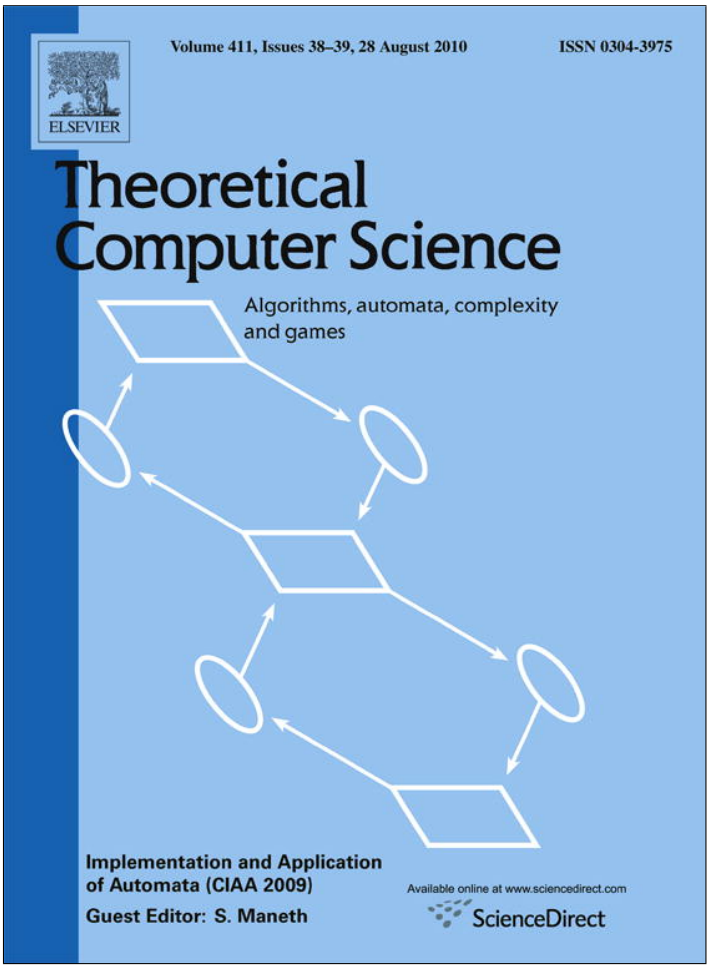

Theoretical Computer Science 411 (2010) 3436–3443

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Amount of nonconstructivity in deterministic finite automata

Rūsiņš Freivalds

Institute of Mathematics and Computer Science, University of Latvia, Raiņa bulvāris 29, LV-1459 Rīga, Latvia

ARTICLE INFO

Keywords:
Finite automata
Nonconstructive methods
Kolmogorov complexity

ABSTRACT

When D. Hilbert used nonconstructive methods in his famous paper on invariants (1888), P. Gordan tried to prevent the publication of this paper considering these methods as non-mathematical. L.E.J. Brouwer in the early twentieth century initiated intuitionist movement in mathematics. His slogan was "nonconstructive arguments have no value for mathematics". However, P. Erdös got many exciting results in discrete mathematics by nonconstructive methods. It is widely believed that these results either cannot be proved by constructive methods or the proofs would have been prohibitively complicated. The author (Freivalds, 2008) [10] showed that nonconstructive methods in coding theory are related to the notion of Kolmogorov complexity.

We study the problem of the quantitative characterization of the amount of nonconstructiveness in nonconstructive arguments. We limit ourselves to computation by deterministic finite automata. The notion of nonconstructive computation by finite automata is introduced. Upper and lower bounds of nonconstructivity are proved.

1. Introduction

The use of nonconstructive methods of proof in mathematics has a long and dramatic history. In 1888 a young German mathematician David Hilbert presented to his colleagues three short papers on invariant theory. Invariant theory was the highly estimated achievement of Paul Gordan who had produced highly complicated constructive proofs but left several important open problems. The young David Hilbert had solved all these problems and had done much more. Paul Gordan was furious. He was not ready to accept the new solutions because they provided no explicit constructions. Hilbert merely proved that the solutions cannot fail to exist. Gordan refused to accept this as mathematics. He even used the term "theology" and categorically objected to publication of these papers. Nonetheless the papers were published first in *Göttingen Nachrichten* and later, in final form, in [14].

Later Hilbert had one more highly publicized controversy. This time Luitzen Egbertus Jan Brouwer was involved. Following Henri Poincare's ideas, Brouwer started a struggle against nonconstructive proofs. The *intuitionist* movement was started in mathematics. This was part of the attempts to overcome the crisis in foundations of mathematics. Many possible ways out of the crisis were proposed in twenties of the 20th century. Hilbert axiomatized geometry and wished to axiomatize all mathematics. K. Gödel proved his famous incompleteness theorems and showed that the crisis cannot be overcome so easily. Brouwer reasonably turned everybody's attention to the fact that considering infinite sets as objectively existing objects is dangerous and it can bring us to unforeseen conclusions not related to our experience. Brouwer challenged the belief that the rules of the classical logic, which have come down to us essentially from Aristotle (384–322 B.C.) have an absolute validity, independent of the subject matter to which they are applied. Nonconstructive proofs were to be thrown out of mathematics.

E-mail address: Rusins.Freivalds@mii.lu.lv.

0304-3975/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2010.05.038

In the forties the situation, however, changed. In spite of all philosophical battles the nonconstructive methods found their way even to discrete mathematics. This was particularly surprising because here all the objects were finite and it seemed that no kind of distinction between *actual infinity* and *potential infinity* could influence these proofs while most of the discussions between intuitionists and classicists were around these notions. Paul Erdös produced many nice nonconstructive proofs, the first paper of this kind being [6].

We try in this paper to go another step. We propose a quantitative approach to measure the amount of nonconstructivity in a proof. A notion of nonconstructive computation is introduced as a result of examination of examples of nonconstructive proofs [3,4,8–10,19]. This notion can easily be used for many types of automata and machines. In this paper we prove several upper and lower bounds for the amount of nonconstructivity in nonconstructive deterministic finite 2-way automata. This type of automata is sufficiently simple but it allows nontrivial constructions.

Karp and Lipton have introduced in [15] a notion *Turing machine that takes advice* which is practically the same notion for Turing machines as our *nonconstructive computation by finite automata* below. Later Damm and Holzer have adapted the notion of advice for finite automata. Since there was no reference to intuitionism in [15], the adaptation was performed in the most straightforward way (what is quite natural). However the notion of *finite automata that take advice* in [5] differs from our notion very much. These notions are equivalent for large amounts of nonconstructivity (or large amounts of advice) but, for the notion introduced in [5] and later extensively used by Yamakami and his coauthors [22,18,21], languages recognizable with polynomial advice are the same languages which are recognizable with a constant advice. Our notion of the amount of nonconstructivity is such that our most interesting results concern the smallest possible amounts of nonconstructivity.

A similar situation was in the sixties of the 20th century with space complexity of Turing machines. At first, space complexity was considered for one-tape off-line Turing machines and it turned out that space complexity is never less than linear. However, it is difficult to prove such lower bounds. Then the seminal paper by Stearns et al. [20] was published and many-tape Turing machines became a standard tool to study sublinear space complexity.

2. Re-examining nonconstructive proofs

2.1. Codes and Kolmogorov complexity

The textbook [13] contains

Theorem 1 ([13]). *For any integer $n \geq 4$ there is a $[2n, n]$ binary code with a minimum distance between the codewords at least $n/10$.*

However the proof of this theorem in [13] has an unusual property. It is nonconstructive. It means that we cannot find these codes or describe them in a useful manner. This is why P. Garrett calls them *mirage codes*.

The paper [10] was written to prove that the size (i.e., the number of states) of a deterministic finite automaton and the size of a probabilistic finite automaton recognizing the same language can differ exponentially, thus concluding a long sequence of papers describing the gap between the size of deterministic and probabilistic finite automata recognizing the same language.

A counterpart of Theorem 1 for cyclic linear codes was needed, but an attempt to prove it failed. Instead of cyclic generating matrices a slightly different kind of generating matrices was considered. Let p be an odd prime number, and x be a binary word of length p. The generating matrix $G(p, x)$ has p rows and $2p$ columns. Let $x = x_1x_2x_3 \ldots x_p$. The first p columns (and all p rows) form a unit matrix with elements 1 on the main diagonal and 0 in all the other positions. The last p columns (and all p rows) is a cyclic matrix with $x = x_1x_2x_3 \ldots x_p$ as the first row, $x = x_px_1x_2x_3 \ldots x_{p-1}$ as the second row, and so on. We will refer below the generating matrices with this property as bi-cyclic.

The notion of Kolmogorov complexity was used to prove the counterpart of Theorem 1 for the codes with a bi-cyclic generating matrix.

Definition 1. We say that the numbering $\Psi = \{\Psi_0(x), \Psi_1(x), \Psi_2(x), \ldots\}$ of 1-argument partial recursive functions is **computable** if the 2-argument function $U(n, x) = \Psi_n(x)$ is partial recursive.

Definition 2. We say that a numbering Ψ is **reducible** to the numbering η if there exists a total recursive function $f(n)$ such that, for all n and x, $\Psi_n(x) = \eta_{f(n)}(x)$.

Definition 3. We say that a computable numbering φ of all 1-argument partial recursive functions is **Gödel numbering** if every computable numbering (of any class of 1-argument partial recursive functions) is reducible to φ.

Theorem 2 ([7]). *There exists a Gödel numbering.*

Definition 4. We say that a Gödel numbering ϑ is a **Kolmogorov numbering** if for arbitrary computable numbering Ψ (of any class of 1-argument partial recursive functions) there exist constants $c > 0$, $d > 0$, and a total recursive function $f(n)$ such that:

1. for all n and x, $\Psi_n(x) = \vartheta_{f(n)}(x)$,
2. for all n, $f(n) \leq c \cdot n + d$.

Theorem 3 ([16]). *There exists a Kolmogorov numbering.*

Definition 5. We say that a binary word w has Kolmogorov complexity s with respect to the Kolmogorov numbering ϑ if the least value of n such that $\vartheta(n) = \omega$ where ω is a natural number whose binary representation equals w.

Unfortunately (or fortunately) there are infinitely many distinct Kolmogorov numberings. Nonetheless, the Kolmogorov complexities of the same word with respect to distinct Kolmogorov numberings differ at most by an additive constant. The Kolmogorov complexity is usually understood as the degree of the extent how much the word can be compressed without loss of information. For an individual word it may be difficult to implement this semantics but if we consider infinite sequences of words then this semantics is applicable to all sufficiently long words.

The crucial point in the proof of the main result of [10] was the following lemma.

Lemma 1 ([10]). *If p is a sufficiently large prime, and the word $x = x_1x_2x_3\ldots x_p$ in the definition of a bi-cyclic matrix has Kolmogorov complexity $p - o(p)$ then the Hamming distance between arbitrary two codewords is at least $\frac{4p}{19}$.*

Kolmogorov complexity brings in an element of nonconstructivity. Indeed, no algorithm can exist finding such a word $x = x_1x_2x_3\ldots x_p$ for a given p. Such words merely exist. Moreover, nearly all the words of the length p have this property. However, every algorithm producing the needed words inevitably fails. On the other hand, if somebody from outside could help us and provide us with a word $x = x_1x_2x_3\ldots x_p$ with the property "*Kolmogorov complexity of this word is maximal possible for the words of this length*", we would be able to construct the bi-cyclic generating matrix.

2.2. Kolmogorov complexity of recursively enumerable sets

Bārzdiņš [3] studied the Kolmogorov complexity of binary words of length n expressing whether a natural number x (where $0 \leq x \leq n-1$) belongs to a recursively enumerable set. Recursively enumerable sets are those for which an algorithm exists enumerating (not always in an increasing order) all the elements of the set.

The result was surprising. It turned out that Kolmogorov complexity of such words never exceed $\log n$. The main technical lemma from Bārzdiņš' paper can be reformulated as follows.

There is no algorithm uniform in n such that it would show which numbers belong to the recursively enumerable set and which ones do not belong. However, when we are trying to decide which natural numbers x (where $0 \leq x \leq n-1$) belong to the recursively enumerable set, somebody from outside would come and provide us with the number of the natural number y such that $0 \leq y \leq n-1$ and the enumerating algorithm lists this y as the last among the numbers x from the elements of the recursively enumerable set such that $0 \leq x \leq n-1$, then we would be able to construct the needed decision algorithm for the initial fragment $0 \leq x \leq n-1$ ourselves.

2.3. Learning programs for total functions

Podnieks studied learning in the limit programs of total recursive functions by deterministic and probabilistic learning algorithms. Among other results he produced two theorems on deterministic learning in the limit of indices of total recursive functions in numberings defined by a total universal function of two arguments $U(n, x) = f_n(x)$.

Theorem 4 ([12,19]). *If the numbering U is such that the algorithmic problem of equivalence of indices is decidable, then there is a learning algorithm which makes on any function $f_i(x)$ (where $0 \leq i \leq n$) no more than $g(n)$ mindchanges where $g(n)$ is a total recursive function arbitrarily slowly monotonically growing to infinity.*

Theorem 5 ([4]). *There is a numbering U such that an arbitrary learning algorithm for infinitely many values of n makes on some function $f_i(x)$ (where $0 \leq i \leq n$) no less than $\frac{n}{2}$ mindchanges.*

We can re-interpret K. Podnieks' results as follows. The learning of indices of total recursive functions in numberings defined by a total universal function of two arguments $U(n, x) = f_n(x)$ demands in general $\frac{n}{2}$ mindchanges for infinitely many functions $f_n(x)$. However, if somebody from outside could come and provide us with the information which indices are equivalent and which ones are not, then we would be able to construct the needed learning algorithm with far less mindchanges.

3. Definitions

All three examples considered in the previous section have something in common. An algorithm is presented in a situation where (seemingly) no algorithm is possible. However, this algorithm has an additional input where a special help is fed in. If this help is correct, the algorithm works correctly. On the other hand, this help on the additional input does not just provide the answer. There still remains much work for the algorithm.

Is this nonconstructivism merely a version of nondeterminism? Not at all. The construction of bi-cyclic generating matrices for codes in Section 2.1 had nothing to do with existence of certain inputs. If the Kolmogorov complexity of a word is high, then its Hamming distance is large.

The additional information about the recursively enumerable sets in Section 2.2 always exists. If this information is provided correctly, the algorithm is correct. The answer YES or NO depends on x and the additional information is much more compact than a list of all the answers.

The additional information in the index learning problem in Section 2.3 also does not provide the needed answers directly. Surely it is also not a version of nondeterminism.

All these examples naturally lead to the following notion of nonconstructive computation.

Definition 6. We say that an automaton A recognizes the language L nonconstructively if the automaton A has an input tape where a word x is read and an additional input tape for nonconstructive help y with the following property. For arbitrary natural numbers n there is a word y such that for all words x whose length does not exceed n the automaton A on the pair (x, y) produces the result 1 if $x \in L$, and A produces the result 0 if $x \notin L$. Technically, the word y can be a tuple of several words and may be placed on separate additional input tapes.

Definition 7. We say that an automaton A recognizes the language L nonconstructively with nonconstructivity $d(n)$ if the automaton A has an input tape where a word x is read and an additional input tape for nonconstructive help y with the following property. For arbitrary natural numbers n there is a word y of the length not exceeding $d(n)$ such that for all words x whose length does not exceed n the automaton A on the pair (x, y) produces the result 1 if $x \in L$, and A produces the result 0 if $x \notin L$. Technically, the word y can be a tuple of several words and may be placed on separate additional input tapes. In this case, $d(n)$ is the upper bound for the total of the lengths of these words.

The automaton A in these definitions can be a finite automaton, a Turing machine or any other type of automata or machines. In this paper we restrict ourselves by considering only deterministic finite automata with 2-way behavior on each of the tapes.

This way, we can characterize the amount of nonconstructivity in the nonconstructive algorithms considered in Sections 2.1–2.3. Freivalds' nonconstructive algorithm for construction of bi-cyclic generating matrices of size $2n \times n$ has nonconstructivity n. Bārzdiņš' nonconstructive algorithm for construction of decision algorithm for initial fragments $[0, n-1]$ of characteristic functions of recursively enumerable languages has nonconstructivity $\log n$. Podnieks' nonconstructive algorithm for learning in the limit of indices of total recursive functions in numberings defined by a total universal function of two arguments $U(m, x) = f_m(x)$ with at most $g(n)$ mindchanges for all the functions in the set $f_0(x), \ldots, f_{n-1}(x)$ has nonconstructivity $const.n^2$. Of course, these are only upper bounds. It is quite possible that for some of these problems nonconstructive algorithms with a lesser nonconstructivity are possible. (This is not the case for recursively enumerable languages since Bārzdiņš has also proven a tight lower bound for the Kolmogorov complexity of recursively enumerable languages.)

4. Results on finite automata

Theorem 6. *There exists a nonregular (and even a nonrecursive) language L such that it can be nonconstructively recognized with nonconstructivity n.*

Proof. Let $a_1, a_2, a_3, \ldots$ be an infinite sequence of zeros and ones. We define the language L as follows. A word x is in L iff it coincides with some initial fragment of the sequence $a_1, a_2, a_3, \ldots$.

If the sequence is not recursive then the language is also nonrecursive. On the other hand, the nonconstructive automaton with the nonconstructive help $a_1, a_2, a_3, \ldots, a_n$ is able to provide the results whether the given word w is in L for all binary words of the length not exceeding n. □

Theorem 7. *For the language L in the proof of Theorem 6 , if $h(n)$ is a total function such that $\log_2 n = o(h(n))$, then no nonconstructive 2-way deterministic finite automaton can recognize L with nonconstructivity $(n - h(n))$.*

Proof. Martin-Löf [17] proved that there exists an infinite sequence $a_1, a_2, a_3, \ldots$ such that infinitely many initial fragments of it have Kolmogorov complexity n and all the initial fragments of it have Kolmogorov complexity no less than $n - \Omega(\log_2 n)$. (He also proved that there are no infinite binary sequences with a higher Kolmogorov complexity.) Take this sequence. Consider the corresponding language L. Assume from the contrary that there exists a nonconstructive deterministic finite automaton such that for infinitely many values of n the nonconstructivity is less than $(n - h(n))$. From the given program of the automaton and from the given nondeterministic help one can algorithmically reconstruct the values $a_1, a_2, a_3, \ldots, a_n$. Hence Kolmogorov complexity of the initial fragment $a_1, a_2, a_3, \ldots, a_n$ exceeds the length of the nonconstructive help no more than by a constant and the initial fragment has Kolmogorov complexity no higher than $(n - h(n))$. Contradiction. □

Theorem 8. *For arbitrary natural numbers k there exists a nonregular language L such that it can be nonconstructively recognized with nonconstructivity not exceeding $n^{\frac{1}{k}}$.*

Proof. At first we consider the language L consisting of all the words

$$0^m 1 0^m 1 0^m 1 \ldots 1 0^m$$

where the number of arrays of zeros is the same as the length of these arrays.

Can we use 0^m as the help-word? Indeed, this help-word helps for $0^m10^m10^m1\ldots10^m$. However, our definition of nonconstructive computation demands that the help-word works for all the shorter input words as well. Unfortunately, 0^m does not help for shorter words. The help-word $0^1 10^2 10^3 1\ldots 10^m$ works both for $0^m10^m10^m1\ldots10^m$ and for all the shorter input words. Unfortunately, the length of this help-word is $O(n)$.

Now we consider the language consisting of all the words

$$0^m10^m10^m1\ldots10^m20^m10^m10^m1\ldots10^m2\ldots20^m10^m10^m1\ldots10^m.$$

This language can be helped by the same help-word $0^1 10^2 10^3 1\ldots 10^m$ and this help-word works for all input words the length of which does not exceed m^3. Hence the length of the help-word in terms of the length of the input word is $n^{\frac{1}{3}}$.

This idea can be extended even more, and the language can be considered which consists of all the words $w_2 3 w_2 3\ldots 3 w_2$ where

$$w_2 = 0^m10^m10^m1\ldots10^m20^m10^m10^m1\ldots10^m2\ldots20^m10^m10^m1\ldots10^m.$$

This language can be helped by the same help-word $0^1 10^2 10^3 1\ldots 10^m$. This reduces the length of help-word to $n^{\frac{1}{3}}$. Iterating this idea r times, we get the length of the help-word $n^{\frac{2}{r}}$ which can be made smaller than any $n^{\frac{1}{k}}$.

This way, we have constructed a language in an alphabet consisting of more than two letters. There is no difficulty to encode the input words in a binary alphabet not damaging recognizability of the language. □

Theorem 9. *There exists a nonregular language L and a function $g(n)$ such that L can be nonconstructively recognized with nonconstructivity $g(n)$ and $\log n \leq g(n) \leq (\log n)^2$.*

Proof. The language L consists of all binary words in the form 0^m20^k such that $k > 0$ is a multiple of the product of the first m primes (where $p_1 = 2, p_2 = 3, p_3 = 5, \ldots$). The help-word is

$$0^{p_1}10^{p_2}10^{p_3}1\ldots10^{p_m}.$$

The nonconstructive automaton has 2-way heads on the two tapes. This allows it to check whether the length k of the array 0^k is a multiple of p_1 being the length of the first array of zeros on the help-tape, whether the length k of the array 0^k is a multiple of p_2 being the length of the second array of zeros on the help-tape, etc.

If the input word w has a prefix 0^m1 and $w \in L$ then the length of w is at least the product of all first m primes. This product is called *primorial of m* and it is known that $Primorial(m) \approx e^{m \ln m}$ (see e.g. [2]). The length of the help-word that helps for all the input words with such a prefix is

$$\Sigma(m) = \sum_{s=1}^{m} p_s + (m-1).$$

It is known that

$$\Sigma(m) \approx \frac{1}{2} m^2 \ln m.$$

However, it is not true that all the input words of such a length have the considered prefix. It may happen that the prefix contains more zeros before the first symbol 1. Nonetheless, the help-word described above works for such input words as well. Indeed, if the input word w has a prefix 0^r1 and $r > m$ and the length of w is less than $Primorial(m)$ then $w \notin L$. On the other hand, $|w| < Primorial(m)$ implies that there exists a natural number u such that $|w|$ does not divide p_u and the help-word described above provides the needed result NO. □

Theorem 10. *If a language L can be nonconstructively recognized with a nonconstructivity bounded by a function $d(n) = o(\log n)$, then L is regular.*

Proof. Our proof is based on the following feature of the definition of nonconstructiveness. We have demanded that for arbitrary natural numbers n there is a word y such that for all words x whose length does not exceed n the automaton A on the pair (x, y) produces a correct result. Let A be a deterministic finite automaton nonconstructively recognizing L, and $d(n)$ be the smallest possible length of the help-word for the words x of the length n. Assume from the contrary that $d(n)$ grows to infinity. It follows that for arbitrary n and for arbitrary y with $|y| < d(n)$ there exists a word $x(y)$ such that the automaton A on $x(y)$ produces a wrong result with the help y. By $x_n(y)$ we denote the shortest word $x(y)$ such that A on $x(y)$ produces a wrong result with the help y. Denote the length of $x_n(y)$ by m. Let $y(m)$ be the shortest possible help-word such that A produces correct results for all input words of the length not exceeding m. Clearly, $|y(m)| \leq d(m)$. Hence A produces on $x_n(y)$ a correct result with the help $y(m)$ and a wrong result with the help y.

The automaton A is two-way on each of the tapes. For arbitrary k we consider the set of all possible configurations of the memory and the help-tape at moments when the head on the work-tape in on the distance k from the beginning of the input word. By $B(k)$ we denote such a set of configurations for A on x_0 with the help $y(n)$ and by $C(k)$ we denote such a set of configurations for A on x_0 with the help $y(m)$.

Each of the sets $B(k)$ and $C(k)$ consists of no more than $const.d(m)$ elements. Hence there can be no more than $\exp(d(m))$ distinct sets $B(k)$, $\exp(d(m))$ distinct sets $C(k)$, and $\exp(d(m))$ distinct pairs $(B(k), C(k))$. Since $d(n) = o(\log n)$, there exist

two distinct k and l such that both $B(k) = B(l)$ and $C(k) = C(l)$. Cutting out the fragment between the k-th and the l-th symbols from $x_n(y)$ we get a word named x_1 shorter than $x_n(y)$. The automaton A produces on x_1 with the help $y(m)$ the same result as on x_1 and this is a correct result, by the definition of $y(m)$. The automaton A produces on x_1 a different result on x_1 with the help y (because the result on $x_n(y)$ and the result on x_1 is the same). Hence $x_n(y)$ is not the shortest word such that A produces a wrong result with the help y. Contradiction.

We have proved that the nonconstructivity is bounded by a constant. Hence there is a help-word which fits for infinitely many n. We can conclude that this help-word fits for all input words x. This universal help-word can be incorporated into the automaton, and we get a deterministic finite 2-way automaton recognizing the language without any help from outside. It follows that the language is regular. □

Theorem 11. *There exists a nonrecursive language L and a function $g(n)$ such that L can be nonconstructively recognized by a DFA with a nonconstructivity $g(n) \in polylog(n)$.*

Proof. To define the language we need an infinite nonrecursive binary sequence $r_1, r_2, r_3, \ldots$. By p_i we denote the i-th prime ($p_1 = 2, p_2 = 3, p_3 = 5, \ldots$). We define a sequence s_i depending on r_i:

$$s_i = \begin{cases} p_{2i} & \text{if } r(i) = 0, \\ p_{2i+1} & \text{if } r(i) = 1. \end{cases}$$

The language L consists of all binary words in the form $0^m 20^k$ such that $k > 0$ is a multiple of the product of the first m numbers in the sequence $s_1, s_2, s_3, \ldots$. The help-word is $0^{s_1}10^{s_2}10^{s_3}1\ldots10^{s_m}$. □

5. Arbitrary languages

Theorem 12. *Every language L over the binary alphabet $\{0, 1\}$ can be recognized nonconstructively with nonconstructivity $O(n2^n)$.*

Proof. The idea is to encode the language as word/answer pairs in the help-word.

We will use the ternary alphabet 0, 1, 2 for our help-word. Suppose that our help-word will have to work for all input words of length not exceeding n. The help-word will be the concatenation of "$w2a_w2$" for all words w of length not exceeding n, where a_w is defined to be 1 if w is in L and 0 otherwise. The order in which these strings are to be concatenated does not matter to our automaton.

The automaton compares the input word with every word w encoded in the help-word and returns the answer a_w if the input word is equal to w. The automaton moves on to the next word if it is not.

Each of the possible input words on the help-word tape has length equal or less than to n. There are exactly $2^n + 2^{n-1} + \cdots + 1 = 2^{n+1} - 1$ possible input words in the help-word together with $2^{n+1} - 2$ boundary markers. Therefore the total length of the help-word is $O(n2^n)$. □

The above theorem is provided because of its simplicity. However it is strengthened below.

A k-ary De Bruijn sequence $B(k, n)$ of order n, named after the Dutch mathematician Nicolaas Govert de Bruijn, is a cyclic sequence of a given alphabet A with size k for which every possible subsequence of length n in A appears as a sequence of consecutive characters exactly once.

According to De Bruijn himself,[1] the existence of De Bruijn sequences for each order together with the above properties were first proved, for the case of alphabets with two elements, by Camille Flye Sainte-Marie in 1894, whereas the generalization to larger alphabets is originally due to Tanja van Aardenne-Ehrenfest and himself.

Theorem 13 (*De Bruijn Sequence [1]*)**.** *For every positive integer n and every alphabet A with size k there exists a sequence $B(k, n)$ with length $k^n + n - 1$ that contains every n character substring over A exactly once.* □

For example, the sequence "0001011100" is a de Bruijn sequence $B(2, 3)$, because it contains all three digit sequences 000, 001, …, 110, 111 as substrings.

Theorem 14. *Every language L over the c-ary alphabet $\{0, 1, \ldots, c-1\}$ can be recognized nonconstructively with nonconstructivity $O(c^n)$.*

Proof. The idea is to do a space-efficient encoding of all possible input words.

The help-word will be taken over the $(c+1)$-ary alphabet $\{0, 1, \ldots, c-1, \mathtt{X}\}$.

For a fixed de Bruijn sequence $B(c, k)$ we define $H(k)$ to be a binary sequence of length $2(c^k + k - 1)$ with its i-th term defined to be:

$$H(k)_{2i-1} = B(c, k)_i$$

$$H(k)_{2i} = \begin{cases} 0, & \text{if } i < k \text{ or word } B(c,k)_{i-k+1}\ldots B(c,k)_i \text{ is not in } L \\ 1, & \text{otherwise.} \end{cases}$$

This definition means that $H(k)$ essentially has every k digit sequence S as a subsequence of its odd-numbered terms, moreover, having located S we can easily check if S is in L, by the definition of $H(k)$'s even-valued terms.

We define our help-word as $H(0)\mathtt{X}H(1)\mathtt{X}\ldots\mathtt{X}H(n)$.

For input word of length i automaton can position itself on the first character of $H(i)$ by skipping over i c+1's. After this step the automaton continues by checking the odd-numbered characters of $H(i)$ to find the input word. When the input word is found as a subsequence of $H(i)_j$ terms for odd j, the automaton reads the answer, which is exactly the next character on the help-word. □

Theorem 15. *There exists a language L such that it cannot be nonconstructively recognized with nonconstructivity less than* 2^n.

Proof. Take the sequence constructed by Martin-Löf in [17]. Define the language L by taking, for arbitrary n, its initial fragment of the length 2^{n+1} and using these bits as values for $ch(w_m)$ where $m = 2^{n+1}$. Martin-Löf in [17] proved that infinitely many initial fragments of his sequence have Kolmogorov complexity n and all the initial fragments of it have Kolmogorov complexity no less than $n - O(\log_2 n)$.

Assume from the contrary that there exists a nonconstructive deterministic finite automaton such that for infinitely many values of n the nonconstructivity is less than $(2^n - O(2^n))$. From the given program of the automaton and from the given nondeterministic help one can algorithmically reconstruct the values $a_1, a_2, a_3, \ldots, a_n$. Hence Kolmogorov complexity of the initial fragment $a_1, a_2, a_3, \ldots, a_n$ exceeds the length of the nonconstructive help no more than by a constant and the initial fragment has Kolmogorov complexity no higher than $(2^n - O(2^n))$. Contradiction. □

Theorem 16. *There exists a language L in a binary alphabet such that it cannot be nonconstructively recognized by a DFA with a nonconstructivity less than* $\Omega(2^n)$.

Proof. 'Martin-Löf in [17] proved that there exists an infinite sequence $a_1, a_2, a_3, \ldots$ such that infinitely many initial fragments of it have Kolmogorov complexity n and all the initial fragments of it have Kolmogorov complexity no less than $n - O(\log_2 n)$ (he also proved that there are no infinite binary sequences with a higher Kolmogorov complexity). A string of length n has Kolmogorov complexity of $O(f(n))$ means that there is no way of describing that string by using less than $O(f(n))$ bits of information.

Let $ch(w)$ be the characteristic function of language L:

$$ch(w) = \begin{cases} 0, & \text{if } w \in L \\ 1, & \text{if } w \notin L. \end{cases}$$

We define this language as:

$$ch(w_i) = i\text{th symbol in Martin-Löf sequence}$$

where w_i is the i th binary word in ordering ϵ, 0, 1, 00, 01, 10, 11, 000, 001, ...

Assume from the contrary that there exists a deterministic finite automaton such that for all values of n the nonconstructivity is less than $\Omega(2^n)$. From the given program of the automaton and from the given nondeterministic help y_n one can algorithmically reconstruct the values $a_1, a_2, a_3, \ldots, a_{2^n}$ of Martin-Löf sequence. Hence Kolmogorov complexity of the initial fragment $a_1, a_2, a_3, \ldots, a_{2^n}$ for every n exceeds the length of the nonconstructive help y_n by no more than a constant and this initial fragment of length 2^n has Kolmogorov complexity less than $\Omega(2^n)$. But this is a contradiction to properties of Martin-Löf sequence. □

This paper is an extended version of [11] presented at CIAA 2009. Section 5 is new.

Acknowledgements

The research was supported by Grant No. 09.1570 from the Latvian Council of Science and by Project 2009/0216/1DP/1.1.2.1.2/09/IPIA/VIA/004 from the European Social Fund.

References

[1] Tanja van Aardenne-Ehrenfest, Nicolaas Govert de Bruijn, Circuits and trees in oriented linear graphs, Simon Stevin 28 (1951) 203–217.
[2] Eric Bach, Jeffrey Shallit, Algorithmic Number Theory, Vol. 1, MIT Press, 1996.
[3] Jānis Bārzdiņš, Complexity of programs to determine whether natural numbers not greater than n belong to a recursively enumerable set, Soviet Mathematics Doklady 9 (1968) 1251–1254.
[4] Jānis Bārzdiņš, Kārlis Podnieks, Towards a theory of inductive inference, in: Proceedings of 2nd Symposium and Summer School on Mathematical Foundations of Computer Science, Štrbske Pleso, High Tatras, Czechoslovakia, 1973, pp. 9–15.
[5] Carsten Damm, Markus Holzer, Automata that take advice, Lecture Notes in Computer Science 969 (1995) 565–613.
[6] Paul Erdös, Some remarks on the theory of graphs, Bulletin of the American Mathematical Society 53 (4) (1947) 292–294.
[7] Yuri Leonidovich Ershov, Theory of numberings, in: E.R. Griffor (Ed.), Handbook of Computability Theory, North-Holland, Amsterdam, 1999, pp. 473–503.
[8] Rūsiņš Freivalds, Complexity of probabilistic versus deterministic automata, Lecture Notes in Computer Science 502 (1991) 565–613.
[9] Rūsiņš Freivalds, Non-constructive methods for finite probabilistic automata, Lecture Notes in Computer Science 4588 (2007) 169–180.
[10] Rūsiņš Freivalds, Non-constructive methods for finite probabilistic automata, International Journal of Foundations of Computer Science 19 (3) (2008) 565–580.
[11] Rūsiņš Freivalds, Amount of nonconstructivity in finite automata, Lecture Notes in Computer Science 5642 (2009) 227–236.
[12] Rūsiņš Freivalds, Jānis Bārzdiņš, Kārlis Podnieks, Inductive inference of recursive functions: complexity bounds, Lecture Notes in Computer Science 502 (1991) 111–155.

[13] Paul Garrett, The Mathematics of Coding Theory, Pearson Prentice Hall, Upper Saddle River, 2004.
[14] David Hilbert, Über die Theorie der algebraischen Formen, Mathematische Annalen 36 (1890) 473–534.
[15] Richard M. Karp, Richard Lipton, Turing machines that take advice, L'Enseignement Mathematique 28 (1982) 191–209.
[16] Andrei Nikolaevich Kolmogorov, Three approaches to the quantitative definition of information, Problems in Information Transmission 1 (1965) 1–7.
[17] Per Martin-Löf, The definition of random sequences, Information and Control 9 (6) (1966) 602–619.
[18] Harumichi Nishimura, Tomoyuki Yamakami, Polynomial time quantum computation with advice, Information Processing Letters 90 (4) (2004) 195–204.
[19] Kārlis Podnieks, Computational complexity of prediction strategies, in: Theory of Algorithms and Programs, vol. 3, Latvia State University, 1977, pp. 89–102 (in Russian).
[20] Richard Edwin Stearns, Juris Hartmanis, Philip M. Lewis II, Hierarchies of memory limited computations, in: Proceedings of FOCS, 1965, pp. 179–190.
[21] Kohtaro Tadaki, Tomoyuki Yamakami, Jack C.H. Lin, Theory of one tape linear time turing machines, Lecture Notes in Computer Science 2932 (2004) 335–348.
[22] Tomoyuki Yamakami, Swapping lemmas for regular and context-free languages with advice, The Computing Research Repository (CoRR), CoRR abs/0808.4122, 2008.