ESF
EIROPAS SOCIĀLAIS FONDS
EIROPAS SAVIENĪBA
IEGULDĪJUMS TAVĀ NĀKOTNĒ

Ultrametric automata and Turing machines

Rūsiņš Freivalds
(University of Latvia)

European Social Fund project Nr. 2009/0216/1DP/1.1.1.2.0/09/APIA/VIAA/044

Pascal and Fermat believed that every event of indeterminism can be described by a real number between 0 and 1 called *probability*. Quantum physics introduced a description in terms of complex numbers called *amplitude of probabilities* and later in terms of probabilistic combinations of amplitudes most conveniently described by *density matrices*.

String theory , chemistry and molecular biology have used p-adic numbers to describe measures of indeterminism.

We consider a new type of indeterministic algorithms called *ultrametric* algorithms. They are very similar to probabilistic algorithms but while probabilistic algorithms use real numbers r with $0 \leq r \leq 1$ as parameters, ultrametric algorithms use *p-adic* numbers as the parameters. Slightly simplifying the description of the definitions one can say that ultrametric algorithms are the same probabilistic algorithms, only the interpretation of the probabilities is different.

Let p be an arbitrary prime number. We will call *p-adic digit* a natural number between 0 and $p-1$ (inclusive). A *p-adic integer* is by definition a sequence $(a_i)_{i \in N}$ of p-adic digits. We write this conventionally as

$$\cdots a_i \cdots a_2 a_1 a_0$$

(that is, the a_i are written from left to right).

If n is a natural number, and

$$n = \overline{a_{k-1} a_{k-2} \cdots a_1 a_0}$$

is its p-adic representation (in other words $n = \sum_{i=0}^{k-1} a_i p^i$ with each a_i a
p-adic digit) then we identify n with the p-adic integer (a_i) with $a_i = 0$ if $i \geq k$.

If α and β turn out to be natural numbers, then their sum as a p-adic integer is no different from their sum as a natural number. So $2+2=4$ remains valid (whatever p is but if $p=2$ it would be written $\cdots 010+\cdots 010=\cdots 100$). Here is an example of a 7-adic addition:

$$\begin{array}{r} \cdots 2\,5\,1\,4\,1\,3 \\ \hline \cdots 1\,2\,1\,1\,0\,2 \\ \hline\hline \cdots 4\,0\,2\,5\,1\,5 \\ \hline \end{array}$$

This addition of p-adic integers is associative, commutative, and verifies $\alpha+0=\alpha$ for all α (recall that 0 is the p-adic integer all of whose digits are 0).

Subtraction of p-adic integers is also performed in exactly the same way as that of natural numbers in p-adic form. Note that this subtraction scheme gives us the negative integers readily: for example, subtract 1 from 0 (in the 7-adics) :

$$\begin{array}{r} \cdots 0\,0\,0\,0\,0\,0 \\ \hline \cdots 0\,0\,0\,0\,0\,1 \\ \hline\hline \cdots 6\,6\,6\,6\,6\,6 \end{array}$$

(each column borrows a 1 from the next one on the left). So $-1 = \cdots 666$ as 7-adics.

Multiplying two p-adic integers on the other hand requires some more work. Here is an example of a 7-adic multiplication:

$$
\begin{array}{r}
\cdots 2\,5\,1\,4\,1\,3 \\
\cdots 1\,2\,1\,1\,0\,2 \\
\hline
\cdots 5\,3\,3\,1\,2\,6 \\
\cdots 0\,0\,0\,0\,0 \\
\cdots 1\,4\,1\,3 \\
\cdots 4\,1\,3 \\
\cdots 2\,6 \\
\cdots 3 \\
\hline
\cdots 3\,1\,0\,4\,2\,6
\end{array}
$$

We have described above the set of p-adic integers, which we will call Z_p, with two binary operations on it, addition and multiplication. Z_p is a commutative ring, i.e., addition is associative and commutative, zero exists and satisfies the properties we wish it to satisfy, that multiplication is associative and commutative, and distributive over addition, and that 1 exists and satisfies the properties we wish it to satisfy (namely, $1\alpha = \alpha$ for all α).

Division of p-adics cannot always be performed. For example, $\frac{1}{p}$ has no meaning as a p-adic integer , that is, the equation $p\alpha = 1$ has no solution, since multiplying a p-adic integer by p always gives a p-adic integer ending in 0. There is nothing really surprising here: $\frac{1}{p}$ cannot be performed in the integers either.

Division by p is essentially the only division which cannot be performed in the p-adic integers. This is one of the reasons why the notion of p-adic integers is generalized and *p-adic numbers* are introduced. They are formal sequences of p-adic digits such that the sequence is infinite in the left-hand direction but finite in the right-hand direction. The notion of *p-adic dot* is introduced.

For example, with our usual example of $p = 7$ we show that the number $\alpha = \cdots 333334$ is the number "one half" by adding it to it itself:

$$\begin{array}{r} \cdots 3\,3\,3\,3\,3\,4 \\ \cdots 3\,3\,3\,3\,3\,4 \\ \hline\hline \cdots 0\,0\,0\,0\,0\,1 \\ \hline \end{array}$$

Thus, in the 7-adic integers, "one half" is an integer. And so are "one third" $(\cdots 44445)$, "one quarter" $(\cdots 1515152)$, "one fifth" $(\cdots 541254125413)$, "one sixth" $(\cdots 55556)$, "one eighth" $(\cdots 0606061)$, "one ninth" $(\cdots 3613613614)$, "one tenth" $(\cdots 462046205)$, "one eleventh" $(\cdots 162355043116235504312)$ and so on.

However, "one seventh", "one fourteenth" and so on, are not 7-adic integers. They are expressed as follows.

$$\cdots 0\,0\,0\,0\,.\,1$$

$$\cdots 0\,0\,0\,0\,.\,0\,1$$

It is important that p-adic numbers not being p-adic integers and irrational real numbers are kind of incompatible. It is known that no p-adic number corresponds to $\sqrt{2}, \pi, e$ and there is a continuum of p-adic numbers not corresponding to any real number. Moreover, if $p_1 \neq p_2$ then p_1-adic and p_2-adic numbers also are incompatible.

However, p-adic numbers is not merely one of generalizations of rational numbers. They are related to the notion of *absolute value* of numbers.

If X is a nonempty set, a distance, or metric, on X is a function d from pairs of elements (x, y) of X to the nonnegative real numbers such that

1. $d(x, y) = 0$ if and only if $x = y$,
2. $d(x, y) = d(y, x)$,
3. $d(x, y) \leq d(x, z) + d(z, y)$ for all $z \in X$.

Absolute value of rational number x is called *trivial* if it equals 0 for the number 0, and eqals 1 for all the other numbers.

For a prime number p, the p-adic absolute value on Q is defined as follows: any non-zero rational x, can be written uniquely as $x = p^n \frac{a}{b}$ with a, b and p pairwise coprime and $n \in \mathbb{Z}$ some integer; so we define

$$|x|_p := \begin{cases} 0, & \text{if } x = 0 \\ p^{-n}, & \text{if } x \neq 0. \end{cases}$$

In 1916 Alexander Ostrowski proved that any non-trivial absolute value on the rational numbers Q is equivalent to either the usual real absolute value or a p-adic absolute value for some prime number p.

A. Ostrowski's theorem shows that using p-adic numbers is not merely one of many possibilities to generalize the definition of deterministic algorithms but rather the only remaining possibility not yet explored.

The *norm* of an element $x \in X$ is the distance from 0:

1. $\| x \| = 0$ if and only if $x = y$,
2. $\| x.y \| = \| x \| . \| xy \|$,
3. $\| x + y \| \leq \| x \| + \| y \|$.

We know one metric on Q induced by the ordinary absolute value. However, there are other norms as well.

A norm is called *ultrametric* if the third requirement can be replaced by the stronger statement: $\| x + y \| \leq \max\{\| x \|, \| y \|\}$. Otherwise, the norm is called *Archimedean*.

Distances using the usual absolute value are called *Archimedean*, and the distances using p-adic absolute values are called *ultrametric*. P.Turakainen proved that probabilistic automata can be generalized using arbitrary real numbers instead of probabilities and the languages recognized by these Archimedean automata are the same stochastic languages.

We generalize probabilistic automata in the same way, only we use arbitrary p-adic numbers numbers instead of probabilities.

There is an important feature that distinguishes p-adic numbers from real numbers. Real numbers (both rational and irrational) are linearly ordered. p-adic numbers cannot be linearly ordered. This is why *valuations* and *norms* of p-adic numbers are considered.

The situation is similar in Quantum Computation. Quantum amplitudes are complex numbers which also cannot be linearly ordered. The counterpart of valuation for quantum algorithms is *measurement* translating a complex number $a+bi$ into a real number a^2+b^2. Norms of p-adic numbers are rational numbers.

Automata recognizing nonrecursive languages cannot be considered natural. Hence we are to restrict our definition.

Definition. Finite p-ultrametric automaton is called **integral** if all the parameters of it are p-adic integers.

Theorem. There exists a finite integral ultrametric automaton recognizing the language $\{0^n 1^n\}$.

Proof. When the automaton reads 0 it multiplies the amplitude to 2, and when it reads 1 it multiplies it to $\frac{1}{2}$.

Theorem. There is a continuum of languages recognizable by finite ultrametric automata.

Proof. Let $\beta = \cdots 2a_3 2a_2 2a_1 2a_0 2$ be an arbitrary p-adic number (not p-adic integer) where $p \geq 3$ and all $a_i \in \{0, 1\}$. Denote by B the set of all possible such β. Consider an automaton A_β with 3 states, the initial amplitudes of the states being $(\beta, -1, -1)$. The automaton is constructed to have the following property. If the input word is $2a_0 2a_1 2a_2 2a_3 2 \cdots 2a_n 2$ then the amplitude of the first state becomes $\cdots 2a_{n+4} 2a_{n+3} 2a_{n+2} 2a_{n+1} 2$. To achieve this, the automaton adds -2, multiplies to p, adds $-a_n$ and again multiplies to p.

Now let β_1 and β_2 be two different p-adic numbers. Assume that they have the same first symbols $a_m \cdots 2a_3 2a_2 2a_1 2a_0 2$ but different symbols a_{m+1} and b_{m+1}. Then the automaton accepts one of the words $a_{m+1} 2a_m \cdots 2a_3 2a_2 2a_1 2a_0 2$ and rejects the other one $b_{m+1} 2a_m \cdots 2a_3 2a_2 2a_1 2a_0 2$. Hence the languages are distinct.

Definition. A square matrix with elements being p-adic numbers is called **balanced** if for arbitrary row of the matrix the product of p-norms of the elements equals 1.

Definition. A finite ultrametric automaton is called **balanced** if all the matrices in its definition are balanced.

Theorem. If a language M can be recognized by a finite ultrametric automaton then M can be recognized also by a balanced finite ultrametric automaton.

Proof. For every state of the automaton we add its duplicate. If the given state has an amplitude γ then its duplicate has the amplitude $\frac{1}{\gamma}$.

Definition. A finite ultrametric automaton is called **regulated** if there exist constants $c > 0$ and λ such that for arbitrary input word x the norm $\lambda - c < \| \gamma \|_p < \lambda + c$. We say that the word x is accepted if $\| \gamma \|_p > \lambda$ and it is rejected if $\| \gamma \|_p \leq \lambda$.

Theorem (1) If a language M is recognized by a regulated finite ultrametric automaton then M is regular. (2) For arbitrary prime number p there is a constant c_p such that if a language M is recognized by a regulated finite p-ultrametric automaton with k states then there is a deterministic finite automaton with $(c_p)^{k \log k}$ states recognizing the language M.

We start with several examples showing that regulated p-ultrametric automata are not as simple objects as one may think.

Since the numbers 1 and 0 are also p-adic numbers, every deterministic finite automaton can be described in terms of matrices for transformation of amplitudes. Hence every regular language is recognizable by a regulated p-ultrametric automaton. There is a natural problem : are there languages for which regulated p-ultrametric automata can have smaller complexity, i.e. smaller number of states.

The following 3 theorems seem to present such an example but there is a catch: these automata are not regulated because the norm of the amplitude to be measured can be arbitrary small (for lengthy input words).

Theorem. For arbitrary prime number $p \geq 3$ the language

$$L_{p-1} = \{1^n \mid n \equiv p-1 (\bmod p)\}$$

is recognizable by a p-ultrametric finite automaton with 2 states.

Proof. A primitive root modulo n is any number g with the property that any number coprime to n is congruent to a power of g modulo n. In other words, g is a generator of the multiplicative group of integers modulo n. Existence of primitive roots modulo prime numbers was proved by Gauss. The initial amplitude 1 of a special state in our automaton is multiplied to an arbitrary primitive root modulo p. When the end-marker is read the amplitude -1 of the other state is added to this amplitude. The result has p-norm p^0 iff $n \equiv p-1$.

Theorem. For arbitrary prime number $p \geq 3$ the language

$$L_p = \{1^n \mid n \equiv p(\bmod p)\}$$

is recognizable by a p-ultrametric finite automaton with 2 states.

Proof. The value 1 of the amplitude of the second state is added to the amplitude of the accepting state at every step of reading the input word. The result has p-norm p^0 iff $n \equiv p$.

Theorem. For arbitrary natural number m there are infinitely many prime numbers p such that the language

$$L_m = \{1^n \mid n \equiv 0(\bmod m)\}$$

is recognizable by a p-ultrametric finite automaton with 2 states.

Proof. Dirichlet prime number theorem, states that for any two positive coprime integers m and d, there are infinitely many primes of the form $m+nd$, where $n \geq 0$. In other words, there are infinitely many primes which are congruent to m modulo d. The numbers of the form $mn+d$ form an arithmetic progression

$$d,\ m+d,\ 2m+d,\ 3m+d,\ \ldots,$$

and Dirichlet's theorem states that this sequence contains infinitely many prime numbers.

Let p be such a prime and g be a primitive root modulo p. Then the sequence of remainders $g, g^2, g^3, \cdots$ modulo p has period m and $n \equiv 0(\bmod m)$ is equivalent to $g^n \equiv d(\bmod p)$. Hence the automaton multiplies the amplitude of the special state to g and and adds $-d$ when reading the end-marker.

Hence we restrict ourselves to consider only ultrametric automata all parameters of which are p-adic integers. However, even such a restriction allows some non-regular languages to be recognizable.

Definition. A finite ultrametric automaton is called **regulated** if there exist constants $c > 0$ and λ such that for arbitrary input word x the norm $\lambda - c < \| \gamma \|_p < \lambda + c$. We say that the word x is accepted if $\| \gamma \|_p > \lambda$ and it is rejected if $\| \gamma \|_p \leq \lambda$.

First, we consider a sequence of languages where the advantages of ultrametric automata over deterministic ones are super-exponential but the advantages are achieved only for specific values of the prime number p.

It is known that every p-permutation can be generated as a product of sequence of two individual p-permutations:

$$a = \begin{pmatrix} 1\,2\,3\cdots & p-1 & p \\ 2\,3\,4\cdots & p & 1 \end{pmatrix}$$

$$b = \begin{pmatrix} 1\,2\,3\cdots & p-1 & p \\ 2\,1\,3\cdots & p-1 & p \end{pmatrix}$$

A string $x \in \{a, b\}^*$ is in the language M_p if the product of these p-permutations equals the trivial permutation.

Theorem. (1) For arbitrary prime p, the language M_p is recognized by a p-ultrametric finite automaton with $p+2$ states.
(2) If a deterministic finite automaton has less than $p! = c^{p.\log p}$ states then it does not recognize M_p.

Idea of the proof. The ultrametric automaton gives initial amplitudes $0, 1, 2, \cdots, p-1$ to p states of the automaton and after reading any input letter only permutes these amplitudes. After reading the endmarker from the input the automaton subtracts the values $0, 1, 2, \cdots, p-1$ from these amplitudes. Notice that if a prime number p' is such that $p' < p$ then the p'-ultrametric automaton can need more than $p+2$ states.

Theorem. If 2 is a primitive root for infinitely many distinct primes then there exists an infinite sequence of regular languages $L_1, L_2, L_3, \ldots$ in a 2-letter alphabet and a sequence of positive integers $p(1), p(2), p(3), \ldots$ such that for arbitrary j:

1. any deterministic finite automaton recognizing L_j has at least $2^{p(j)}$ states,
2. for arbitrary prime number $p \geq 3$ there is a regulated p-ultrametric finite automaton with $(4p(j) + 1)$ states recognizing L_j.

Corollary. Assume Artin's Conjecture. Then Theorem holds.

Corollary. Assume Generalized Riemann Hypothesis. Then Theorem holds.

To explain the main idea, we consider the following probabilistic automaton. The initial probability distribution:

1/8, 1/8, 1/8, 1/8, 1/8, 1/8, 1/8, 1/8, 0, 0, 0, 0, 0, 0, 0, 0

The first 8 states are accepting, the last 8 states are rejecting.

To explain the main idea, we consider the following probabilistic automaton. The initial probability distribution:

1/8, 1/8, 1/8, 1/8, 1/8, 1/8, 1/8, 1/8, 0, 0, 0, 0, 0, 0, 0, 0

The first 8 states are accepting, the last 8 states are rejecting.

There are 2^8 possible input letters. They interchange the states $q_i \rightarrow q_{i+8}$.

The input word is in the language if all the non-zero probabilities have returned to the first 8 states. It is easy to prove that any deterministic automaton needs 2^8 states for this language. Unfortunately, the probabilistic automaton has non-isolated cut-point.

To explain the main idea, we consider the following probabilistic automaton. The initial probability distribution:

1/8, 1/8, 1/8, 1/8, 1/8, 1/8, 1/8, 1/8, 0, 0, 0, 0, 0, 0, 0, 0

The first 8 states are accepting, the last 8 states are rejecting.

There are 2^8 possible input letters. They interchange the states $q_i \rightarrow q_{i+8}$.

The input word is in the language if all the non-zero probabilities have returned to the first 8 states. It is easy to prove that any deterministic automaton needs 2^8 states for this language. Unfortunately, the probabilistic automaton has non-isolated cut-point.

To explain the main idea, we consider the following probabilistic automaton. The initial probability distribution:

1/8, 1/8, 0, 1/8, 0, 0, 1/8, 1/8, 0, 0, 1/8, 0, 1/8, 1/8, 0, 0

The first 8 states are accepting, the last 8 states are rejecting.

There are 2^8 possible input letters. They interchange the states $q_i \rightarrow q_{i+8}$.

The input word is in the language if all the non-zero probabilities have returned to the first 8 states. It is easy to prove that any deterministic automaton needs 2^8 states for this language. Unfortunately, the probabilistic automaton has non-isolated cut-point.

To explain the main idea, we consider the following probabilistic automaton. The initial probability distribution:

1/8, 1/8, 0, 1/8, 0, 0, 1/8, 1/8, 0, 0, 1/8, 0, 1/8, 1/8, 0, 0

The first 8 states are accepting, the last 8 states are rejecting.

There are 2^8 possible input letters. They interchange the states $q_i \rightarrow q_{i+8}$.

The input word is in the language if all the non-zero probabilities have returned to the first 8 states. It is easy to prove that any deterministic automaton needs 2^8 states for this language. Unfortunately, the probabilistic automaton has non-isolated cut-point.

To explain the main idea, we consider the following probabilistic automaton. The initial probability distribution:

1/8, 1/8, 0, 1/8, 0, 0, 1/8, 1/8, 0, 0, 1/8, 0, 1/8, 1/8, 0, 0

The first 8 states are accepting, the last 8 states are rejecting.

There are 2^8 possible input letters. They interchange the states $q_i \rightarrow q_{i+8}$.

The input word is in the language if all the non-zero probabilities have returned to the first 8 states. It is easy to prove that any deterministic automaton needs 2^8 states for this language. Unfortunately, the probabilistic automaton has non-isolated cut-point.

To explain the main idea, we consider the following probabilistic automaton. The initial probability distribution:

1/8, 1/8, 0, 0, 1/8, 1/8, 1/8, 1/8, 0, 0, 1/8, 1/8, 0, 0, 0, 0

The first 8 states are accepting, the last 8 states are rejecting.

There are 2^8 possible input letters. They interchange the states $q_i \rightarrow q_{i+8}$.

The input word is in the language if all the non-zero probabilities have returned to the first 8 states. It is easy to prove that any deterministic automaton needs 2^8 states for this language. Unfortunately, the probabilistic automaton has non-isolated cut-point.

Linear codes is the simplest class of codes. The alphabet used is a fixed choice of a finite field $GF(q)=F_q$ with q elements. For most of this paper we consider a special case of $GF(2)=F_2$. These codes are binary codes.

A generating matrix G for a linear [n, k] code over F_q is a k-by-n matrix with entries in the finite field F_q, whose rows are linearly independent. The linear code corresponding to the matrix G consists of all the q^k possible linear combinations of rows of G. The requirement of linear independence is equivalent to saying that all the q^k linear combinations are distinct.

Linear codes is the simplest class of codes. The alphabet used is a fixed choice of a finite field $GF(q)=F_q$ with q elements. For most of this paper we consider a special case of $GF(2)=F_2$. These codes are binary codes.

A generating matrix G for a linear [n, k] code over F_q is a k-by-n matrix with entries in the finite field F_q, whose rows are linearly independent. The linear code corresponding to the matrix G consists of all the q^k possible linear combinations of rows of G. The requirement of linear independence is equivalent to saying that all the q^k linear combinations are distinct.

$$\begin{pmatrix} 001101001 \\ 101011100 \\ 101110001 \end{pmatrix}$$

Linear codes is the simplest class of codes. The alphabet used is a fixed choice of a finite field GF(q)=F_q with q elements. For most of this paper we consider a special case of GF(2)=F_2. These codes are binary codes.

A generating matrix G for a linear [n, k] code over F_q is a k-by-n matrix with entries in the finite field F_q, whose rows are linearly independent. The linear code corresponding to the matrix G consists of all the q^k possible linear combinations of rows of G. The requirement of linear independence is equivalent to saying that all the q^k linear combinations are distinct.

$$\begin{pmatrix} 001101001 \\ 101011100 \\ 101110001 \end{pmatrix} \qquad \begin{matrix} 0 \\ 1 \\ 1 \end{matrix} \quad \rightarrow \quad 000101101$$

The linear combinations of the rows in G are called codewords. However we are interested in something more. We need to have the codewords not merely distinct but also to be removed as far as possible each from another in terms of Hamming distance. Hamming distance between two vectors $v=(v_1, ..., v_n)$ and $w=(w_1, ..., w_n)$ is the number of indices i such that $v_i \neq w_i$.

The textbook *P.Garret „The Mathematics of Coding Theory”(2004)* contains

Theorem A. For any integer n≥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 has a serious defect. It is non-constructive. It means that we cannot find these codes or describe them in a useful manner. This is why P.Garret calls them mirage codes.

Definition. A generating matrix G of a linear code is called *cyclic* if along with an arbitrary row $(v_1, v_2, v_3, ..., v_n)$ the matrix G contains also a row $(v_2, v_3,...,v_n,v_1)$.

We would wish to prove a reasonable counterpart of Theorem A for cyclic mirage codes, but this attempt fails.

Instead we construct a slightly more complicated structure of mirage codes for which a counterpart of Theorem A can be proved.

We consider binary generating matrices. Let p be an odd prime number, and x be a binary word of the length p. The generating matrix G(p, x) has p rows and 2p columns. Let $x = x_1\ x_2\ x_3\ ...\ x_p$.

The first p columns (and all p rows) make 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) make a cyclic matrix with $x = x_1\ x_2\ x_3\ ...\ x_p$ as the first row, $x_p\ x_1\ x_2\ x_3\ ...\ x_{p-1}$ as the second row, and so on.

$1\ 0\ 0\ 0\ ...\ 0\ x_1\ \ x_2\ \ x_3\ \ x_4 ...\ x_p$

$0\ 1\ 0\ 0\ ...\ 0\ x_p\ \ x_1\ \ x_2\ \ x_3\ ...\ x_{p-1}$

$0\ 0\ 1\ 0\ ...\ 0\ x_{p-1}\ x_p\ \ x_1\ \ x_2\ ...\ x_{p-2}$

.....

$0\ 0\ 0\ 0\ ...\ \ 1\ x_2\ \ x_3\ \ x_4\ \ x_5 ...\ x_1$

Definition. 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. We say that a numbering ψ is *reducible* to the numbering ξ if there exists a total recursive function $f(n)$ such that, for all n and x, $\psi_n(x) = \xi_{f(n)}(x)$.

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

Definition. We say that a Gödel numbering κ 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:

-for all n and x, $\psi_n(x) = \kappa_{f(n)}(x)$,

- for all n, $f(n) \leq cn + d$.

There exist many distinct Kolmogorov numberings. We now fix one of them and denote it by κ . Since Kolmogorov numberings give indices for all partial recursive functions, for arbitrary x and p, there is an i such that $\kappa_i(p)=x$. Let i(x, p) be the minimal i such that $\kappa_i(p) = x$. It is easy to see that if $x_1 \neq x_2$, then $i(x_1,p) \neq i(x_2,p)$. We consider all binary words x of the length p and denote by x(p) the word x such that i(x,p) exceeds i(y,p) for all binary words y of the length p different from x. It is obvious that $i \geq 2^{p-1}$.

Until now we considered generating matrices $G(p, x)$ for independently chosen p and x. From now on we consider only odd primes p such that 2 is a primitive root modulo p and the matrices $G(p, x(p))$. The essence of the proof is that if p is **sufficiently large** and $x(p)$ is **the p-digit word with the maximum Kolmogorov complexity**, then Hamming distances between arbitrary two codewords in this linear code is at least $\frac{4p}{19}$.

Under the input symbol a the states are permuted as follows:

$q_1 \to q_3$	$q_2 \to q_4$	$q_{2p+1} \to q_{2p+3}$	$q_{2p+2} \to q_{2p+4}$
$q_3 \to q_5$	$q_4 \to q_6$	$q_{2p+3} \to q_{2p+5}$	$q_{2p+4} \to q_{2p+6}$
$q_5 \to q_7$	$q_6 \to q_8$	$q_{2p+5} \to q_{2p+7}$	$q_{2p+6} \to q_{2p+8}$
$\dots$	$\dots$	$\dots$	$\dots$
$q_{2p-3} \to q_{2p-1}$	$q_{2p-2} \to q_{2p}$	$q_{4p-3} \to q_{4p-1}$	$q_{4p-2} \to q_{4p}$
$q_{2p-1} \to q_1$	$q_{2p} \to q_2$	$q_{4p-1} \to q_{2p+1}$	$q_{4p} \to q_{2p+2}$

The permutation of the states under the input symbol b depends on $G(p, x(p))$. Let be

$$G(p, x(p)) = \begin{pmatrix} g_{11} & g_{12} & \cdots & g_{1\ 2p} \\ g_{21} & g_{22} & \cdots & g_{2\ 2p} \\ \cdots & \cdots & \cdots & \cdots \\ g_{p1} & g_{p2} & \cdots & g_{p\ 2p} \end{pmatrix}$$

For arbitrary $i \in \{1, 2, \dots, p\}$,

$$\begin{cases} q_{2i-1} \to q_{2i-1} & , \text{if } g_{1i} = 0 \\ q_{2i} \to q_{2i} & , \text{if } g_{1i} = 0 \\ q_{2i-1} \to q_{2i} & , \text{if } g_{1i} = 1 \\ q_{2i} \to q_{2i-1} & , \text{if } g_{1i} = 1. \end{cases}$$

We introduce a language

$$L_{G(p,x(p))} = \{w | CW(w) = 000\ldots0\}.$$

Lemma. If 2 is a primitive root modulo p and p is sufficiently large, then the automaton $R(p)$ recognizes the language $L_{G(p,x(p))}$ with the probability $\frac{19}{36}$.

Lemma. For arbitrary p and arbitrary deterministic finite automaton A recognizing $L_{G(p,x(p))}$ the number of states of A is no less than 2^p.

Theorem. If 2 is a primitive root for infinitely many distinct primes then there exists an infinite sequence of regular languages $L_1, L_2, L_3, \ldots$ in a 2-letter alphabet and a sequence of positive integers $p(1), p(2), p(3), \ldots$ such that for arbitrary j:

1. any deterministic finite automaton recognizing L_j has at least $2^{p(j)}$ states,
2. for arbitrary prime number $p \geq 3$ there is a regulated p-ultrametric finite automaton with $(4p(j) + 1)$ states recognizing L_j.

Corollary. Assume Artin's Conjecture. Then Theorem holds.

Corollary. Assume Generalized Riemann Hypothesis. Then Theorem holds.

Theorem. There exists a language T with the following properties.

(1) There is a regulated 3-ultrametric 1-way pushdown automaton recognizing the language T.
(2) No deterministic 1-way pushdown automata can recognize the language T.
(3) No probabilistic 1-way pushdown automata can recognize the language T can have bounded error.

Let $A = \{a, b, c, d, e, f, g, h, k, l, m, p, q, r, s, t, u, v\}$. Now we consider a language T in the alphabet $A \cup \{\#\}$ for which both the deterministic and probabilistic 1-way pushdown automata cannot recognize the language but there exists an ultrametric 1-way pushdown automaton recognizing it.

The language T is defined as the set of all the words x in the input alphabet such that either x is in all 9 languages T_i described below or in exactly 6 of them or in exactly 3 of them or in none of them where

$$T_1 = \{x\#y \mid x \in A^* \wedge y \in A^* \wedge proj_{ab}(x) = proj_{ab}(y)\},$$

$$T_2 = \{x\#y \mid x \in A^* \wedge y \in A^* \wedge proj_{cd}(x) = proj_{cd}(y)\},$$

$$T_3 = \{x\#y \mid x \in A^* \wedge y \in A^* \wedge proj_{ef}(x) = proj_{ef}(y)\},$$

$$T_4 = \{x\#y \mid x \in A^* \wedge y \in A^* \wedge proj_{gh}(x) = proj_{gh}(y)\},$$

$$T_5 = \{x\#y \mid x \in A^* \wedge y \in A^* \wedge proj_{kl}(x) = proj_{kl}(y)\},$$

$$T_6 = \{x\#y \mid x \in A^* \wedge y \in A^* \wedge proj_{mp}(x) = proj_{mp}(y)\},$$

$$T_7 = \{x\#y \mid x \in A^* \wedge y \in A^* \wedge proj_{qr}(x) = proj_{qr}(y)\},$$

$$T_8 = \{x\#y \mid x \in A^* \wedge y \in A^* \wedge proj_{st}(x) = proj_{st}(y)\},$$

$$T_9 = \{x\#y \mid x \in A^* \wedge y \in A^* \wedge proj_{uv}(x) = proj_{uv}(y)\}.$$

Theorem. If a language M is recognizable by a probabilistic Turing machine in a polynomial time then for arbitrary $p \geq 3$ there is a p-ultrametric Turing machine recognizing M in a polynomial time.

Proof. The class PP of all languages recognizable in a polynomial time has natural complete problems, for example, $MAJSAT$. $MAJSAT$ is a decision problem in which one is given a Boolean formula F. The answer must be YES if more than half of all assignments $x_1, x_2, \cdots, x_n$ make F true and NO otherwise. Hence M is reducible to $MAJSAT$ in deterministic polynomial time. On the other hand, $MAJSAT$ is recognizable by a p-ultrametric Turing machine in a polynomial time. This machine considers in parallel all possible assignments for $x_1, x_2, \cdots, x_n$ and adds a p-adic number 2^{-n} to the amplitude α of a special state. F is in $MAJSAT$ iff the resulting amplitude α has p-norm 0.

We denote by pUP the class of all languages recognizable by p-ultrametric Turing machines in a polynomial time. This is a large class of languages.

Theorem. For arbitrary prime number $p \geq 3$ there is a pUP-complete language.

Riemann surface is a notion useful to study functions of complex variable.

Definition. A *discrete Riemann surface* on the rectangle $[a, b] \times [c, d]$ is a map from (x, y, z) (where $x \in [a, b]$, $y \in [c, d]$ and z is a string of symbols from a finite alphabet Σ whose length equals $y - c$) to a finite alphabet Δ. For each triple its *neighbors* are defined as the triples:
(1) (x, y', z) where either $y' = y + 1$ or $y' = y + 1$,
(2) (x', y, z') where either $x' = x - 1$ and z' is z with the last symbol omitted, or $x' = x + 1$ and z' is z with the one symbol attached at its end.

Definition. A *discrete Dirichlet condition* is a 5-tuple consisting of: (1) a map from (x, y) where $y = c$ to Δ, (2) a map from (x, y) where $y = d$ to Δ, (3) (x, y) where $x = a$ to Δ, (4) (x, y) where $x = b$ to Δ, and (5) neighboring conditions that may forbid some simultaneous maps of neighboring triples.

Definition. The *discrete Dirichlet problem* is whether or not a Riemann surface is possible consistent with the given discrete Dirichlet condition.

Theorem. For arbitrary prime number $p \geq 3$, there is a $pU\,P$-complete language.

Idea of the proof. The language consists of all discrete Dirichlet conditions such that the discrete Dirichlet problem has a positive answer. The map in the Riemann surface can be used to describe the work of a ultrametric Turing machine. The symbols of Δ for all possible values of x for a fixed y and z describe the configuration of the tape at the moment y with the choices z made before the moment y and the amplitudes accumulated. The discrete Dirichlet problem asks whether the ultrametric machine accepts the input word. The difference $d - c$ represents the computation time allowed.

Thank you