\cap B=\emptyset$ there exists a decidable set C that separates A from B (that is $A\subset C$ and $B\cap C=\emptyset$).

Obviously, all decidable sets and all simple sets are r-separable. M. Kummer and F. Stephan proved that the enumerable frequency decidable [4] sets are r-separable [5]. For any function f consider the set $\{(x,n)\,|\,f(x)<n\}$; we will call it the *overgraph* of f. The overgraph of an entropy function is also r-separable. It is interesting to note that the very rich class of frequency decidable sets does not contain the overgraph of an entropy function (M. Kummer's theorem [4]). On the other hand, no m-complete enumerable set is r-separable (this follows from two facts: all m-complete enumerable

[2] Here KS denotes the simple entropy and KP denotes the prefix entropy. It is well known that $KS(z)-O(1)<KP(z)<KS(z)+O(\log(KS(z)))$.

[3] The results of the first part were announced in [10].

[4] A set G is called frequency decidable with parameter m if there is a computable function α such that for each set H of m elements $\alpha(H)$ is a function from H to $\{0,1\}$ which differs from the restriction of the characteristic function of G.

sets are recursively isomorphic to the universal set and there exists a nonseparable pair of enumerable sets).

In the general theory of algorithms, many notions find their effective analogs (often in various ways).

Definition 1.2 (*S. Positselsky*). An r-separable set A is called effectively r-separable if there exists the following algorithm γ. For any enumerable set B not intersecting A, the algorithm γ terminates on the text of any program β enumerating B. In addition, $\gamma(\beta)$ is the text of the program for recognition of some set C that separates A from B. (The program for recognition of a set outputs 1 on the elements of this set and 0 on the other elements.)

It is clear that all decidable sets are effectively r-separable.

Theorem 1.1 (S. Positselsky). *All effectively r-separable sets are decidable.*

Proof. Here and in the sequel we will construct enumerable sets using in the construction the text of a program enumerating the set that we are constructing. To avoid a vicious circle, we must construct the set corresponding to a program enumerating another set. Then we make the two sets coincide, using the fixed point theorem of S. Kleene. The same method is used to construct computable functions (partially defined).

Let A be an effectively r-separable set. Suppose an algorithm γ ensures this effectiveness. For each n let us construct the set B_n, which is enumerated by the program β_n. If γ is defined on β_n, let us run the program $\gamma(\beta_n)$ on the input n. If the output of $\gamma(\beta_n)$ on the input n is equal to 1, then $B_n = \{n\}$. If either γ is not defined on β_n, or the output of the program $\gamma(\beta_n)$ on the input n is not defined or is not equal to 1, then $B_n = \emptyset$.

We claim that $n \notin A \Leftrightarrow [\gamma(\beta_n)](n) = 0$; therefore, the complement of the set A is enumerable.

Let us prove "$\Rightarrow$". If $n \notin A$, then B_n in any case does not intersect A. Therefore, the algorithm γ is defined on β_n and the program $\gamma(\beta_n)$ is defined on any input. If $[\gamma(\beta_n)](n) = 1$, then $B_n = \{n\}$. But the set recognized by $\gamma(\beta_n)$ separates A from B_n. We see that $[\gamma(\beta_n)](n) = 0$.

Let us prove "$\Leftarrow$". If $[\gamma(\beta_n)](n) = 0$, then $B_n = \emptyset$. It follows from the definition of γ that $A \subset \{x \mid [\gamma(\beta_n)](x) = 1\}$. We see that $n \notin A$. $\square$

More fruitful is the following weak effectivization of r-separability.

Definition 1.3 (*S. Positselsky*). An r-separable set A is called resilient if there exists the following algorithm γ. If B is an enumerable set not intersecting A, this algorithm γ is defined on the text of any program β enumerating B. The output $\gamma(\beta)$ is the text of a program enumerating a decidable set C which separates A from B.

The only difference between resilience and effective r-separability is that for the former, though the separating set C remains decidable, one finds only an enumeration (and not recognition) of C starting from an enumeration of B.

It is clear that all decidable sets and all strongly effectively simple[5] sets are resilient. In the second part of this paper, we will show that the overgraph of an entropy function is resilient.

The classes of r-separable and resilient sets have a good property of being lattices (just as the classes of decidable, enumerable, frequency decidable sets).

Theorem 1.2 (S. Positselsky). *The class of r-separable sets is closed under the operations of union and intersection.*

The class of resilient sets is effectively closed under the operations of union and intersection.

In the second statement, the effectiveness means that starting from algorithms enumerating two sets A_1 and A_2 and algorithms ensuring their resilience, one effectively constructs algorithms ensuring the enumerability and resilience of $A_1 \cup A_2$ and $A_1 \cap A_2$.

Proof. The proofs of these two statements are very similar. Let A_1 and A_2 be r-separable (or, respectively, resilient) sets.

Let us prove that the set $A_1 \cup A_2$ is r-separable (resilient). Let B be an enumerable set not intersecting $A_1 \cup A_2$. Then B is separated from A_1 and B is separated from A_2. We can find enumeration programs for two decidable sets C_1 and C_2 such that $A_1 \subset C_1$, $A_2 \subset C_2$, $B \cap C_1 = \emptyset$, $B \cap C_2 = \emptyset$. Then the sets $A_1 \cup A_2$ and B are separated by the decidable set $C_1 \cup C_2$ (whose enumeration program is known).

Let us prove that the set $A_1 \cap A_2$ is r-separable (resilient). Let B be an enumerable set which does not intersect $A_1 \cap A_2$. Since $A_1 \cap (A_2 \cap B) = \emptyset$, the set A_1 is separated from the enumerable set $A_2 \cap B$. We can find an enumeration program for a decidable set C such that $A_1 \subset C$ and $(A_2 \cap B) \cap C = \emptyset$. Since $A_2 \cap (B \cap C) = \emptyset$, the set A_2 is separated from the enumerable set $B \cap C$ (whose enumeration program we know). Therefore, we can find an enumeration program for a decidable set D such that $A_2 \subset D$ and $(B \cap C) \cap D = \emptyset$. Now, we see that $A_1 \cap A_2 \subset C \cap D$ and $B \cap (C \cap D) = \emptyset$. Thus, the decidable set $C \cap D$ separates $A_1 \cap A_2$ from B (and we know an enumeration program of $C \cap D$). □

It is obvious that the class of r-separable (resilient) sets is (effectively) closed under cylindrification. It follows that this class is also (effectively) closed under many other operations. For example, the Cartesian product of two sets is an intersection of two cylinders.

[5] An enumerable set A is called strongly effectively simple if given a program enumerating a set B not intersecting A one can effectively construct a number larger than $\max(B)$. The simple set of Post [11] is strongly effectively simple.

We turn now to the questions of completeness with respect to the various reducibilities. We need the following result, which is due to Lachlan [6]. If the set $A \cap B$ is m-complete and the set A is enumerable, then either A or B is m-complete. Here we formulate and give a new proof of an effectivization of this statement.

Theorem 1.3 (A. Lachlan). *Let U be an enumerable m-complete set. Then there exists an algorithm that given a program enumerating a set A constructs a program computing a function f such that $f(U) \subset U$, $f(\bar{U}) \subset A \setminus U$, and if the set $U \cup A$ is not m-complete, then f is a total function.*

Proof. Using the fixed point theorem, let us construct an auxiliary function $\lambda xyz.g_{x,y}(z)$. Having a program computing this function and two elements x and y, we find the program computing $h = \lambda z.g_{x,y}(z)$. It is known that given a program h one can effectively find an input v such that if $h(v)$ terminates, then $v \in U \Leftrightarrow h(v) \in U$.[6] Running the enumerations of the sets U and A, we wait until y is caught in U or v is caught in $U \cup A$. If neither of the two events ever happens, then $g_{x,y}(z)$ is undefined for all z. If it is first revealed that $y \in U$, then $\forall z\ g_{x,y}(z) = y$. If it is first revealed that $v \in U \cup A$, then $\forall z\ g_{x,y}(z) = x$.

Let us construct the function $\lambda x.f(x)$. Given an input x, we look over all the values of y and find a value such that for the corresponding v it is true that $g_{x,y}(v) = x$. Then we put $f(x) = v$.

Let us prove that the function f is well defined. We will show that if for some x_0 the value of $f(x_0)$ is not defined, then the set $U \cup A$ is m-complete.

Construct the function $\lambda y p(y)$ which m-reduces the set U to the set $U \cup A$. Let $p(y)$ be equal to v that corresponds to the pair x_0, y. If $y \in U$, then $\forall x,z\ g_{x,y}(z)$ is defined. It follows that $g_{x_0,y}(v) = y$. This implies that $v \in U \Leftrightarrow y \in U$. That is, $p(y) \in U$. If $y \notin U$, then $v \notin U \cup A$; otherwise, we have $g_{x_0,y}(v) = x_0$ and $f(x_0)$ is defined. Thus, $y \in U \Leftrightarrow p(y) \in U \cup A$.

Suppose that the function f is defined on an input x. This means that $\exists y\ g_{x,y}(f(x)) = x$. It follows that $f(x) \in U \cup A$ and $f(x) \in U \Leftrightarrow x \in U$. This proves the desired property of f. □

Let us illustrate this proof by the next three pictures (see Figs. 1–3).

The next theorem is a stronger version of Kobzev's result [3] about non-*btt*-completeness of *r*-separable sets. However, in our case the non-bT-completeness cannot be proved in the same way. Kobzev's reasoning is based on the following fact: if an enumerable set A can be *btt*-reduced to an *r*-separable set, then A is *r*-separable. Indeed, we will show that there exists an enumerable non-*r*-separable set A which can be *bw*-reduced to an *r*-separable set.

Theorem 1.4 (S. Positselsky). *No r-separable set is bT-complete.*

[6] For the universal set U this fact immediately follows from the fixed-point theorem.

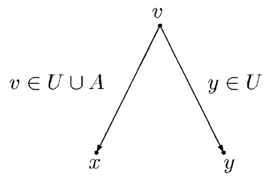

Fig. 1. Function $g_{x,y}$.

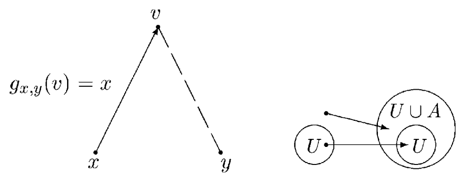

Fig. 2. Function $f(x)$.

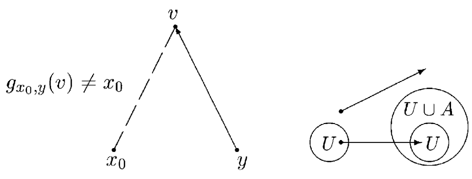

Fig. 3. Function $p(y)$.

Proof. Let B be an r-separable set and U be an enumerable m-complete set. The proof is by reductio ad absurdum. Suppose $U \leqslant_{bT} B$ and γ is an algorithm which bT-reduces U to B and has the minimal possible bound on the number of questions to the oracle. If this bound is equal to zero, then the set U is decidable. If the bound is equal to $n+1$, let us construct a *new* algorithm. The new algorithm will reduce U to B and for any input it will ask the oracle no more than n questions.

For each program δ enumerating the set D_δ consider the set A_δ of all the inputs y on which the algorithm γ either does not ask the oracle any questions, or the first question asked belongs to D_δ. It is clear that A_δ is enumerable uniformly in δ. Let f_δ be the function given by the construction of Theorem 1.3 applied to the sets U and A_δ. Denote by $q(y)$ the first question that the algorithm γ poses to the oracle on the input y.

Suppose the new algorithm has received an input x. Consider the set C of all elements of the form $q(f_\delta(x))$, where x is fixed and δ is changing (notice that q and f_δ are partial functions). It is clear that C is enumerable uniformly in x.

Suppose that $B \cap C = \emptyset$; then for some ε the set D_ε is decidable, B is contained in D_ε, and C does not intersect D_ε (due to r-separability of B). The set $U \cup A_\varepsilon$ can be reduced to the set B in a way that requires not more than n questions to the oracle on each input. Indeed, if the algorithm γ asks the oracle no questions on the input y, then $y \in A_\varepsilon$. If $q(y) \in D_\varepsilon$, then $y \in A_\varepsilon$. If $q(y) \notin D_\varepsilon$, then $y \notin A_\varepsilon$. Now, we can use the algorithm γ to find out whether y belongs to U without asking oracle the first question (the answer is already known: $q(y) \notin D_\varepsilon \Rightarrow q(y) \notin B$). One can find out whether $q(y)$ belongs to D_ε, because this set is decidable.

If for some x the set $U \cup A_\varepsilon$ is m-complete, the new algorithm m-reduces the set U to the set $U \cup A_\varepsilon$ and then bT-reduces the set $U \cup A_\varepsilon$ to the set B, as described.

If for each x the set $U \cup A_\varepsilon$ is not m-complete, then by Theorem 1.3 the value $f_\varepsilon(x)$ is defined, $f_\varepsilon(x) \in U \cup A_\varepsilon$, and $x \in U \Leftrightarrow f_\varepsilon(x) \in U$. If $q(f_\varepsilon(x))$ is defined, then $q(f_\varepsilon(x)) \in C$, hence $q(f_\varepsilon(x)) \notin D_\varepsilon$ and $f_\varepsilon(x) \notin A_\varepsilon$, hence $f_\varepsilon(x) \in U$ and $x \in U$.

We see that there at least one of the next three conditions holds: ($B \cap C \neq \emptyset$), or ($f_\varepsilon(x)$ is defined, but $q(f_\varepsilon(x))$ is not defined), or ($x \in U$). Let us return to the construction of the new algorithm. Given an input x, it waits until any of the following three events happens: a program δ is found such that $q(f_\delta(x)) \in B$; a program ε is found such that $f_\varepsilon(x)$ is defined, but $q(f_\varepsilon(x))$ is not; or x is caught in the enumeration of U. In the first case, the new algorithm emulates the work of the algorithm γ on the input $f_\delta(x)$ without asking the oracle the first question (since the answer is known). In the second case, the new algorithm emulates the work of the algorithm γ on the input $f_\varepsilon(x)$ without asking the oracle any questions. In the third case, we actually know that $x \in U$. Recall that if $f(x)$ is defined, then $x \in U \Leftrightarrow f(x) \in U$. □

Theorem 1.5 (S. Positselsky). *There exists a nonseparable pair of enumerable sets such that each of them is bw-reducible to a resilient set.*

Proof. Let A be a nondecidable resilient set. By the decomposition theorem [2, 9] there exist enumerable sets B_1, B_2, for which the following holds. First, $A = B_1 \cup B_2$; second, $B_1 \cap B_2 = \emptyset$; third, B_1 and B_2 are not separable by a decidable set. An algorithm bw-reducing B_i to A works as follows. Given an input x, if the oracle answers "$x \notin A$", the output of the algorithm is "$x \notin B_i$". If the oracle answers "$x \in A$", we enumerate the sets B_1 and B_2 until x is caught in one of them. In the latter case, we use the assumption that the oracle's answer is correct (hence it is not a tt-reducing algorithm). □

Lachlan proved [7] that for an enumerable set B the properties of bw-completeness and btt-completeness are equivalent. Here we will prove a stronger result, replacing bw-completeness by bT-completeness and eliminating the enumerability requirement. Theorem 1.4 can be deduced from Theorem 1.6 and the preservation property of r-separability with respect to btt-reducibility, which was mentioned earlier. Nevertheless, we give an independent proof, which will be useful in the second part of this paper (see the remark after Theorem 2.3).

Theorem 1.6 (An. Muchnik). *If an enumerable m-complete set U is bT-reducible to a set B, then U is also btt-reducible to B.*

Lemma. *Let U be an enumerable m-complete set, $A_0 \supset A_1 \supset \cdots \supset A_k$ be enumerable sets, and $\overline{A_0} = A_k = \emptyset$. Then there exists a total computable function r and a number i such that $r(U) \subset U$ and $r(\bar{U}) \subset A_i \backslash (A_{i+1} \cup U)$.*

Proof of Lemma. For the identity function q_0 we have $q_0(U) \subset U$ and $q_0(\bar{U}) \subset A_0 \backslash U$. Let us argue by induction on i. Given a total computable function q_i for which $q_i(U) \subset U$ and $q_i(\bar{U}) \subset A_i \backslash U$, we will construct either an analogous function q_{i+1}, or a function r required in the Lemma. Because no function q_k exists for which $q_k(\bar{U}) \subset A_k \backslash U$, at a certain step i a function r will be constructed. Let s be a computable injective function whose image is the enumerable set $U \cup A_i$. The preimage of U under s is an enumerable m-complete set, since the composition of functions $s^{-1} q_i = \lambda x. s^{-1}(q_i(x))$ m-reduces U to the set $s^{-1}(U)$. According to the proof of Theorem 1.3, for the enumerable set $s^{-1}(A_{i+1})$ there exists either a total computable function f for which $f(s^{-1}(U)) \subset s^{-1}(U)$ and $f(\overline{s^{-1}(U)}) \subset s^{-1}(A_{i+1}) \backslash s^{-1}(U)$, or a total computable function p for which $p(s^{-1}(U)) \subset s^{-1}(U)$ and $p(\overline{s^{-1}(U)}) \subset \overline{s^{-1}(A_{i+1}) \cup s^{-1}(U)}$. In the first case, we can put $q_{i+1} = s f s^{-1} q_i$; then $q_{i+1}(U) \subset U$ and $q_{i+1}(\bar{U}) \subset A_{i+1} \backslash U$. In the second case, we set $r = sps^{-1} q_i$. $\square$

Proof of Theorem. Assume that U is reduced to B by a bT-algorithm γ posing no more than n questions to the oracle. We will construct a btt-algorithm δ, reducing U to B and asking the oracle less than 2^n questions. In addition, the construction of δ will not depend on B.

Let us assign to each input y of the algorithm γ a set $D(y)$ of binary charts $\langle d_1, d_2, \ldots, d_j, u \rangle$. A chart belongs to the set $D(y)$ if the algorithm γ gives the output u on the input y, provided that it receives $d_1, \ldots, d_j$ as the oracle's answers to the algorithm's questions (no matter whether the oracle's answers correspond to the set B). It is clear that $|D(y)| \leqslant 2^n$ for each y. For each $0 \leqslant i \leqslant 2^n + 1$ we set $A_i = \{y \mid |D(y)| \geqslant i\}$. It is obvious that sets A_i satisfy the conditions of the Lemma. Let r be a total computable function for which $r(U) \subset U$ and $r(\bar{U}) \subset A_i \backslash (A_{i+1} \cup U)$.

Given an input z, the algorithm δ enumerates the sets U and $D(r(z))$ simultaneously. We know that $z \in U$ or $|D(r(z))| = i$. If z is caught in U, then the output of δ is known. Suppose that i elements have been caught in the enumeration of $D(r(z))$ (denote the set of those elements by E). The algorithm δ asks the oracle at once all the questions which would be posed by the algorithm γ on the input $r(z)$, assuming that the latter algorithm receives the answers from E. If the oracle's answers turn out to be compatible with one of the charts from E, then the output of δ is equal to the output of γ mentioned in this chart (there can be no more than one such chart). If the oracle's answers turn out incompatible with every chart from E, then the output of δ is "$z \in U$". In the first case, the algorithm δ is correct, because the algorithm γ is. In the second case, if the oracle's answers correspond to the set B and the algorithm γ is correct, then a chart which does not belong to E will eventually appear in $D(r(z))$. The latter is incompatible with "$z \notin U$". $\square$

The next result is symmetric to the previous one.

Theorem 1.7 (An. Muchnik). *If a set B is bT-reducible to an enumerable m-complete set U, then B is also btt-reducible to U.*

Proof. Assume that B is reduced to U by a bT-algorithm γ posing no more than n questions to the oracle. We will construct a btt-algorithm δ, reducing B to U and asking the oracle less than 2^{n+2} questions. First, let us modify the algorithm γ so that it would satisfy the following *requirement*. If the oracle answered "yes" to the question "$x \in U$?", the enumeration of U is started; after x is caught in U, the usual work of γ resumes. That is, if the true answer to the question is "no", then the modified algorithm will never stop. Obviously, the modified algorithm γ still bT-reduces B to U.

Now let us use the sets $D(y)$ defined in the proof of the previous theorem. Since the set $\{(y,z) \mid z \in D(y)\}$ is enumerable, it is m-reducible to U (say, by a function f). Given an input y, the algorithm δ asks the oracle about all the elements $f(y,z)$, where z is a binary chart of length no more than $n+1$. If the oracle answers correctly, we will know the set $D(y)$.

Due to the requirement imposed on the algorithm γ, the set of true oracle's answers to the questions of γ and the output of γ on the input y can be found as follows. Let us define by induction a sequence of sets $D_0 \supset D_1 \supset D_2 \ldots$. Put $D_0 = D(y)$. In all the charts from the set D_j the first j digits will correspond to the true oracle's answers to the first j questions of γ on the input y. If D_j contains a chart of length $j+1$, then this chart is true. Otherwise, let us define D_{j+1}. If D_j contains some charts in which the $(j+1)$th digit means "yes", then D_{j+1} consists of all such charts. Otherwise $D_{j+1} = D_j$.

This construction gives some output even when the oracle answers the questions of δ incorrectly. $\square$

The following answer to a question of G. Kobzev is a corollary of Theorems 1.6 and 1.7.

Corollary. *The complete bT-degree contains exactly one btt-degree.*

Theorem 1.8 (S. Positselsky). *Any resilient set is either decidable, or T-complete.*

This theorem will follow from Theorem 1.9, which was proven in [1].

Definition 1.4 (*M. Blum, I. Marques*). An enumerable set A is called subcreative if there exists an algorithm γ with the following properties. If B is an enumerable set not intersecting A, the algorithm γ is defined on the text of any program β enumerating B. The result $\gamma(\beta)$ is the text of a program enumerating a set C such that $A \subsetneq C$ and $B \cap C = \emptyset$.

Obviously, if a nondecidable set is resilient, then it is subcreative.

Theorem 1.9 (M. Blum, I. Marques). *Any subcreative set is T-complete.*

Part 2

Theorem 2.1 (An. Muchnik). *The overgraph of any conditional entropy function is an m-complete set.*

Proof. We will use an idea from the proof of M. Kummer's theorem on the unconditional simple entropy.

Let U be an enumerable set. We would like to construct an algorithm m-reducing U to the set $M=\{(x,y,n)\mid K(x\mid y)<n\}$, where $K(x\mid y)$ denotes the entropy of x conditional to y.[7] We will need the following auxiliary construction. It depends on a natural number d considered as a free parameter.

To any condition y we assign a *scale*. Any scale has a *pointer* and 2^d *points*, numbered by the integers between 1 and 2^d. During the time when the construction is performed, the number of the point under the pointer will never decrease. Some points will be *marked* with a pair of natural numbers each, where the first component of the pair coincides with the number of the point. No point is marked more than once, and no pair of numbers marks more than one point. Some pointers will be *tied* to their positions at certain moments of time and do not move thereafter.

Let us fix certain enumerations of the sets U and M. Denote by U^t and M^t the subsets enumerated in the first t steps. The construction consists of a sequence of finite stages. At the first stage, all the pointers are on the lowest points of their scales (numbered by 1) and no point is marked. Let us describe the stage number t.

For each $v\in U^t$ all the pointers placed over the points marked by pairs of the form (w,v) are tied to those points. Let us process one by one the first t scales whose pointers are not tied. Consider the scale assigned to the condition y. We move its pointer to the minimal point x for which $(x,y,d)\notin M^t$. Such an x does exist, because $\forall y,n\ |\{x\mid (x,y,n)\in M\}|<2^n$. If the new position of the pointer differs from the previous one, let us take the minimal z for which the pair (x,z) has not been used as a mark yet. We mark the xth point on the yth scale by the pair (x,z). This completes the description of the auxiliary construction.

Assume that the parameter d is large enough. Let us prove that

$$\begin{array}{l}\text{if the pointer of the } y\text{th scale is tied}\\ \text{to the point number } x, \text{ then } (x,y,d)\in M.\end{array} \qquad (\alpha)$$

Indeed, given y and d, we start the construction with the parameter d and wait until the pointer of the yth scale is tied, x is the number of the point under this pointer. Then we have $K(x\mid\langle y,d\rangle)<c$, where c is a constant. Therefore, $K(x\mid y)<c'\log d<d$ (where c' is a constant and d is large enough).

[7] There exist more than five natural definitions of conditional entropy (either being or not being monotone with respect to the condition). This theorem holds for all of them. For the proof to be correct in all cases one should take x and y only of the form $0\ldots01$.

Now, let us fix a sufficiently large value for d. Let b be the maximal point number which is reached by infinitely many pointers in the process of the auxiliary construction. The algorithm m-reducing U to M, works as follows.

Given an input z, let us find the condition y_z for which the bth point of the scale assigned to y_z is marked by (b,z). Such a condition exists due to our choice of b. We claim that $z \in U$ if and only if $(b, y_z, d) \in M$ for all z except a finite set of the elements z for which the pointer of the scale assigned to y_z stops above b. The proof immediately follows from the description of the auxiliary construction, from (α), and the choice of b. $\square$

Note that the one-dimensional section $\{y \mid (b,y,d) \in M\}$ of the three-dimensional set M is already m-complete.

From now on, we will consider the unconditional entropy only.

Theorem 2.2 (S. Positselsky). *The overgraph of any entropy function is a resilient set.*

Proof. Put $M = \{(x,n) \mid K(x) < n\}$ and let B be an enumerable set not intersecting M. We claim that the second component of the pairs from B is bounded, and the bound can be effectively found starting from a program enumerating B.

Given n, consider the first pair of the form (x,n) caught in the enumeration of B. If such a pair exists, then $K(x) < C \log n$, where the factor C effectively depends on a program enumerating B. For large n we have $C \log n < n$ and therefore $K(x) < n$. The latter means that (x,n) belongs to M, which contradicts our assumption that M and B do not intersect.

Let d be the bound found in the previous paragraph. Then it is easy to write a program enumerating the set $D = M \cup \{(x,n) \mid n > d\}$. We know that M is contained in D and $B \cap D$ is empty. It remains to prove that D is decidable. Indeed, $D = (M \cap \{(x,n) \mid n \leqslant d\}) \cup \{(x,n) \mid n > d\}$. The second term of this union is obviously decidable. The first term is finite for the simple and the prefix entropy. For the cases of the decision entropy, the a priori entropy, and the monotone entropy it suffices to show that for any n the set $E = \{x \mid K(x) < n\}$ is decidable. The three mentioned entropies are defined on binary words. It follows easily from their definitions that the set E contains all initial subwords of any of its words. Besides, any subset of the set E such that none of its elements are initial subwords of one another contains no more than 2^n elements. Hence, there are not more than 2^n infinite strings having the property that all of their initial subwords belong to E. Let us denote those strings by $y_1, \dots, y_i \dots$, and the set of all their initial subwords by F. For any word $z \in E \backslash F$, let us define an initial subword called the *trunk* of z. It is the shortest initial subword of z which does not belong to F. It is clear that all the trunks belong to E and they are not one another's initial subwords. Thus, there are not more than 2^n trunks. All the words with the same trunk form a tree without infinite branches, which is therefore finite. Hence, the set $E \backslash F$ is finite. Each y_i is computable, since E is enumerable and for any long enough w which is an initial subword of y_i, exactly one of the two words $w0$ and $w1$ belongs to E. $\square$

Theorem 2.3 (An. Muchnik). *The overgraph of any entropy function is not bT-complete.*

Proof. This follows from Theorems 1.4 and 2.2. $\square$

An argument analogous to the proof of Theorem 1.4 shows that the universal enumerable set is not bT-reducible to the overgraphs of the entropy functions, even if those functions were relativized by any oracle.

The idea of the next proof was invented nearly before the notion of the entropy itself. Since similar arguments were suggested by many people independently, we do not indicate who the author is.

Theorem 2.4. *The overgraph of any entropy function is w-complete.*

Proof. We would like to construct an algorithm w-reducing an enumerable set U to the set $M = \{(x,n) \mid K(x) < n\}$. Let us fix some enumerations of U and M. The algorithm has to find out whether $y \in U$. Let d be the length of the binary representation of y.

Using the oracle, we find the values of K on all the binary words of length d^2. Since these values are bounded by $\text{const}\, d^2$, all the questions to the oracle can be posed simultaneously. For each word z of length d^2 such that $K(z) < d^2$, let us find the number of steps $t(z)$ in which the pair (z, d^2) will be caught in the enumeration of M. Here we presume that the oracle gave the correct answers; otherwise, our algorithm may never stop! Let us denote by s the maximal value of $t(z)$ on the words z of length d^2 on which t is defined. We claim that if d is large enough, then y is caught in the first s steps of the enumeration of U whenever y belongs to U. Indeed, assume the contrary. We have $y \in U$. Let r denote the number of steps in which y gets caught in U. Then we have $s < r$. Consider the set V of all the words z of length d^2 for which the pair (z, d^2) gets caught in the enumeration of M in the first r steps. Since $|V| < 2^{d^2}$, there is a word of length d^2 which does not belong to V. If w is the first such word, then it is sufficient to know y in order to find w. Hence, we have $K(w) < K(y) + \text{const} < \text{const}\, d$. The rightmost term of the inequality is smaller than d^2 for large d. That is a contradiction.

It remains to “repair” the constructed algorithm on a finite number of inputs of small lengths to make it correct. $\square$

Theorem 2.5 (M. Kummer). *The overgraph of any simple entropy function is tt-complete.*

Proof. The argument is parallel to our proof of Theorem 2.1.

Let U be an enumerable set. We would like to construct an algorithm tt-reducing U to the set $M = \{(x,n) \mid KS(x) < n\}$, where KS is the simple entropy. Let us describe an auxiliary construction with a parameter d.

This construction differs from the one introduced for the proof of Theorem 2.1 in the following. In the previous construction, the scales were assigned to conditions; now they are numbered by positive integers expressing the lengths of binary representations.[8] On the stage number t the pointer of the yth scale is moved to the point x, which is defined below (if this pointer is not yet tied). We take the minimal x for which there is $p \leqslant x2^{y-d}$ such that $(p,y) \notin M^t$. Such a value of x exists, because $|\{p \mid (p,y) \in M\}| < 2^y = 2^d 2^{y-d}$.

The claim (α) from the proof of Theorem 2.1 is modified as follows:

$$\begin{array}{l} \text{if the pointer of the } y\text{th scale is tied to the point } x, \text{ then} \\ \text{the entropy of any number from the semi-interval} \\ ((x-1)2^{y-d}, x2^{y-d}] \text{ is smaller than } y - d/2. \end{array} \qquad (\alpha)$$

Let us prove that (α) is true for d large enough. We want to construct a number q from the semi-interval $((x-1)2^{y-d}, x2^{y-d}]$. In order to do that, it suffices to know the value of d and the binary representation of the number $(x2^{y-d} - q)$, completed by zeroes to the left so that there are exactly $(y-d)$ bits. Let us denote this binary word by v. Further, let $\partial(v)$ denote the length of v and $\hat{v}$ be the number represented by v. We can recover y as $d + \partial(v)$. Let us start the auxiliary construction with the parameter d and wait until the pointer of the yth scale is tied to some point. If x is the number of this point, then $q = x2^{y-d} - \hat{v}$. Now let us estimate the entropy of q:

$$\begin{aligned} KS(q) &< KS(\langle d, v\rangle) + \text{const} <^{9} \text{const} \log d + \partial(v) \\ &= \text{const} \log d + y - d < y - d/2. \end{aligned}$$

The reducing algorithm chooses b and given an input z finds y_z exactly as in Theorem 2.1. We claim that for all z but a finite set of exceptions the following implications hold (here we write y instead of y_z):

$$z \in U \Rightarrow \forall p \in ((b-1)2^{y-d}, b2^{y-d}] \quad KS(p) < y - d/2,$$

$$z \notin U \Rightarrow \exists p \in ((b-1)2^{y-d}, b2^{y-d}] \quad KS(p) \geqslant y.$$

As in Theorem 2.1, these implications follow straightforwardly from the description of the auxiliary construction, the statement (α), and the choice of b.

Note that these implications allow to answer the question “Does z belong to U?” even when the oracle’s answers are incorrect. Of course, if the oracle answers incorrectly, then the reducing algorithm can answer incorrectly, as well. □

[8] This distinction is, of course, informal.

[9] It is for this inequality that we need the entropy to be *simple*.

Unlike in the previous theorem, in the following theorem we deal with one specific entropy function. On the other hand, our considerations do not depend on the entropy type (it is correct for the simple entropy as well as for the prefix, monotone, decision, or a priori entropy).

Theorem 2.6 (An. Muchnik). *There exists an entropy function with a tt-complete overgraph.*

Proof. Let U be an enumerable set and K be any entropy function. Let us modify the function K to get another entropy function K'. If $z = 0^x 1v$ and $x \notin U$, then $K'(z)$ equals the number $K(z) + 2$ if the latter is even, and $K(z) + 3$ otherwise. If $z = 0^y$ or $z = 0^x 1v$ and $x \in U$, then $K'(z)$ is equal to the number $K(z) + 1$ if it is odd, and $K(z) + 2$ otherwise.

If K is a monotone entropy function then it is determined by an encoding function ϕ. Then define the encoding function ϕ' for the entropy function K' as follows.

For all z

- if a string w of an odd length is a ϕ-description of z then $001w$ is a ϕ'-description of z,
- if a string w of an even length is a ϕ-description of z then $01w$ is a ϕ'-description of z.

Additionally, for $z = 0^y$ and for $z = 0^x 1v$, where $x \in U$

- if a string w of an odd length is a ϕ-description of z then $01w$ is a ϕ'-description of z,
- if a string w of an even length is a ϕ-description of z then $1w$ is a ϕ'-description of z.

Finally, if a string w is a ϕ'-description of z then all strings ww' are also ϕ'-descriptions of z.

It is easy to check that K' is an entropy function (of the same type as the function K).

Let us construct an algorithm *tt*-reducing the set U to the overgraph of K'. We fix large enough c and for each $p < cx$ ask the oracle whether the pair $(0^x 1, p)$ belongs to this overgraph. The output of our reducing algorithm on the input x is defined by the condition that $x \in U$ if and only if $K'(0^x 1)$ is odd.

Note that the algorithm terminates even for incorrect answers of the oracle. □

Theorem 2.7 (An. Muchnik). *There exists a prefix entropy function with a non-tt-complete overgraph.*

Proof. We will use the following notion of a *finite game*. A finite game is determined by

- a finite set of *positions*,
- two directed graphs on that set (α-graph and β-graph),

- two complementary subsets of the position set (α-set and β-set), and
- an *initial* position d_0.

The union of α- and β-graph should be acyclic.

The game is played by two *players* (α- and β-player). The game starts in the position d_0 and consists of an infinite sequence of *moves* taken by the players in turn (player α has the first move). When the game is in a position d, the α-player can either stay in the same position or move to some position d' such that the edge (d, d') is present in the α-graph. The move of the player β is defined symmetrically. Since the union of α- and β-graph is acyclic, the game will stabilize at some position. If this position is in α-set, then α-player wins. Otherwise, he loses.

By the induction on the number of game positions we prove that there exists a winning strategy for one of the players. Let the set D_α contain all positions d such that the edge (d_0, d) is in the α-graph. Define the set D_β symmetrically. For every position $d \in D_\alpha$ define a game E^d_β. The position set of this game is the position set of the initial game without the position d_0. The α- and β-graphs as well as the α- and β-sets are the graphs and sets of the initial game restricted to the new position set. The initial position is d. Player β is the first to move. Symmetrically, for every position $d \in D_\beta$ we define a game E^d_α. If there exists a position $d \in D_\alpha$ such that the player α has a winning strategy in the game E^d_β then the winning strategy for α in the initial game starts with the move $d_0 \to d$ and continues as in E^d_β. If for every position $d \in D_\alpha$ player β has a winning strategy in E^d_β and, besides, the player β has a winning strategy in E^d_α for some $d \in D_\beta$, then there exists (an obvious) winning strategy for β in the initial game. Suppose that for every position $d \in D_\alpha$ player β has a winning strategy in E^d_β and for every position $d \in D_\beta$ player α has a winning strategy in E^d_α. Then α has a winning strategy if the initial position d_0 is in the α-set. If the initial position is in the β-set then β has a winning strategy. By the induction hypothesis, in every game E^d_α and every game E^d_β either α or β has a winning strategy. Consequently, one of the shown cases takes place, and either α or β has a winning strategy in the initial game. In fact, this proof leads to an *effective* construction of the winning strategy for one of the players.

Now return to the proof itself. Let us call a function f above enumerable if it has an enumerable overgraph. Denote by f^s the upper bound for f which one can obtain from the first s steps of the above enumeration of f. The difference $(f^{s+1} - f^s)$ can be any nonnegative function that is positive only in finite number of places. Let KP be any prefix entropy function. We will construct an enumerable set U and an above-enumerable function F for which the function $H = \min(KP+2, F)$ will be a prefix entropy function. In addition, the set U will not be tt-reducible to the overgraph of H. We assume that KP^0 and F^0 have finite values and $\sum_x 2^{-F^0(x)} \leqslant \frac{1}{4}$. The set U will be constructed as a subset of the universe of all triples of natural numbers. Let $\{\gamma_n\}$ be an enumeration of all tt-reducing partially defined algorithms. Using inputs of the form (n, i, j), we will make sure that the algorithm γ_n does not reduce U to the overgraph of H. The numbers n will be processed in the following order $1, 1, 2, 1, 2, 3, 1, 2, 3, 4, 1, 2, 3, 4, 5, \ldots$.

To each number n whose processing has already started we assign two natural numbers i_n, j_n. Also, for some n's we assign a finite game G_n and a position of that game. When we return to the number n the next time, we can either let one of the partners make his next move in the game G_n, or change the numbers i_n, j_n and the game G_n. A position in the game G_n is a finite set A_n (which is fixed for the game), a function $h_n : A_n \to \mathbb{N}$, and two rational numbers q_n and p_n (h_n, q_n and p_n may change with any move). Moves of the first player will determine the decrease of the estimate F^s. Moves of the second player will be determined by the decrease of the estimate KP^s. When we are processing the number n for the first time, we set $i_n = 1$ and $j_n = 1$. When we return to the number n again, we increase the value of i_n by 1 if the following holds: there exists a number m smaller than n such that the position in the game G_m or the game G_m itself was changed when we were dealing with m for the last time. If a change of the estimate KP^s violates the rules of the game G_n, we increase the value of j_n by 1.

Assume that we are processing the number n at the lth step. Define the values of i_n and j_n. If the numbers i_n and j_n remain the same, and the game G_n is defined, make a move in that game. If either i_n or j_n changed or G_n is undefined, run l steps of the algorithm γ_n on the input (n, i_n, j_n). If the algorithm did not produce a table[10] in the output, then G_n is undefined. If the algorithm produced a table, then the game G_n is defined as follows. The questions to oracle are of the form "is $H(x)$ smaller than r?", that is, a question is related to a pair (x, r). Let B_n be the set of all the first coordinates of the questions to the oracle from the table; then we put $A_n = B_n \setminus \bigcup_{m<n} A_m$. The function h_n on the set A_n should be bounded above by the function $\min(KP^l + 2, F^l)$. The numbers q_n and p_n should be less than 2^{-n-i_n-2}. In the initial position, we define $h_n = \min(KP^l + 2, F^l)$ and q_n=0, $p_n = 0$. The first player can move from a position (h_n, q_n, p_n) to a position (h'_n, q'_n, p'_n) such that $h'_n \leqslant h_n$, $q'_n - q_n = (\sum_x 2^{-h'_n(x)}) - (\sum_x 2^{-h_n(x)})$, $p'_n = p_n$. The second player can move from a position (h_n, q_n, p_n) to a position (h'_n, q'_n, p'_n) such that $h'_n \leqslant h_n$, $q'_n = q_n$, $p'_n - p_n = (\sum_x 2^{-h'_n(x)}) - (\sum_x 2^{-h_n(x)})$. To each function from B_n to $\mathbb{N}$ the table assigns the answer of the algorithm γ_n to the question whether the triple (n, i_n, j_n) belongs to U. We know that either the first player has a winning strategy to make the game stabilize at the answer "yes" or the second player has a winning strategy for stabilization at the answer "no". In the latter case, the first player also has a winning strategy for stabilization at the answer "no". Really, the game considered is "symmetric" except for who takes the first move. So, if the first player stays in the initial position after the first move, he further can follow the second player's winning strategy. In any case, there is one answer such that the first player has a winning strategy for stabilization at it. Let his winning set be the set of positions corresponding to that answer (then, the first player has a winning strategy). In addition, if this answer is "no", we put the element (n, i_n, j_n) into the set U.

We can assume that for even s we have $H^s = \min(KP^{s/2} + 2, F^{s/2})$ and for odd s we have $H^s = \min(KP^{(s-1)/2} + 2, F^{(s+1)/2})$. Suppose the game G_n was defined at the

[10] Which shows questions to the oracle and the outputs of the algorithm depending on the oracle's answers.

lth step. Then the relation between the position in this game after the vth move and the estimates for KP and F is determined by the equality $h_n = H^{2l+v}$ (for arguments in A_n). If the games with numbers different from n will not "interfere" and the second player will not violate the prohibition to change the value of $(\sum_x 2^{-h_n(x)})$ too much, the algorithm γ_n will make an error on the input (n, i_n, j_n) (if one considers this algorithm as attempting to reduce U to the overgraph of H). The games with numbers greater than n cannot interfere, since A_m does not intersect B_n for $m > n$. Let us prove by induction on n (as usually in the priority method) that the values of i_n, j_n, and G_n change a finite number of times. By the induction hypothesis, starting from some moment all the games with numbers smaller than n do not change. Since the games are finite, after some moment their positions are not changed. Therefore, the value of i_n stabilizes. After that, any change of j_n is the consequence of an increase of the value of $(\sum_x 2^{-KP^s(x)-2})$ by at least 2^{-n-i_n-2} (which is now fixed). Since $(\sum_x 2^{-KP^s(x)}) \leqslant 1$, the value of j_n stabilizes. When i_n and j_n are fixed, the table produced by the algorithm γ_n on the input (n, i_n, j_n) is uniquely determined if it exists. Therefore, G_n stabilizes as well. It remains to prove that the function $H = \min(KP + 2, F)$ is a prefix entropy. For this, one has to prove that $H \leqslant KP + \text{const}$, function H is above enumerable, and $\sum_x 2^{-H(x)} \leqslant 1$. The first follows straightforwardly from the definition of H. The function H is above enumerable because KP and F are. Finally, let us prove that $\sum_x 2^{-H(x)} \leqslant 1$. We will use the identity $\sum_x 2^{-H(x)} = \sum_x 2^{-H^0(x)} + \sum_x \sum_s (2^{-H^{s+1}(x)} - 2^{-H^s(x)})$. Since $\sum_x 2^{-H^0(x)} \leqslant \sum_x 2^{-F^0(x)}$, we have $\sum_x 2^{-H^0(x)} \leqslant \frac{1}{4}$. Every nonzero difference $(2^{-H^{s+1}(x)} - 2^{-H^s(x)})$ occurs either due to the decrease of the estimate for $(KP(x) + 2)$ or due to a move of the first player in one of the games. For any n and i there may be several games in which the second player violated the game prohibition, and no more than one game where the prohibition was not broken. Let us partition the set of pairs (x, s) into three parts. The first part consists of pairs with an odd s. The second part contains the pairs with an even s corresponding to the games with the prohibition unbroken. The third part consists of pairs with an even s corresponding to the games with the prohibition violated. The sum $(2^{-H^{s+1}(x)} - 2^{-H^s(x)})$ over pairs (x, s) from the first part is not greater than the sum $(2^{-KP^{s+1}(x)-2} - 2^{-KP^s(x)-2})$ over all pairs (x, s). The sum $(2^{-H^{s+1}(x)} - 2^{-H^s(x)})$ over pairs (x, s) from the second part does not exceed $\sum_{n,i} 2^{-n-i-2} = \frac{1}{4}$. The sum $(2^{-H^{s+1}(x)} - 2^{-H^s(x)})$ over pairs (x, s) from the third part is not greater than the sum $(2^{-KP^{s+1}(x)-2} - 2^{-KP^s(x)-2})$ over all pairs (x, s). Since $\sum_x \sum_s (2^{-KP^{s+1}(x)-2} - 2^{-KP^s(x)-2}) < \sum_x 2^{-KP(x)-2} \leqslant \frac{1}{4}$, we obtain $\sum_x 2^{-H(x)} \leqslant 1$. □

In conclusion let us prove a "quantitative" theorem.

Theorem 2.8 (An. Muchnik). *For any δ there exist x and v such that*

$$KS(x) + \delta < KS(v)$$

and

$$KP(v) + \delta < KP(x).$$

Proof. [11] Suppose that there is a δ such that for all x and v $KS(x)+\delta<KS(v)\Rightarrow KP(x)<KP(v)+\delta$. Then the algorithm from Theorem 2.5, which *tt*-reduces the enumerable set U to the overgraph of KS, can be modified to obtain an algorithm γ which *tt*-reduces the set U to the overgraph of KP. This contradicts Theorem 2.7 when U is *tt*-complete.

Recall that the proof of Theorem 2.5 implies the existence of an algorithm that takes an input z and constructs a natural number y and a set A of binary words of length y such that

$$z \in U \Rightarrow \forall p \in A \quad KS(p) < y-4d,$$

$$z \notin U \Rightarrow \exists p \in A \quad KS(p) \geqslant y.$$

The number d here is fixed in advance and may be as large as one wishes.

The new algorithm γ works as follows. If for all $p\in A$ there exists a binary word q of length $y-2d$ such that $KP(p)<KP(q)$, then γ decides that $z\in U$. Otherwise, γ decides that $z\notin U$.

To prove that the algorithm γ is well defined, we need two well-known facts.

Fact 1: $\exists c\ \forall n$ *for any word* w *of length* n *it is true that*

$$KP(w) < n+KP(n)+c.$$

Fact 2: $\exists c\ \forall n$ *there exists a word* w *of length* n *such that*

$$KP(w) > n+KP(n)-c.$$

To prove Fact 1 note that the function $f=\lambda w(\partial(w)+KP(\partial(w)))$ is above enumerable and $\sum_w 2^{-f(w)}\leqslant 1$ (where $\partial(w)$ denotes the length of w).

To prove Fact 2 we use the well-known relation between the prefix entropy and semimeasures enumerable from below. Consider the function $g=\lambda n\sum_{\partial(w)=n}2^{-KP(w)}$. Since g is enumerable from below and $\sum_n g(n)\leqslant 1$ we have $\forall n\ g(n) < \mathrm{const}\,2^{-KP(n)}$. In the sum $\sum_{\partial(w)=n}2^{-KP(w)}$ at least one of the summands must be less than 2^{-n} const $2^{-KP(n)}$, since there are 2^n summands. Passing to the logarithms, we obtain Fact 2.

Let us return to our algorithm γ.

If $\forall p\in A\ KS(p)<y-4d$ then for any random word r of length $y-3d$ we have $\forall p\in A\ KS(p)+\delta<KS(r)$. Indeed, $KS(r)\geqslant\partial(r)$, and one may assume that $d>\delta$. From the initial hypothesis we obtain $\forall p\in A\ KP(p)<KP(r)+\delta$. Using Fact 2 we find a word q of length $y-2d$ such that $KP(q)>y-2d+KP(y-2d)-c$. From Fact 1 we get $KP(r)<y-3d+KP(y-3d)+c$. Finally, we obtain $\forall p\in A\ KP(p)<KP(q)-d+KP(y-3d)-KP(y-2d)+\delta+2c$. As we know, $\exists c'\ \forall i,j\ KP(i)<KP(j)+KP(i|j)+c'$. Therefore, $KP(y-3d)-KP(y-2d)<KP(y-3d|y-2d)+c'$. To find $y-3d$ knowing

[11] If this theorem holds for one pair of functions KS and KP then it also holds for all such pairs. This follows from the fact that the difference between two entropy functions of the same type is bounded by a constant.

$y-2d$, it suffices to have d. Thus, $\forall p \in A\ KP(p) < KP(q) - d + \text{const} \log d$. For large d we get $\forall p \in A\ KP(p) < KP(q)$.

If for some $p \in A$ we have $KS(p) \geqslant y$, then consider a word s of length $y-d$ such that $KP(s) > y - d + KP(y-d) - c$ (here we use Fact 2). Since $KS(s) < \partial(s) + \text{const}$, for large d we have $KS(s) + \delta < KS(p)$. By the initial hypothesis, $KP(p) > KP(s) - \delta$. According to Fact 1, for any word q of length $y-2d$ we have $KP(q) < y - 2d + KP(y-2d) + c$. Combining all these inequalities, for any word q of length $y-2d$ we get $KP(p) > KP(q) + d - KP(y-2d) + KP(y-d) - \delta - 2c$. Applying the conditional entropy $KP(y-2d|y-d)$, we obtain $\exists p \in A\ \forall q\ (\partial(q) = y - 2d \Rightarrow KP(p) > KP(q))$ if d is large enough.

This proves that the algorithm γ is well defined. □

The question of *tt*-completeness was studied for the simple and the prefix entropies. One can prove an analogue of Theorem 2.5 for the decision entropy. An analogue of Theorem 2.7 can be verified for the a priori entropy.

Problem. *Is there a monotone entropy function whose overgraph is not tt-complete*?

Acknowledgements

The authors express gratitude to the Institute of New Technologies of Education, Moscow, for supporting their research in these unfortunate days for all Russian science. They thank the leaders of Kolmogorov Seminar – A. Semenov, A. Shen, N. Vereshchagin – and its participants for fruitful discussions. Part of this work was supported through the CNRS exchange program during the visit of one of the authors (AM) to the Laboratoire d'Informatique du Parallelisme (Lyon, France) kindly arranged by B. Durand. L. Positselsky and M. Semenova helped very much with English translation of this article. The author thank F. Stephan for reading carefully the preliminary version of this paper and making extremely useful suggestions. They also thank all those who contributed to the present work. The authors were partly supported by Russian Foundation Basic Research Grant 01-01-00505.

References

[1] M. Blum, I. Marques, On complexity properties of recursively enumerable sets, J. Symbolic Logic 38 (1973) 579–593.

[2] R.M. Friedberg, Three theorems on recursive enumeration, J. Symbolic Logic 23 (3) (1958) 309–316.

[3] G.N. Kobzev, On r-separable sets, Issledovanija po matematicheskoj logike i teorii algoritmov, University of Tbilisi, 1975, pp. 19–30 (in Russian).

[4] M. Kummer, On the complexity of random strings, Proc. 13th Symp. on Theoretical Aspects of Computer Science, Lecture Notes in Computer Science, vol. 1046, Springer, Berlin, 1996, pp. 25–36.

[5] M. Kummer, F. Stephan, Recursion theoretic properties of frequency computation and bounded queries, Inform. and Comput. 120 (1) (1995) 59–77.

[6] A.H. Lachlan, A note on universal sets, J. Symbolic Logic 31 (4) (1966) 573–574.

[7] A.H. Lachlan, wtt-Complete sets are not necessarily *tt*-complete, Proc. Amer. Math. Soc. 48 (2) (1975) 429–434.

[8] Al.A. Muchnik, On separability of recursively enumerable sets, Dok. Akad. Nauk SSSR 109(1) (1956) 29–32 (in Russian) (Translation available in Soviet Math. Dok.).

[9] Al.A. Muchnik, On reduction of the problems of decidability of enumerable sets to separability problems, Izv. Akad. Nauk SSSR Serija Mat. 29(3) (1965) 717–724. (Translation available in Soviet Math. Izv.)

[10] An.A. Muchnik, S.E. Positselsky, On one class of enumerable sets, Uspehi Mat. Nauk 54(3) (1999) 171–172 (in Russian) (Translation available in Russian Math. Surveys).

[11] E.L. Post, Recursively enumerable sets of positive integers and their decision problems, Bull. Amer. Math. Soc. 50 (5) (1944) 284–316.

[12] V. Uspensky, A. Semenov, Algorithms: Main Ideas and Applications, Kluwer Academic Publishers, Dordrecht, 1993.

[13] V. Uspensky, A. Shen, Relations between varieties of Kolmogorov complexities, Math. Systems Theory 29 (3) (1996) 271–292.