ELSEVIER

Theoretical Computer Science 271 (2002) 37–46

Theoretical Computer Science

www.elsevier.com/locate/tcs

Comparison between the complexity of a function and the complexity of its graph

Bruno Durand[a,*], Sylvain Porrot[b]

[a] *CMI - LIM, Technopôle de Château-Gombert, 39, rue Joliot-Curie, 13453 Marseille Cedex 13, France*
[b] *LAIL and LIFL, Bât. P2, Université des Sciences et Technologies de Lille, 59655 Villeneuve d'Ascq Cedex, France*

Abstract

This paper investigates in terms of *Kolmogorov complexity* the differences between the information necessary to compute a recursive function and the information contained in its graph. Our first result is that the complexity of the initial parts of the graph of a recursive function, although bounded, has almost never a limit. The second result is that the complexity of these initial parts approximate the complexity of the function itself in most cases (and in the average) but not always.

Keywords: Kolmogorov complexity; Recursive infinite sequences; Data flow complexity

0. Introduction

This paper is the long version of [2]. Its goal is to compare the information contained in the graph of a function to the information needed to compute the function. Our approach is based on *Kolmogorov complexity* (also known as *algorithmic information theory*). In this framework we compare the Kolmogorov complexity of a recursive function f, i.e. the size of a smallest program that computes f, with the conditional Kolmogorov complexities of initial parts of the graph of f. As far as we know, the only result in this field is a theorem of Meyer (see Theorem 2 in this paper), reported in the well-known article of Loveland [5]. A proof of this theorem is also given in the fundamental article of Zvonkin and Levin [8]. However, the point of view of these papers is different from ours: they are mainly interested in non-recursive sequences and in randomness. They also investigate varieties of Kolmogorov complexity (see on

* Corresponding author.
E-mail addresses: bruno.durand@gyptis.univ-mrs.fr, bdurand@cmi.univ-mrs.fr (B. Durand), porrot@lifl.fr (S. Porrot).

0304-3975/02/$ - see front matter
PII: S0304-3975(01)00029-9

this topic the paper by Uspensky and Shen [7]). We focus on recursive sequences (or functions).

Our study is also motivated by the analysis of data flows (see also [6]). Imagine a flow that, step by step, produces integer numbers. The information contained in the flow up to time t can be understood as the conditional Kolmogorov complexity of the outputs obtained before time t, knowing t. Our goal is to analyze the variations of this information when t varies.

Our results are rather surprising: the first one is that this information is bounded when the function is recursive, but has no limit, except for a finite number of functions (Theorem 1). Our second result is that the complexities of the initial parts of a graph do not always constitute an approximation of the complexity of the function (Theorem 3). In the case of data flows it means that, if we consider any recursive family of systems producing data flows, the amount of information issued by some of the systems is much lower than the information contained in the systems themselves. But we prove in Theorem 4 and in Corollary 1 that this strange behaviour appears rather rarely in the family, and that, on the average, this approximation is justified.

A more theoretical field of investigation is to compare the maximum of the complexity of the graph, its lim sup, the Kolmogorov complexity of the function and some other varieties of definitions of its Kolmogorov complexity relativised to oracles (e.g. the standard oracle set $\mathbb{K}$, also called as $0'$ in recursion theory), see [3].

1. Preliminaries

1.1. Kolmogorov complexity

Theory of *Kolmogorov complexity* [4], also called *Algorithmic Information Theory* [1], gives rigorous mathematical foundations to the notion of "information content" of an object x (represented by a word over the binary alphabet $\{0,1\}$). This quantity $K(x)$ is the length of a smallest program that halts and outputs x on an empty input. The programming language must satisfy an important technical property called *additive optimality* which is true in all "natural" programming languages:

$$\forall K_1, K_2 \ \exists C \ \forall x \ |K_1(x) - K_2(x)| < C,$$

where $K_1(x)$ and $K_2(x)$ are Kolmogorov complexities defined for two different additively optimal programming languages.

In order to talk about the complexity of integers, we use the following one-to-one mapping between words and integers: we associate each word with its index in the ordering, first by length, then lexicographically.

$$(\varepsilon,0),(0,1),(1,2),(00,3),(01,4),(10,5),(11,6),\ldots\ .$$

1.2. Definitions of models

We study recursive functions and their graphs $\mathscr{G}_f = \{\langle x, y\rangle,\ y = f(x)\}$. We denote by $\mathscr{G}_f^n$ the initial part of the graph $\mathscr{G}_f$, i.e. $\mathscr{G}_f^n = \{\langle x, y\rangle,\ x \leqslant n, y = f(x)\}$. As it is defined here, $\mathscr{G}_f^n$ is a set. So there is a choice of different representations for this set. We choose to identify $\mathscr{G}_f^n$ with $\langle f(0), f(1), \ldots, f(n)\rangle_n$ where $\langle . \rangle_n$ denotes the standard encoding of $\mathbb{N}^n$ in $\mathbb{N}$. Any other recursively equivalent definition could have been chosen.

Note that the special case where the images of the functions are restricted to the pair $\{0, 1\}$ is equivalent to the study of recursive infinite sequences. All results in the sequel remain valid if recursive functions from $\mathbb{N}$ to $\mathbb{N}$ are replaced by recursive sequences over the alphabet $\{0, 1\}$.

Definition 1. A program P is a *weak model* of a function f over a domain D if D is an infinite subset of $\mathbb{N}$ and $\forall n \in D\ P(n)$ halts and outputs $\mathscr{G}_f^n$.

Note that if $n \notin D$, either $P(n)$ does not halt, or $P(n)$ halts and its output can be different from $\mathscr{G}_f^n$.

Definition 2. The *weak complexity* of a function f is defined by

$$K_w(f) = \limsup_{n \to \infty} K(\mathscr{G}_f^n | n).$$

Note that the graph $\mathscr{G}_f^n$ has been properly encoded (see above) thus $K(\mathscr{G}_f^n | n)$ is correctly defined and any limit point of $K(\mathscr{G}_f^n | n)$ (reached over an infinite subset D of $\mathbb{N}$) corresponds to the size of a smallest weak model of f over D.

Definition 3. A *strong model* of a function f is a program P accepting one input n and that, for all n, halts and outputs $\mathscr{G}_f^n$, i.e. $\forall n \in \mathbb{N}\ P(n) = \mathscr{G}_f^n$.

We could present a different definition of a strong model e.g. $\forall n \in \mathbb{N}\ P(n) = f(n)$ instead of $\forall n \in \mathbb{N}\ P(n) = \mathscr{G}_f^n$. It gives, up to an additive constant, the same notion of strong complexity (see below).

Definition 4. The *strong complexity* $K_s(f)$ of a function f is the length of a smallest strong model of f if it exists, the infinity otherwise.

Remark that f is recursive if and only if $K_s(f)$ is finite.

2. A study of weak models

A strong model of a function is also a weak model of this function. Thus we get the following straightforward proposition:

Proposition 1. *If f is a recursive function, then there exists a strong model of f and we have $K_w(f) \leqslant K_s(f)$.*

If other (equivalent) definitions of K_w and K_s are given, then an additive constant is added to the previous proposition: $K_w(f) \leqslant K_s(f) + C$.

We now present some results concerning the series $K(\mathscr{G}_f^n|n)$ in order to justify the choice of the upper limit in the definition of the weak complexity. Indeed,

1. the limit of $K(\mathscr{G}_f^n|n)$ exists at most for a finite number of recursive functions f (Theorem 1);
2. the lower limit of $K(\mathscr{G}_f^n|n)$ is bounded by a constant not depending on f (Lemma 1), due to the fact that the whole information describing f can be infinitely often found encoded in the parameter n;
3. the upper limit of $K(\mathscr{G}_f^n|n)$ is the size of a smallest weak model for which the input n provides no information on f; thus a counting argument proves that it cannot be uniformly bounded (Lemma 2).

Lemma 1. *There exists a universal weak model for recursive functions: there exists a program $P_u(n)$ such that for all recursive function f, P_u is a weak model of f.*

Proof. Let $P_u(n)$ be the following program:

```
Program P_u(n)
    If there exist k,x such that n = 1^k 0 x then
        Simulate program x on input n
    Else
        Loop indefinitely
    End if
End
```

Let f be a recursive function of strong model P_g. P_u is a weak model of f since for all n in $\{1^k 0 P_g \mid k \in \mathbb{N}\}$, $P_u(n)$ computes $\mathscr{G}_f^n$. □

Lemma 2. *Given $A > 0$, the set of functions having a weak complexity bounded by A is finite.*

Proof. Given $A > 0$, assume that there are infinite many functions f such that $K_w(f) \leqslant A$. Let $f_1, \ldots, f_{2^{2A}}$ be such distinct functions. We have

1. $\forall k = 1, \ldots, 2^{2A}\ \exists n_k\ \forall n \geqslant n_k\ K(\mathscr{G}_{f_k}^n|n) \leqslant A$,
2. $\exists n_0\ \forall n \geqslant n_0\ \forall i, j = 1, \ldots, 2^{2A}\ i \neq j\ \mathscr{G}_{f_i}^n \neq \mathscr{G}_{f_j}^n$.

Let $n = \max\{n_0, n_1, \ldots, n_{2^{2A}}\}$. There exist 2^{2A} distinct graphs $\mathscr{G}_{f_k}^n$ with complexity less than A, which is obviously impossible. □

Lemmas 1 and 2 clearly imply the following theorem:

Theorem 1. *The set of recursive functions such that* $\lim_{n\to\infty} K(\mathscr{G}_f^n|n)$ *exists is finite.*

In this theorem the finite number of functions such that a limit exists depends on the programming system. Let us first present a system in which this number is zero. Consider a standard enumeration of partial recursive functions ϕ_i and let us define the following programming system: $\psi_0 = f_u$, f_u being the partial recursive function computed by P_u, $\psi_1, \ldots, \psi_{1998}$ are functions of which indexes are programs that always diverge, $\psi_i = \phi_i$ for all $i > 1998$. In this system, $\liminf K(\mathscr{G}_f^n|n) = 0$ and clearly $\limsup K(\mathscr{G}_f^n|n) \geqslant \text{length}(1998)$.

Now let us present a programming system in which the limit exists for some functions: $\gamma_0, \ldots, \gamma_{1998}$ are distinct total recursive functions, $\gamma_{1999} = f_u$ and $\gamma_i = \phi_i$ for all $i > 1999$.

3. Comparison between strong and weak models

3.1. Existence of a strong model

A well-known theorem (here Theorem 2) due to Meyer and reported in [5, 8] states that if $K(\mathscr{G}_f^n|n)$ is bounded over an infinite recursively enumerable domain D, then f is recursive. A weaker version of this theorem states that if the weak complexity of f is finite then f is recursive. In other terms, the hypothesis is that there is a finite number of weak models computing $\mathscr{G}_f^n$ for all n. This result is not obvious (and is rather strong) since we do not know a priori which one of all weak models computes $\mathscr{G}_f^n$ for a given n in D.

Theorem 2 (Meyer). *A function f is recursive if and only if there exists an infinite recursively enumerable set $D \subset \mathbb{N}$ where $K(\mathscr{G}_f^n|n)$ is bounded.*

3.2. Comparing weak and strong complexities

We have just seen that a finite weak complexity implies the existence of a strong model. Does weak complexity approximate strong complexity? The proof of Theorem 2 does not provide any answer to this question, because it is not constructive. Indeed, no proof of this theorem can be constructive as shown below in Theorem 3. As Kolmogorov complexity is defined up an additive constant, we need a family of functions to express this fact.

Theorem 3. *Let $\mathscr{F} = \{f_i\}_{i\in\mathbb{N}}$ be any recursive family of distinct recursive functions. Then $\forall C\ \exists f_i \in \mathscr{F}\ K_s(f_i) - K_w(f_i) > C$.*

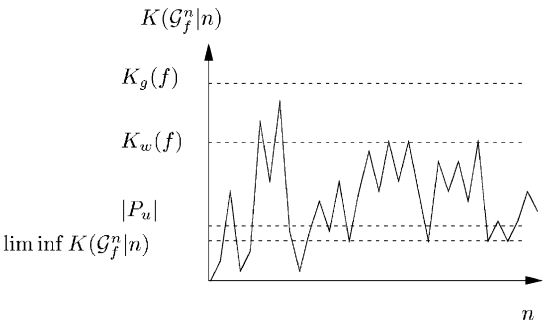

Fig. 1. Graph of $K(\mathcal{G}_f^n|n)$.

More precisely, a recursive family of recursive functions is a family such that $\exists\mathcal{P}\ \forall i,x\ \mathcal{P}(i,x)$ halts and outputs $f_i(x)$. In the sequel, we prove a stronger result: K_s is infinitely often of order k when K_w is of order $\log(k)$.

Example. The family $\mathcal{F}$ defined by

$$\forall n \leqslant i\ f_i(n) = 1 \quad \text{and} \quad \forall n > i\ f_i(n) = 0$$

satisfies the hypothesis of Theorem 3, and therefore we cannot bind by a constant the difference between the weak and the strong complexities of functions of this family.

In general, according to Lemma 1 and Theorem 3, the behaviour of the series $K(\mathcal{G}_f^n|n)$ as a function of n is illustrated by Fig. 1.

In order to prove Theorem 3 we need some preliminary results. In the following, the standard notation $\text{Step}(P,x,t)$ denotes the program that simulates t steps of program P with x as input, gives as result $P(x)+1$ if convergence is observed, else gives as result 0. In the proofs, the notation $\mathcal{O}_k(1)$ (resp. $\mathcal{O}_{k,n}(1)$) denotes a function of k (resp. of k and n) that is bounded by a constant.

Definition 5. Let P be a program and f a function. We define the *extension* properties $P \subset f$ and for all n $P \subset_n f$ by

$$(P \subset f) \ \Leftrightarrow\ (\text{if } P(x) \text{ converges, then } P(x) = f(x)),$$

$$(P \subset_n f) \ \Leftrightarrow\ (\forall x \leqslant n\ \text{Step}(P,x,n) \neq 0 \Rightarrow \text{Step}\ (P,x,n) = f(x)+1).$$

Definition 6 (*Goodness*). We say that i is *good* for k, $i \leqslant 2^k$, if

$$\exists P, |P| < k,\ (P \subset f_i \text{ and } \forall i' \leqslant 2^k, i' \neq i, P \not\subset f_{i'});$$

We say that i is *n-good* for k, $i \leqslant 2^k$, if

$$\exists P, |P| < k,\ (P \subset_n f_i \text{ and } \forall i' \leqslant 2^k, i' \neq i,\ P \not\subset_n f_{i'}).$$

We say that i is *bad* for k (resp. *n-bad*) if it is not good (resp. n-bad), i.e., respectively

$$\forall P, |P| < k,\ (P \subset f_i) \Rightarrow (\exists i' \leqslant 2^k, i' \neq i, P \subset f_{i'}),$$

$$\forall P, |P| < k,\ (P \subset_n f_i) \Rightarrow (\exists i' \leqslant 2^k, i' \neq i, P \subset_n f_{i'}).$$

Lemma 3. *For all n,k there exists $i \leqslant 2^k$ such that i is n-bad for k.*

Proof. There is 1 more function $\{f_i\}_{i \leqslant 2^k}$ than programs of length $\leqslant k$. □

The relation $\subset_n$ is an approximation of $\subset$ as proved in the following lemma, but beware that the convergence speed of $\subset_n$ to $\subset$ is not computable.

Lemma 4. *Given k, for all large enough n, $i(n,k)$ is stationary and equal to $i(k)$.*

The idea is that only finitely many n can change the n-goodness of the finitely many (i,k).

Proof. All integers between 0 and $i(k)-1$ are good for k while $i(k)$ is bad for k. First consider the integer 0. As it is good we are sure that there exists a program P, $|P|<k$, such that $P \subset f_0$. Clearly $P \subset_n f_0$. We are sure not to have $P \subset f_i$, $i \neq 0$ and $i \leqslant 2^k$. Thus there exists n_0 such that for $n \geqslant n_0$ we do not have $P \subset_n f_i$, $i \neq 0$ and $i \leqslant 2^k$. Hence for all $n \geqslant n_0$, 0 is n-good for k. Then let us iterate this process and find $n_1 \geqslant n_0$ such that for all $n \geqslant n_1$, 1 is n-good for k. Let us continue up to $i(k)-1$. Then let us examine the integer $i(k)$. $i(k)$ is bad for k: it means that either there does not exist a program of size less than k that computes a restriction of $f_{i(k)}$, or there exists such a program, but in this case it also computes a restriction of another f_i, $i \leqslant 2^k$. In the first case, we just have to wait for all programs of length lower than k to give an output not compatible with f. Hence, for all sufficiently large n, $i(k)$ is n-bad for k. In the second case, if P computes the restriction of two functions f_i, then it is also the case when restricted to n steps. For all n, $i(k)$ is n-bad for k. In conclusion, for n large enough $i(n,k) = i(k)$. □

Proof of Theorem 3 (*see acknowledgements*)**.** It is sufficient to prove the following inequalities:

1. $\forall k\ K_s(f_{i(k)}) \geqslant k$,
2. $\forall k\ \limsup_{n\to\infty} K(\mathscr{G}^n_{f_{i(k)}}|n) \leqslant \log(k) + \mathcal{O}_k(1)$.

Step 1: Suppose there exists k such that $K_s(f_{i(k)}) < k$. Then there exists a program $P(n)$, $|P|<k$, computing $f_{i(k)}(n)$ for all n. Since $i(k)$ is bad for k, there exists $i' \neq i(k)$ such that $P \subset f_{i'}$. Since P always converges, it means that for all $n, P(n)$ halts and outputs $f_{i'}(n)$. Thus $f_{i(k)} = f_{i'}$ which is false according to the hypothesis on the family $\mathscr{F}$.

Step 2: Since $\mathscr{F}$ is a recursive family of recursive functions there exists a program $\mathscr{P}$ such that for all i,x, $\mathscr{P}(i,x)$ halts and outputs $f_i(x)$. Consider the following program $P(n,k)$ accepting two inputs n and k and computing $i(n,k)$:

Program $P(n,k)$
Simulate n steps of all programs of length less than k with all inputs less or equal to n.
Compute the 2^k first functions f_i on $[0,n]$ using program $\mathscr{P}$.
Compare the results and output the smallest i that is bad for k.
End

Lemma 3 indicates that $P(n,k)$ computes $i(n,k)$ for all n and all k.

Because of the recursivity hypothesis on family $\mathscr{F}$, we can compute $\mathscr{G}^n_{f_i}$ from input i and n using $\mathscr{P}$, thus $\forall i\ K(\mathscr{G}^n_{f_i}|n) \leqslant K(i|n) + \mathscr{O}_n(1)$. Moreover, since for all n,k, $P(n,k)$ computes $i(n,k)$, we have

$$\begin{aligned}\forall n,k\ K(\mathscr{G}^n_{f_{i(n,k)}}|n) &\leqslant |P(.,k)| + \mathscr{O}_{n,k}(1)\\ &\leqslant \log(k) + \mathscr{O}_{n,k}(1).\end{aligned}$$

With Lemma 4, given k, there exists n_k such that $\forall n \geqslant n_k\ i(n,k) = i(k)$. Therefore, we get

$$\forall k\ \exists n_k\ \forall n \geqslant n_k\ K(\mathscr{G}^n_{f_{i(k)}}|n) \leqslant \log(k) + \mathscr{O}_{n,k}(1)$$

and finally we obtain

$$\limsup_{n\to\infty} K(\mathscr{G}^n_{f_{i(k)}}|n) \leqslant \log(k) + \mathscr{O}_k(1). \qquad \square$$

We have just seen that even for reasonable family of functions, the complexities of the initial parts of a graph do not always constitute an approximation of the complexity of the function itself, since some functions appear to be pathological. However, in most cases, both complexities are close to each other, as shown in the following theorem:

Theorem 4. *Let $\mathscr{F} = \{f_i\}_{i\in\mathbb{N}}$ be any recursive family of distinct recursive functions. There exist a constant A such that for all v and all k, there are at most a proportion of $A/2^v$ functions, among the k first functions f_i, for which the difference between strong and weak complexities is greater than v, i.e.*

$$\exists A\ \forall v\ \forall k\ \frac{\mathrm{Card}\{i \leqslant k \mid K_s(f_i) - K_w(f_i) \geqslant v\}}{k} \leqslant \frac{A}{2^v}.$$

Proof. Since the family $\mathscr{F}$ is a recursive family of recursive functions, there exists a program computing f_i for all i, thus we have

$$\exists A\ \forall i\ K_s(f_i) \leqslant \log(i) + A. \tag{1}$$

Since $K_w(f_i) = \limsup_{n\to\infty} K(\mathscr{G}^n_{f_i}|n)$ we get

$$\forall i\ \exists n'_i\ \forall n \geqslant n'_i\ K_w(f_i) \geqslant K(\mathscr{G}^n_{f_i}|n) \tag{2}$$

and from Eqs. (1) and (2) we obtain

$$\exists A\ \forall i\ \exists n'_i\ \forall n \geqslant n'_i\ K_s(f_i) - K_w(f_i) \leqslant \log(i) + A - K(\mathscr{G}^n_{f_i}|n). \tag{3}$$

Let k and v be fixed. Since all functions of family $\mathscr{F}$ are distinct, there exists n_k such that, for all $n \geqslant n_k$, all truncated graphs $(\mathscr{G}^n_{f_i})_{i<k}$ are distinct. Let $S(v,k)$ denote the set $\{i \leqslant k, K_s(f_i) - K_w(f_i) \geqslant v\}$. Given $n \geqslant \max\{n_k, (n'_i)_{i\leqslant k}\}$ and according to Eq. (3) we have

$$\begin{aligned} S(v,k) &\subset \{i \leqslant k, \log(i) + A - K(\mathscr{G}^n_{f_i}|n) \geqslant v\} \\ &= \{i \leqslant k, K(\mathscr{G}^n_{f_i}|n) \leqslant \log(i) + A - v\} \\ &\subset \{i \leqslant k, K(\mathscr{G}^n_{f_i}|n) \leqslant \log(k) + A - v\}. \end{aligned}$$

Since $n \geqslant n_k$ all truncated graphs $(\mathscr{G}^n_{f_i})_{i\leqslant k}$ are distinct. Thus all smallest programs computing these truncated graphs are distinct too and we can conclude that

$$\mathrm{Card}(S(v,k)) \leqslant 2^{\log(k)+A-v}. \qquad \square$$

Corollary 1. *Let $\mathscr{F} = \{f_i\}_{i\in\mathbb{N}}$ be any recursive family of distinct recursive functions. The difference between strong and weak complexities is in average bounded by a constant*:

$$\exists C\ \forall k \frac{\sum_{i\leqslant k}(K_s(f_i) - K_w(f_i))}{k} \leqslant C.$$

Proof. Let $d(v,k) = \mathrm{Card}\{i \leqslant k \mid K_s(f_i) - K_w(f_i) = v\}$ for all v and k. According to Theorem 4 we have $\exists A\ \forall v\ \forall k\ d(v,k) \leqslant A/2^v k$. Thus,

$$\begin{aligned} \sum_{i\leqslant k}(K_s(f_i) - K_w(f_i)) &= \sum_{v=0}^{\infty} d(v,k)v \\ &\leqslant \sum_{v=0}^{\infty} \frac{A}{2^v} kv. \end{aligned}$$

Since $\sum(A/2^v)v$ converges, we have the expected result. $\square$

4. Open problems

In this paper we have studied graphs of recursive functions. These graphs are relations on $\mathbb{N}\times\mathbb{Z}$ with some additional properties: not only the relation is recursive but also all its projections to given abscissas are uniformly recursive. If we consider a data flow, we may represent it by a relation R such that for all x there exists a finite number of y such that $R(x, y)$. It means that at each time step, a finite (but not bounded) number of outputs is issued. It may be interesting to extend this study to these relations or, more generally, to any recursive relation.

Acknowledgements

The authors are indebted to, chronologically, Pr. Alexander Shen and Pr. Nikolaï Vereshchagin for their help in proving Theorem 3 and earlier versions (see also [3]). We thank Pr. Max Dauchet for his constructive remarks, comments and encouragements.

References

[1] C. Calude, Information and Randomness, Springer, Berlin, 1994.
[2] B. Durand, S. Porrot, in: Comparison between the complexity of a function and the complexity of its graph, MFCS'98, Lecture Notes in Computer Science, Springer, Berlin, 1998.
[3] B. Durand, A. Shen, N. Vereshchagin, Descriptive complexity of computable sequences, Theoret. Comput. Sci. 271 (this Vol.) (2002) 47–58.
[4] M. Li, P. Vitányi, An Introduction to Kolmogorov Complexity and its Applications, 2nd ed., Springer, Berlin, 1997.
[5] D.W. Loveland, A variant of the Kolmogorov concept of complexity, Inform. and Control 15 (1969) 510–526.
[6] S. Porrot, M. Dauchet, B. Durand, N. Vereshchagin, in: Deterministic rational transducers and random sequences, FOSSACS'98, Lecture Notes in Computer Science, Springer, Berlin, 1998.
[7] V.A. Uspensky, A. Kh. Shen, Relations between varieties of Kolmogorov complexities, Math. Systems Theory 29 (3) (1996) 270–291.
[8] A.K. Zvonkin, L.A. Levin, The complexity of finite objects and the development of the concepts of information and randomness by means of the theory of algorithms, Russian Math. Surveys 25 (1970) 83–124.