Baltic J. Modern Computing, Vol. 4 (2016), No. 3, pp. 561–582

Unconventional Finite Automata and Algorithms

Kaspars BALODIS[1,2]

[1] Faculty of Computing, University of Latvia, Raiņa bulvāris 19, Riga, LV-1586, Latvia
[2] Institute of Mathematics and Computer Science, University of Latvia, Raiņa bulvāris 29, Riga, LV-1459, Latvia

balodis.kaspars@lu.lv

Abstract. We investigate several unconventional models of finite automata and algorithms.
We show that two-way alternating automata can be smaller than fast bounded-error probabilistic automata. We introduce ultrametric finite automata which use p-adic numbers to describe the branching process of the computation. We examine the size complexity of all the above-mentioned automata for the counting problem. We also examine two-way frequency finite automata.
We define ultrametric query algorithms and examine ultrametric query complexity for Boolean functions. We generalize the notion of frequency computation by requiring some structure for the correct outputs.

Keywords: unconventional computing, finite automata, descriptional complexity, p-adic numbers, ultrametric automata, frequency computing

Introduction

Typically computer is viewed as a deterministic machine. However scientists have also extensively studied other models of computation. Some of them are based on nondeterminism and alternation, some of them are intended to be physically realizable or are inspired by natural processes. Such examples include randomized and quantum computing, cellular automata, DNA computing and neural networks. However others are just a mathematically elegant formal systems that describe a process which can be viewed as a computation (lambda calculus, Markov algorithms, Wang tiles, etc.).

There is no globally accepted definition of what type of computation should be considered unconventional. For example the notion of nondeterminism is so widely used in computer science that practically nobody would consider it unconventional. However, as noted by Arora and Barak (2009) in their textbook "Computational Complexity: A Modern Approach": "One should note that, unlike standard Turing machines, nondeterministic Turing machines are not intended to model any physically realizable computation device." Nevertheless, despite its physical nonrealizability the nondeterministic

Turing machine is undoubtedly a very useful concept and in fact one of central in computational complexity theory. Therefore, we can see that the physical realizability of a model of computation is not a determining indicator of conventionality.

Also it is not uncommon in mathematics that a new concept or result is not readily applicable at the time of its discovery, but may find its use several centuries later. As a classical example, it is highly unlikely that the ancient Greek mathematicians when thinking about prime numbers could have imagined their use in cryptography more than 2000 years later. Sometimes results in one field have unexpected applications in other fields. For example, the methods of quantum computing have in many occasions turned out useful to prove classical results in fields that have nothing to do with quantum computing (see Drucker and Wolf (2009) for a survey).

Therefore it is not impossible that unconventional models of computation, however unimaginable as physical devices, can later turn out useful in unexpected ways.

Subject and Goals of the Research

Different degrees of unconventionality are possible. Unconventional computation can be mildly unconventional by simply considering a computation in a setting that is not usually (conventionally) considered, or it might be highly unconventional by introducing some novel, almost unimaginable type of computation.

The goal of the research was to examine different unconventional computational models and prove new, previously unknown properties about them. A large part of the work is devoted to unconventional finite automata. We focus on three unconventional types of computation – probabilistic, ultrametric, and frequency. We consider several models of computation with varying degrees of unconventionality, namely:

- probabilistic and alternating two-way finite automata,
- ultrametric finite automata and query algorithms,
- frequency finite automata and structured frequency algorithms.

Arguably the most conventional computation type considered is the two-way finite automata. Traditionally finite automata have been considered in one-way setting and with focus on the languages that can be recognized. Indeed, if one cares only about the class of languages recognizable by finite automata then there is no need to consider nondeterministic or alternating automata or two-way versions of them because all of those models can recognize exactly the same class of languages – the class of regular languages. However the situation changes remarkably if we are interested in the state complexity of the automata.

As noted by Kapoutsis, a rich and meaningful complexity theory arises when one considers the size complexity of two-way finite automata. It is quite similar to the Turing machine time and space complexity theory and contains all the standard features of a complexity theory such as complexity classes, reductions, complete problems and even alternating hierarchy. Moreover, the connection to Turing machine complexity theory is not just conceptual – some results about polynomial-size deterministic and nondeterministic automata would directly translate to space-bounded Turing machine complexity theory, specifically the $L \stackrel{?}{=} NL$ question. For an overview of this two-way

finite automata size complexity theory, see, for example, Kapoutsis (2009), Kapoutsis (2012), and Královic (2010). We prove a relationship between the two of the least conventional types of automata in this theory, namely the alternating and probabilistic automata.

For every prime number p the p-adic numbers are a completion of the field of rational numbers with respect to the p-adic metric, in the same way as real numbers are a completion of rational numbers with respect to the usual metric. A theorem by Ostrowski shows that every non-trivial absolute value on the rational numbers is equivalent to either the usual real absolute value or a p-adic absolute value. Therefore p-adic numbers can be considered as one of two possible natural extensions of the field of rational numbers (the other one being usual real numbers). p-adic numbers have been used in various ways by theoretical physicists in attempts to understand the nature of fundamental physics. However, the use of p-adic numbers in computational models is a relatively new concept recently introduced by Freivalds. In ultrametric computation p-adic numbers are used to describe the branching of a computation. In this way the p-adic numbers have been used to define ultrametric automata, ultrametric Turing machines, ultrametric query algorithms, and ultrametric learning algorithms. The ultrametric computations, unlike deterministic, randomized or even quantum computations, are not meant to be realizable as physical devices, at least we are not aware of a practical way to implement them. Nevertheless, they are equally interesting as an abstract computational model. In this paper we define ultrametric finite automata and consider various properties that arise out of their definition. Additionally we introduce the model of ultrametric query algorithms which is similar to probabilistic and quantum query algorithms. We prove results about ultrametric query complexity of Boolean functions.

Frequency computation is a notion that was introduced in 1960 by G. Rose. A frequency algorithm (m, n)-computes a function f if given any n distinct inputs $x_1, \ldots, x_n$ it produces n outputs $y_1, \ldots, y_n$ for which at least m of the equations $y_i = f(x_i)$ hold, i.e., the algorithm is required to output the correct answer on at least m of n different inputs. While for some this notion might already seem rather unconventional by itself, we modify it in two different ways therefore imparting an additional degree of unconventionality. The first way involves finite automata. While frequency finite automata have already been considered earlier, they have been considered only in the setting of one-way automaton which simultaneously reads a symbol from each of the input words. We introduce two-way frequency finite automata – a model which has not been considered earlier in the literature. We show their relationship with finite automata with linearly bounded counters and the complexity class *LOGSPACE*.

The other modification is changing the definition of frequency computation by requiring some structure for the correct answers therefore obtaining the so-called structured frequency computation. It is also a new model which has not been considered previously. In this setting we examine which structures allow the computing of recursive sets and which – nonrecursive. We also investigate graph frequency computation where the size of the structures is limited to 2 so they can be represented as the edges of a graph.

The Organization of the Paper

The paper contains six sections each of which is devoted to one unconventional computation model. We start by considering finite automata.

In the first section we consider two-way probabilistic and alternating automata which are the most conventional computation type considered in this paper. In this section we do not define new computation models, but instead work in a theoretical framework introduced by Sakoda and Sipser and further developed by Kapoutsis, Královič and others. We shortly describe the complexity theory of two-way automata and give definitions for two-way probabilistic and alternating finite automata and the corresponding complexity classes. Then we prove the main result of the section – show a language that can be recognized with a linear-size one-way alternating automaton, but cannot be recognized with any polynomial-size probabilistic two-way automaton. This is a previously unknown, novel result and it shows that the corresponding complexity classes $2P_2$ and $2A$ are not equal.

In the second section we consider ultrametric finite automata. We describe the p-adic numbers and give definitions for ultrametric finite automata and investigate their properties. We show that with a regulated ultrametric automaton only regular languages can be recognized and we give a bound on the number of states for an equivalent deterministic automaton. We show that for some languages ultrametric automata can be smaller than deterministic automata. We also compare different definitions for ultrametric automata. In the second part of the section we consider multi-head ultrametric automata. We show that ultrametric one-way automata can recognize languages not recognizable by any k-head nondeterministic automata. For two-way automata we prove that adding an extra head to the automaton increases the class of recognizable languages.

In the third section we focus on counting with finite automata by examining how many states are needed for different models of automata to recognize the language $C_n = \{1^n\}$. At first we list known results about deterministic, nondeterministic and alternating finite automata, and then proceed to prove new results about probabilistic and ultrametric finite automata. We show an optimal 3-state one-way probabilistic automaton. We show how to construct a constant-sized two-way probabilistic automaton with the number of states not depending on the required probability for the correct answer. We also show an optimal 2-state ultrametric automaton.

In the fourth section we consider two-way frequency automata. We introduce their definition and show that with frequency (m, n) it is possible to recognize any language recognizable with a two-way automaton with $n - m$ linearly bounded counters. We also show that any language from the class *LOGSPACE* can be recognized by some frequency automaton.

Next, we consider unconventional algorithms. In the fifth section we introduce a new type of algorithms – ultrametric query algorithms which use p-adic numbers to describe the branching of an algorithm. We show results about the ultrametric query complexity. We show that the exact ultrametric query complexity is at most the polynomial degree of a Boolean function. For several functions we show ultrametric query algorithms with complexity 1.

In the sixth section we introduce structured frequency computation. We prove that with overlapping structures only recursive sets can be recognized and show that the size of overlapping structures is at least $\sqrt{n}$. We also show how to construct structures which nearly achieves this bound by using projective planes. In the second part of the section we consider the special case of graph structures and categorize which graphs allow only computation of recursive sets.

In conclusion we list some open problems and further research directions in each of the considered topics.

1 Classical Automata

The most conventional computation type considered in this paper is the two-way finite automata. Traditionally finite automata have been considered in deterministic one-way setting and with the focus on the languages that can be recognized. Indeed, if one cares only about the class of languages recognizable by finite automata then there is no need to consider nondeterministic or alternating automata or two-way versions of them because all of those models can recognize exactly the same class of languages – the class of regular languages. However the situation changes remarkably if we are interested in the state complexity of the automata.

This section is based on Balodis (2013).

1.1 Two-Way Finite Automata Complexity Theory

In this section we work in a theoretical framework of state complexity of two-way finite automata introduced by Sakoda and Sipser (1978) and developed by Kapoutsis and others (for an overview, see, for example, Kapoutsis (2009), Kapoutsis (2012), and Královic (2010)).

As noted by Kapoutsis, a rich and meaningful complexity theory arises when one considers the size complexity of two-way finite automata. It is quite similar to the Turing machine time and space complexity theory and contains all the standard features of a complexity theory such as complexity classes, reductions, complete problems and even alternating hierarchy.

Following this framework, instead of individual languages and finite automata we consider families of languages and families of finite automata. We say that a family of automata $\mathcal{A} = (A_1, A_2, A_3, \dots)$ recognizes a family of languages $\mathcal{L} = (L_1, L_2, L_3, \dots)$, if for every $h \geq 1$ the automaton A_h recognizes L_h. Special interest is given to families of automata where the increase in the number of states can be bounded by a polynomial.

The automata are called small if there is a polynomial p such that for each h the automaton $\mathcal{M}_h$ has at most $p(h)$ states.

2D is the class of families of problems solvable by a family of small two-way deterministic automata (2DFAs):

$$2D = \left\{ (\mathcal{L}_h)_{h\geq 1} \,\middle|\, \begin{array}{c}\text{there exist 2DFAs } (\mathcal{M}_h)_{h\geq 1} \text{ and polynomial } p \text{ such that} \\ \mathcal{M}_h \text{ solves } \mathcal{L}_h \text{ with at most } p(h) \text{ states, for all } h\end{array} \right\}$$

2N is the class of families of problems solvable by a family of small nondeterministic finite automata (2NFAs) and co-2N is the class of families of problems whose complements are solvable by a family of small 2NFAs. Analogous classes for one-way automata are 1D, 1N and co-1N.

The increase in complexity when going from a deterministic to a nondeterministic model has been an important topic in computer science. Most notorious of the kind is the question whether polynomial-time Turing machines recognize the same class of languages as polynomial-time nondeterministic Turing machines, i.e., the $P \stackrel{?}{=} NP$ question.

In two-way finite automata complexity theory the central question is similar – whether the class 2D is equal to the class 2N. It is analogous to the $P \stackrel{?}{=} NP$ and $L \stackrel{?}{=} NL$ questions in the Turing machine time and space complexity theories.

Moreover the connection to the complexity theory of Turing machines is not just conceptual. It has been known for a long time that proving $2D \neq 2N$ using only polynomially long instances would imply that $L \neq NL$ (by Berman and Lingas (1977)).

1.2 Alternating and Probabilistic Automata

In this subsection we prove a relationship between the two of the least conventional types of automata in this theory, namely the alternating and probabilistic automata.

We use the well-known standard definition for alternating automata.

Definition 1. *A two-way alternating finite automaton (2AFA) is a tuple* $\mathcal{M} = (Q_\exists, Q_\forall, \Sigma, \delta, q_0, F)$*, where*

$Q_\exists$ *and* $Q_\forall$ *are finite sets of existential and universal states, respectively, with* $Q_\exists \cap Q_\forall = \emptyset$ *and* $Q = Q_\exists \cup Q_\forall$*,*
Σ *is the input alphabet,*
$\delta : Q \times (\Sigma \cup \{\vdash, \dashv\}) \to 2^{Q \times \{L,N,R\}}$ *is the transition function, where* $\vdash, \dashv \notin \Sigma$ *are the left and right endmarkers, and L, N, R denote the head movement,*
$q_0 \in Q$ *is the starting state, and*
$F \subseteq Q$ *is the set of accepting states (also called final states).*

The input word w is presented to the automaton enclosed by the endmarkers as $\vdash w \dashv$.

The automaton $\mathcal{M}$ is said to be one-way (1AFA) if its input head motions are restricted to R and N. For one-way machines, we usually do not embed the input between endmarkers.

An alternation is a computation step in which the automaton switches from a state $q \in Q_\exists$ to a state $q' \in Q_\forall$ or from a state $q \in Q_\forall$ to a state $q' \in Q_\exists$.

A nondeterministic automaton (2NFA) is a special case of 2AFA $\mathcal{M} = (Q_\exists, Q_\forall, \Sigma, \delta, q_0, F)$ for which $Q_\forall = \emptyset$. A deterministic automaton (2DFA) is a special case of 2NFA for which the transition function δ assigns at most one successor state and movement direction for each state and input symbol. Similarly the automata are one-way (1NFA and 1DFA, respectively) if their input head motions are restricted to R and N.

For probabilistic automata (PFA) we also use the standard well-known definition.

Definition 2. *A two-way probabilistic finite automaton (2PFA) is a tuple* $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ *where*

Q is the finite set of states,
Σ is the input alphabet,
$\delta : Q \times (\Sigma \cup \{\vdash, \dashv\}) \times Q \times \{L, N, R\} \to P$ is the probabilistic transition function,
$\vdash, \dashv \notin \Sigma$ are the left and right endmarkers, respectively, and L, N, R denote the head movement, and $P \subseteq [0, 1]$ is the set of allowed probabilities,
$q_0 : Q \to P$ is the starting distribution, and
$F \subseteq Q$ is the set of accepting states.

For 2PFAs the transition function $\delta : Q \times (\Sigma \cup \{\vdash, \dashv\}) \times Q \times \{L, N, R\} \to P$ assigns for each possible transition the probability of making this transition and $q_0 : Q \to P$ shows the starting probability distribution.

If $P = \mathbb{Q} \cap [0, 1]$, where $\mathbb{Q}$ is the set of rational numbers, then we call the automaton *rational*. If $P = \{0, \frac{1}{2}, 1\}$, then we call the automaton *coin-flipping*. The 2PFA $\mathcal{M}$ is said to be one-way (1PFA) if its input head motions are restricted to R and N.

Precise descriptions of how the automata work can be found in the work or in the literature of automata theory.

Let $\mathcal{A}(x)$ denote the probability that a PFA $\mathcal{A}$ accepts the word x. We say that a PFA $\mathcal{A}$ recognizes language L with cutpoint $\lambda \in [0, 1]$ if $\forall x \in L\ \mathcal{A}(x) > \lambda$ and $\forall x \notin L\ \mathcal{A}(x) < \lambda$. We say that a PFA $\mathcal{A}$ recognizes language L with an isolated cutpoint $\lambda \in [0, 1]$ if there exists $\delta > 0$ (called isolation radius) such that $\forall x \in L\ \mathcal{A}(x) > \lambda + \delta$ and $\forall x \notin L\ \mathcal{A}(x) < \lambda - \delta$. The PFAs with isolated cutpoint are called $\mathrm{P_2FA}$s.

We say that a 2PFA $\mathcal{M}$ is fast if there exists a polynomial p such that for each word $w \in \Sigma^*$ the expected running time of $\mathcal{M}$ is upper-bounded by $p(|w|)$.

Similarly as for other types of automata, we define classes of families of languages corresponding to the alternating and probabilistic finite automata.

Definition 3.

$$2A = \left\{ (\mathcal{L}_h)_{h \geq 1} \middle| \begin{array}{c} \textit{there exist } 2AFAs\ (M_h)_{h \geq 1} \textit{ and polynomial } p \textit{ such that} \\ M_h \textit{ solves } \mathcal{L}_h \textit{ using at most } p(h) \textit{ states for all } h \end{array} \right\}$$

For all $k \geq 1$:

$$2\Sigma_k = \left\{ (\mathcal{L}_h)_{h \geq 1} \middle| \begin{array}{c} \textit{there exist } 2AFAs\ (M_h)_{h \geq 1} \textit{ and polynomial } p \textit{ such that} \\ M_h \textit{ solves } \mathcal{L}_h \textit{ starting in an existential state and using} \\ \textit{at most } k-1 \textit{ alternations and } p(h) \textit{ states for all } h \end{array} \right\}$$

$$2\Pi_k = \left\{ (\mathcal{L}_h)_{h \geq 1} \middle| \begin{array}{c} \textit{there exist } 2AFAs\ (M_h)_{h \geq 1} \textit{ and polynomial } p \textit{ such that} \\ M_h \textit{ solves } \mathcal{L}_h \textit{ starting in an universal state and using} \\ \textit{at most } k-1 \textit{ alternations and } p(h) \textit{ states for all } h \end{array} \right\}$$

$1\Sigma_k$ and $1\Pi_k$ denote the corresponding classes for one-way automata.

The complexity class of problems solvable by a family of small and fast rational $\mathrm{2P_2FA}$s is $\mathrm{2P_2}$.

Definition 4.

$$2P_2 = \left\{ (\mathcal{L}_h)_{h \geq 1} \middle| \begin{array}{c} \textit{there exist rational } 2P_2FAs\ (M_h)_{h \geq 1} \textit{ and polynomials } p, q, r \textit{ such that} \\ M_h \textit{ solves } \mathcal{L}_h \textit{ with isolation radius } \frac{1}{r(h,n)} \textit{ using at most } p(h) \textit{ states and} \\ q(h, n) \textit{ steps on average, for all } h \textit{ and all } n \textit{ and all } n\textit{-long instances} \end{array} \right\}$$

We also define two classes for less restricted types of 2PFAs. The complexity class for unrestricted runtime is $2P_2X$. If we drop the requirement for the cutpoint to be isolated then the corresponding complexity class is 2P.

As the main result of the section we show two families of languages EVALUATE-DNF-FUNCTION and EVALUATE-CNF-FUNCTION, each of which can be recognized by a family of linear-size 1AFAs which make only 1 alternation (and start in an existential or universal state respectively), but for every family of fast bounded-error 2PFAs the state increase is more than polynomial. So for small automata even very restricted alternation (one-way and using only 1 alternation) gives advantage over a natural (fast and bounded-error) class of probabilistic automata.

Theorem 1. *There exists a family of 1AFAs $(\mathcal{A}_h)_{h\geq 1}$ with $2h+3$ states that solves the problem* EVALUATE-DNF-FUNCTION *starting in an existential state and using 1 alternation.*

Theorem 2. *There exists a family of 1AFAs $(\mathcal{A}'_h)_{h\geq 1}$ with $2h+3$ states that solves the problem* EVALUATE-CNF-FUNCTION *starting in a universal state and using 1 alternation.*

Theorem 3. *There exists no family of 2NFAs of fast $2P_2$FAs that solves* EVALUATE-DNF-FUNCTION *or* EVALUATE-CNF-FUNCTION *with a polynomial number of states.*

Corollary 1. *Neither $1\Sigma_2$ nor $1\Pi_2$ is contained in $2N \cup co\text{-}2N \cup 2P_2$.*

This result shows a previously unknown relationship between complexity classes in the theoretical framework of Sakoda, Sipser and Kapoutsis. Is solves an open problem in this model thereby increasing our current understanding about two-way finite automata complexity.

This result is also interesting because in the literature there are many results which show advantages of probabilistic automata over non-probabilistic ones. However this result is one of the very few showing advantage in the opposite direction – that a class of non-probabilistic finite automata can be more powerful than a class of probabilistic finite automata.

In this section we also show results in the opposite direction, however in a somehow weaker form. If we do not require the automata to be fast or the isolation radius to decrease no faster than polynomially then such small automata can recognize families of languages which cannot be recognized by small 2AFAs.

Corollary 2. *Neither 1P nor $2P_2X$ is contained in 2A.*

An open problem is whether it is true also if the probabilistic automata is left with both of those restrictions.

2 Ultrametric Automata

p-adic numbers have been used in various ways by theoretical physicists in attempts to understand the nature of fundamental physics. However the use of p-adic numbers in

computer science is a relatively new concept recently introduced by Freivalds. p-adics have been used to define ultrametric finite automata, ultrametric Turing machines, ultrametric query algorithms and ultrametric learning algorithms. The ultrametric computations, unlike deterministic, randomized or even quantum computations, are not meant to be realizable as physical devices, at least we are not aware of a practical way to implement them. Nevertheless, they are equally interesting as an abstract computational model.

This section is partly based on Balodis et al. (2013).

2.1 *p*-adic Numbers

For every prime number p the p-adic numbers are a completion of the field of rational numbers with respect to the p-adic metric, in the same way as real numbers are a completion of rational numbers with respect to the usual metric. A theorem by Ostrowski shows that every non-trivial absolute value on the rational numbers is equivalent to either the usual real absolute value or a p-adic absolute value for some prime p. Therefore p-adic numbers can be considered as one of two possible natural extensions of the field of rational numbers (the other one being usual real numbers).

For real numbers it is common to have an infinite number of digits to the right of the point in the decimal (or any other base) notation. This infinite sequence can be periodic, e.g., $\frac{1}{3} = 0.3333\ldots$ or aperiodic, e.g., $\pi = 3.1415926\ldots$. The p-adic numbers have it the other way – to the right of the point there is only a finite number of digits, but an infinite number to the left of the point.

Let p be an arbitrary prime number. A p-adic digit is a natural number between 0 and $p-1$ (inclusive). A p-adic integer is a sequence $\cdots a_i \cdots a_2 a_1 a_0$ of p-adic digits which is infinite to the left side.

Especially significant is the p-adic norm.

Definition 5. *The p-norm for a rational number $\alpha = \pm 2^{\alpha_2} 3^{\alpha_3} 5^{\alpha_5} 7^{\alpha_7} \cdots$, where $\alpha_i \in \mathbb{Z}$ is:*

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

More information about the p-adic numbers can be found in the work or in the introductory text by Madore (2000).

2.2 Ultrametric Finite Automata

In this section we examine ultrametric finite automata and prove various properties about them. We use the following novel definition which is made by modifying the original definition of ultrametric finite automata. Later in the section it is analyzed what advantages does the current definition have compared to the original definition.

Definition 6. *p-ultrametric finite automaton (U_pFA) is a sextuple $\langle Q, \Sigma, q_0, \delta, Q_A, Q_R \rangle$ where*

Q is a finite set – the set of states,

Σ *is a finite set – input alphabet,*
$q_0 : Q \to \mathbb{Q}_p$ *is the initial amplitude distribution,*
$\delta : \Sigma \times Q \times Q \to \mathbb{Q}_p$ *is the transition function,*
$Q_A, Q_R \subseteq Q$ *are the sets of accepting and rejecting states, respectively.*

The following expressions show how automaton works. In general U_p*FA* works similarly as a probabilistic automaton with the difference that instead of probabilities – real numbers between 0 and 1 – it uses amplitudes – p-adic numbers.

Definition 7.

$$s_\varepsilon = q_0$$

$$s_{w_1 \dots w_i}(q) = \sum_{q' \in Q} s_{w_1 \dots w_{i-1}}(q') \cdot \delta\left(w_i, q', q\right)$$

$$\sum_{q \in Q_A} \|s_w(q)\|_p > \sum_{q \in Q_R} \|s_w(q)\|_p \Leftrightarrow w \text{ is accepted}$$

An interesting class of U_p*FA*s are regulated U_p*FA*s because they recognize the same class of languages as deterministic finite automata – regular languages.

Definition 8. *If for* U_p*FA* $M = \langle Q, \Sigma, s_0, \delta, Q_A, Q_R \rangle$ *all transition amplitudes in* δ *are* p*-adic integers and there exist constants* $d_1, d_2 \in \mathbb{Z}$ *such that on any word* $w \in \Sigma^*$ *in any state* $q \in Q$ *either the amplitude* $s_w(q)$ *in state* q *after reading word* w *is equal to* 0 *or* $p^{-d_2} \leq \|s_w(q)\|_p \leq p^{-d_1}$ *then we call the automaton regulated (or more specifically –* (d_1, d_2)*-regulated).*

We show an upper bound on the increase in the number of states when constructing a *DFA* from a regulated U_p*FA*.

Theorem 4. *If a* k*-state* (d_1, d_2)*-regulated* U_p*FA* $M = \langle Q, \Sigma, s_0, \delta, Q_A, Q_R \rangle$ *recognizes a language* L*, then there exists a DFA with* $2^{k(d_2 - d_1 + 1)\log_2 p}$ *states recognizing* L*.*

We give an example of a language that shows a gap between the state complexity of deterministic and ultrametric automata.

Theorem 5. *For every* k, m *there exists a language* $L_{k,m}$ *such that:*

Every DFA recognizing $L_{k,m}$ *needs at least* k^m *states.*
For every prime p *there is a regulated* U_p*FA recognizing* $L_{k,m}$ *with* $(k+1) \cdot m - 1$ *states.*
For every prime $p > m$ *there is a* U_p*FA recognizing* $L_{p,m}$ *with* $m + 1$ *states.*

We show that $k + 1$ state suffices for a regular U_p*FA* to recognize the complement of a language that can be recognized by a regulated k-state U_p*FA*.

Theorem 6. *If a* k*-state* U_p*FA* $M = \langle Q, \Sigma, s_0, \delta, Q_A, Q_R \rangle$ *recognizes a language* L *with the property that on no word* w *the equality* $\sum_{q \in Q_A} \|s_w(q)\|_p = \sum_{q \in Q_R} \|s_w(q)\|_p$ *holds, then there exists a* k*-state* U_p*FA* M' *recognizing the complement language* $\bar{L}$*.*

Theorem 7. *If a (d_1, d_2)-regulated k-state U_pFA recognizes a language L, then there is a $(d_1, d_2 + 1)$-regulated $k + 1$-state U_pFA that recognizes L with the property that there is no word w for which the equality $\sum_{q \in Q_A} \|s_w(q)\|_p = \sum_{q \in Q_R} \|s_w(q)\|_p$ holds.*

Corollary 3. *If a (d_1, d_2)-regulated k-state U_pFA recognizes a language L, then there exists a (d_1, d_2+1)-regulated $(k+1)$-state U_pFA recognizing the complement language $\bar{L}$.*

We also compare how the ultrametric automata were defined originally (here called p-ultrametric threshold automata (U_pFTA)). U_pFTA differ from U_pFA in two aspects. First, in the U_pFTA definition it is requested that at the end of the input word is a special endmarker \$. Second, a U_pFTA has only accepting states and after reading the word, the amplitude in the accepting states is compared to a given threshold λ (instead of the amplitude in rejecting states, as in U_pFA).

It is justified why the new definition is superior. We show that there is no need to use the endmarker.

Theorem 8. *For every U_pFTA $M = (Q, \Sigma, q_0, \delta, F, \Lambda)$ there exists a U_pFTA $M' = (Q', \Sigma, q_0', \delta', F', \Lambda)$ with $|Q| + |F|$ states such that for every word w: $\sum_{q \in F} \|s_{w\$}(q)\|_p = \sum_{q \in F'} \|s'_w(q)\|_p$, where s and s' are the amplitude distributions of U_pFTAs M and M', respectively.*

We show that for any regulated U_pFTA there exists a regulated U_pFA recognizing the same language with the number of states increased by an amount which represents the complexity of representing the threshold as a sum of norms of p-adic numbers.

Theorem 9. *If a language L is recognized (without endmarker) by a U_pFTA $M = (Q, \Sigma, q_0, \delta, F, (\lambda, \diamond))$ such that there exists $\lambda' = \sum_{i=a}^{b} l_i \cdot p^i$ such that $\forall w \in \Sigma^* \sum_{q \in F} \|s_w(q)\|_p \diamond \lambda \Leftrightarrow \sum_{q \in F} \|s_w(q)\|_p \tilde{\diamond} \lambda'$ (where $\tilde{\leq}$ is $<$ and $\tilde{\geq}$ is $>$) then there exists a U_pFA M' with $|Q| + \sum_{i=a}^{b} l_i$ states which recognizes L.*

Theorem 10. *If a language L is recognized by a regulated U_pFTA $M = (Q, \Sigma, q_0, \delta, F, (\lambda, \diamond))$ then there exists $\lambda' = \sum_{i=a}^{b} l_i \cdot p^i$ such that $\forall w \in \Sigma^* \sum_{q \in F} \|s_w(q)\|_p \diamond \lambda \Leftrightarrow \sum_{q \in F} \|s_w(q)\|_p \tilde{\diamond} \lambda'$.*

We also examine multi-head ultrametric automata. For one-way automata the following result is proven:

Theorem 11. *For every $k \geq 1 \in \mathbb{N}$, there exists a language L_k such that:*

for every prime p there exists a $1U_pFA(1)$ that recognizes L_k,
L_k cannot be recognized by any $1NFA(k)$.

For two-way automata it is shown that there exists a hierarchy, i.e., by adding an extra head to a k-head automaton the class of recognizable languages increases.

Theorem 12.

$$\mathcal{L}(2U_pFA(k)) \subsetneqq \mathcal{L}(2U_pFA(k+1)).$$

3 Counting with Automata

In this section, we investigate the descriptional complexity advantages for probabilistic and ultrametric automata compared with deterministic, nondeterministic and alternating automata. We limit our focus to unary languages containing exactly one word. We say that an automaton counts to n if it recognizes the language C_n.

Definition 9. $C_n = \{1^n\}$.

We get new results about probabilistic and ultrametric automata for the counting problem and show that they can be very succinct, requiring only a constant number of states in many models.

This section is based on Balodis (2014).

3.1 Probabilistic Automata

We show an optimal probabilistic one-way automaton which recognizes C_n.

Theorem 13. *For each n, there exists a 1PFA that recognizes C_n with 3 states with an isolated cutpoint.*

Theorem 14. *If $n > 1$ then any 1PFA that recognizes C_n has at least 3 states.*

Notice that although each individual automaton $\mathcal{A}_n$ from Theorem 13 has an isolated cutpoint, the isolation radius tends towards 0 as n increases. We show that in the two-way case there is no such deficiency and a constant number of states suffices for all n (where the constant even does not depend on the required probability to give the correct answer).

Theorem 15. *There exists a constant c such that for every $\varepsilon > 0$ and for each n there exists a 2PFA that recognizes C_n with c states with probability $1 - \varepsilon$.*

3.2 Ultrametric Automata

We show that an ultrametric regulated automaton can count to n with 2 states and two states are necessary even if the automaton is not regulated.

Theorem 16. *For each n and each prime p there exists a regulated U_pFA that recognizes C_n with 2 states.*

Theorem 17. *If $n > 0$ then any U_pFA that recognizes C_n has at least 2 states.*

4 Frequency Finite Automata

The notion of frequency computation was introduced by Rose (1960). A frequency algorithm (m,n)-computes a function f, if given any n pairwise distinct $x_1,\ldots,x_n$ it produces n outputs $y_1,\ldots,y_n$ of which at least m are correct (i.e., for m different values of i the equation $y_i = f(x_i)$ holds). If the function is of type $f : \Sigma^* \to \{0,1\}$ then we speak of a special case of (m,n)-computable languages.

While frequency finite automata have already been considered earlier, they have been considered only in the setting of one-way automata which simultaneously reads a symbol from each of the input words. Here we introduce two-way finite automata which have not been considered earlier in the literature and show their relationship with deterministic two-way finite automata with linearly-bounded counters and the complexity class *LOGSPACE*.

4.1 Two-Way Frequency Automata

We know such classical result about frequency computing with Turing machines.

Theorem 18 (Trakhtenbrot 1963). *If $\frac{m}{n} > \frac{1}{2}$, then every language that is (m,n)-recognizable with a Turing machine is recursive.*

If $\frac{m}{n} \leq \frac{1}{2}$, then there is a continuum of languages that are (m,n)-recognizable with a Turing machine.

Similar theorem is known about one-way frequency finite automata.

Theorem 19 (Kinber 1976; Austinat et al. 2005). *For one-way finite automata:*

If $\frac{m}{n} > \frac{1}{2}$, then every (m,n)-recognizable language is regular.

If $\frac{m}{n} \leq \frac{1}{2}$, then there is a continuum of (m,n)-recognizable languages.

Analogously one could expect that in the two-way automata case with a frequency (m,n) with $\frac{m}{n} > \frac{1}{2}$ only the same languages could be recognized that can be recognized with a deterministic two-way automaton (i.e., only regular languages), however we will see that it is not the case and already allowing only 1 error (i.e., with a frequency $(n-1,n)$) a larger class of languages can be recognized.

We introduce the following definition which is constructed by applying the notion of frequency computing to two-way finite automata in the most natural way.

Definition 10. *For natural numbers m,n $(1 \leq m \leq n)$ a two-way (m,n)-frequency finite automaton ((m,n)-2FFA) is a tuple $\mathcal{A} = (Q,\Sigma,\delta,q_0,F)$, where*

Q is the finite set of states,
Σ is the input alphabet,
$\delta : Q \times (\Sigma \cup \{\vdash,\dashv\})^n \to Q \times \{L,N,R\}^n$ is the transition function, where $\vdash,\dashv \notin \Sigma$ are the left and right endmarkers, respectively, and L, N, R denote the head movement (to the left, no movement, and to the right, respectively),
$q_0 \in Q$ is the starting state, and
$F : Q \to \{0,1\}^n$ is the acceptance function.

Definition 11. *We say that a language $L \subseteq \Sigma^*$ is recognized by an (m,n)-2FFA $\mathcal{A}$ if for every n distinct input words $x_1, \ldots, x_n \in \Sigma^*$ the automaton $\mathcal{A}$ when started on $x_1, \ldots, x_n$ gives an output $(y_1, \ldots, y_n) \in \{0,1\}^n$ such that at least m of the following hold:*

$$\begin{aligned} y_1 = 1 &\Leftrightarrow x_1 \in L \\ y_2 = 1 &\Leftrightarrow x_2 \in L \\ &\vdots \\ y_n = 1 &\Leftrightarrow x_n \in L \end{aligned}$$

4.2 Results

It is not surprising that with a frequency (n,n) only regular languages can be recognized.

Theorem 20. $\mathcal{L}\left((n,n)\text{-2FFA}\right) = \mathcal{L}\left((1,1)\text{-2FFA}\right) = REG$

However, with a frequency $(n-1,n)$ the class of recognizable languages is larger.

We denote an automaton with k linearly bounded counters by 2*BCA*(k) (the value of the counter never exceeds the length of the input). We show that every language recognizable with a 2*BCA*(k) is also recognizable with a $(n-k,n)$-2FFA.

Theorem 21. *For any $n > k$:*

$$\mathcal{L}\left((n-k,n)\text{-2FFA}\right) \supseteq \mathcal{L}\left(2BCA(k)\right)$$

This result shows that with a frequency $(n-1,n)$ a lot of non-trivial languages can be recognized.

Corollary 4. *For any $n > 1$ the languages*

$\left\{1^{2^m} \mid m \geq 0\right\}$,
$\left\{1^{2^{2^m}} \mid m \geq 0\right\}$,
$\left\{1^{42^{m^2}} \mid m \geq 0\right\}$,
$\left\{1^{11^p} \mid p \text{ is a prime}\right\}$,
$\left\{0^m 1^{m^2} \mid m \geq 0\right\}$,
$\left\{0^m 1^{2^m} \mid m \geq 0\right\}$

can be recognized by an $(n-1,n)$-2FFA.

An open question is whether an $(n-k,n)$-2FFA can recognize any language that cannot be recognized by k linearly bounded counters. The following theorem provides a sufficient condition for it.

Theorem 22.

$$(\mathcal{L}\,(2BCA(k+1)) \setminus \mathcal{L}\,(2BCA(k))) \cap \mathcal{L}\,((1,k+1)\text{-}2FFA) \neq \emptyset \Rightarrow$$
$$\forall n > k \;\; \mathcal{L}\,((n-k,n)\text{-}2FFA) \supsetneq \mathcal{L}\,(2BCA(k))$$

We also show that it is not correct to interpret the frequency (m,n) as a fraction $\frac{m}{n}$. The following result shows that any language recognizable with a logarithmic-space Turing machine (i.e., any language from *LOGSPACE*) can be (m,n)-recognized with a two-way frequency automaton with the fraction $\frac{m}{n}$ arbitrary close to 1.

Theorem 23.

$$\forall L \in LOGSPACE \;\; \exists k \, \forall n > k \; L \in \mathcal{L}\,((n-k,n)\text{-}2FFA)$$

5 Ultrametric Query Algorithms

Query algorithms is a well-known model in theoretical computer science. We have to calculate the result of a function f on the input $x = x_1x_2 \ldots x_n$, i.e., $f(x)$. Usually Boolean functions $f : \{0,1\}^n \to \{0,1\}$ are considered. The function f is known to the algorithm. The value of the input x is unknown and the only way to access it is via queries to a "black box". A query is the call of the function $O_x : \{1,\ldots,n\} \to \{0,1\}$ where $O_x(i) = x_i$, i.e., by querying i the algorithm learns the value of x_i. The task of the algorithm is to compute $f(x)$ by making as few queries as possible.

Randomized query algorithm can probabilistically choose which variable to query. Quantum query algorithm can make the queries in a quantum superposition.

The query algorithm model is extensively studied for quantum query algorithms. It is a lower bound for time complexity.

In this section we introduce ultrametric query algorithms. This model is constructed similarly as deterministic, probabilistic and quantum query algorithms.

Here we examine several properties of the ultrametric query algorithms. Some of the main results are relating ultrametric query complexity with the polynomial degree of the Boolean function.

This section is partly based on Jērinš et al. (2014).

5.1 Ultrametric Query Complexity

Ultrametric query algorithms are based on p-adic numbers. A p-ultrametric algorithm is described by a directed acyclic graph with exactly one vertex (root) which has no incoming edges. The nodes with no outgoing edges are leafs and they are the final (accepting) states of the algorithm.

Definition 12. *We say that a p-ultrametric query algorithm is one-endpoint if it has exactly one accepting state.*

Definition 13. *We say that a p-ultrametric query algorithm is exact if for every input the sum of norms of the final amplitudes is either 0 or 1.*

The complexity of a p-ultrametric query algorithm is defined as the maximum number of queries in a branch from the root to an accepting state.

The p-ultrametric query complexity (denoted by $U_p(f)$) of a function f is the complexity of an optimal p-ultrametric query algorithm which computes f. The exact p-ultrametric query complexity (denoted by $U_{p,E}(f)$) of a function f is the complexity of an optimal p-ultrametric query algorithm which exactly computes f. We denote the corresponding complexities for one-endpoint algorithms by $U^1_p(f)$ and $U^1_{p,E}(f)$. Of course, for every p and f it holds that $U_p(f) \leq U_{p,E}(f) \leq U^1_{p,E}(f)$ and $U_p(f) \leq U^1_p(f)$.

Figure 1 depicts a 2-ultrametric query algorithm for XOR_n function.

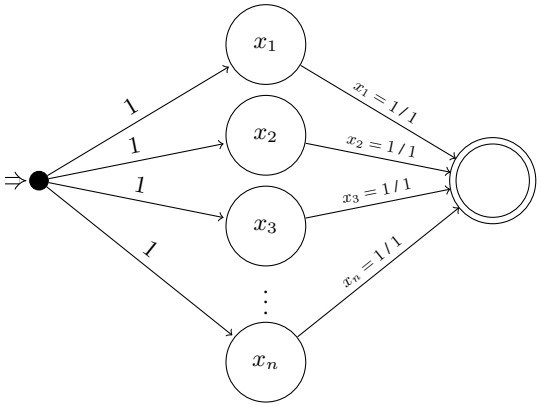

Fig. 1. 2-ultrametric query algorithm for XOR_n

Let $deg(f)$ be the degree of the unique multilinear polynomial representing the Boolean function f. We show that one-endpoint p-ultrametric exact query complexity is at most the polynomial degree of the function therefore is at most two times the quantum exact query complexity.

Theorem 24. $U^1_{p,E}(f) \leq deg(f) \leq 2Q_E(f)$

The 2-ultrametric algorithm for XOR_n hints for a more general way of constructing 2-ultrametric algorithms with a small complexity.

Definition 14. *Denote by $deg_2(f)$ the binary polynomial degree of function f, i.e., the minimal degree of a polynomial $p(x)$ such that $p(x) \equiv f(x) \pmod 2$.*

Theorem 25. $U^1_2(f) \leq deg_2(f)$

An interesting class of functions are those for which there exist p-ultrametric query algorithms with a small complexity for a specific p.

Definition 15.

$$NDIV_{n,p}(x) = \begin{cases} 0, & \textit{if the number } \overline{x_n x_{n-1} \dots x_1} \textit{ is divisible by } p \\ 1, & \textit{otherwise} \end{cases}$$

Theorem 26. *For any n and any prime p there exists a one-endpoint p-ultrametric query algorithm with complexity* 1 *that computes $NDIV_{n,p}$.*

6 Structured Frequency Algorithms

In the original definition of frequency computation the algorithm receiving n different inputs is required to produce n outputs of which at least m are correct. In this section we introduce a new, modified definition of frequency computation – structured frequency computation. We fix several subsets of inputs (all together called structure). The algorithm is required that in at least one of the fixed subsets all of the outputs are correct. Structured frequency computation had not been considered earlier in the literature, it is defined by Balodis, Iraids, and Freivalds (2015).

6.1 Frequency Computation

We will examine the computing of sets. By $\chi_A : \mathbb{N} \to \{0, 1\}$ we denote the characteristic function of a set A:

$$\chi_A(x) = \begin{cases} 1, & \text{if } x \in A \\ 0, & \text{if } x \notin A \end{cases}$$

The original definition of frequency computation is the following:

Definition 16 (Rose 1960). *A set A is (m, n)-computable iff there is a total recursive function f which assigns to all distinct inputs $x_1, x_2, \ldots, x_n$ a binary vector $(y_1, y_2, \ldots, y_n)$ such that at least m of the equations $\chi_A(x_1) = y_1, \chi_A(x_2) = y_2, \ldots, \chi_A(x_n) = y_n$ hold (see Figure 2).*

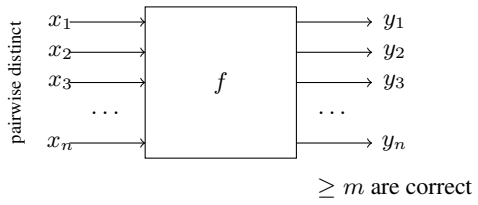

Fig. 2. Frequency computing

We construct the notion of frequency computing by changing the definition of frequency computing by requiring that the correct answers cover at least one of the subsets of the structure. The first natural question is – what structures allow the computing of recursive sets and which allow computing something more? We show that even when only $\sqrt{n}$ of the n answers are required to be correct, only recursive sets can be recognized. A similar property is known about the classical frequency (m, n)-computation

with frequency $\frac{m}{n} > \frac{1}{2}$. We also investigate graph frequency computation where the size of the structures is limited to 2 so they can be represented as the edges of a graph.

Definition 17. *By a structure of a finite set K we call a set of K's subsets $S \subseteq 2^K$. Assume $K = \{1, 2, \ldots, n\}$.*

Definition 18. *A set A is (S, K)-computable (or computable with a structure S) iff there is a total recursive function f which assigns to all distinct inputs $x_1, x_2, \ldots, x_n$ a binary vector $(y_1, y_2, \ldots, y_n)$ such that $\exists B \in S\ \forall b \in B\ \chi_A(x_b) = y_b$.*

Definition 19. *By the size of a structure $S \subseteq 2^K$ we denote the size of the smallest subset - $\min_{A \in S} |A|$. We call the structure* size consistent *iff $\neg\exists K' \subseteq K\ \min_{A' \in S} \frac{|A' \cap K'|}{|K'|} > \min_{A \in S} \frac{|A|}{|K|}$*

6.2 Projective Plane Frequency Computation

In this subsection we prove that there are structures of size $O(\sqrt{n})$ which only allow computation of recursive sets. Therefore the algorithm is required to give the correct answer on a small fraction of inputs – $O\left(\frac{\sqrt{n}}{n}\right) = O\left(\frac{1}{\sqrt{n}}\right)$ – (which is much less than $\frac{1}{2}$ which corresponds to the original definition), however only recursive sets can be computed. We use projective planes to construct these structures.

Definition 20. *We call a structure $S \subseteq 2^K$ overlapping iff $\forall A, B \in S\ A \cap B \neq \emptyset$.*

Theorem 27. *If A is computable with an overlapping structure then A is recursive.*

Theorem 28. *For any set K of size $n = q^2 + q + 1$ where q is a prime power there exists a size consistent overlapping structure of size $q + 1$.*

We also prove that for overlapping structures the fraction obtained by the finite planes is close to the best possible.

Theorem 29. *Every size consistent overlapping structure $S \subseteq 2^K$ has size at least $\sqrt{n}$ where $n = |K|$.*

6.3 Graph Frequency Computation

The smallest interesting size of structures is 2. In this subsection we focus on such structures. A convenient and well-known way to represent such structures is using graphs.

Definition 21. *We call a structure $S \subseteq 2^K$ a* graph structure *iff $\forall A \in S\ |A| = 2$.*

A natural question arises – for which graphs G are the G-computable sets recursive? We show for some graphs G that only recursive sets can be G-computed or show that there exists a continuum of G-computable sets (therefore among them are also non-recursive sets).

Proposition 1. *If the graph G is either a triangle (C_3) or a star graph (S_k) then every G-computable set is recursive (see Figure 3).*

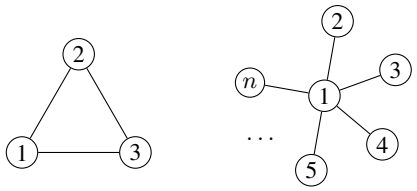

Fig. 3. Graphs C_3 and S_n

Theorem 30. *If a graph G contains as a subgraph a cycle of length 4 (C_4) or two vertex-disjoint paths of length 3 ($2P_3$) then there is a continuum of G-computable sets, namely, every $(1,2)$-computable set is also G-computable (see Figure 4).*

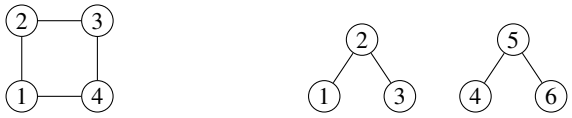

Fig. 4. Graphs C_4 and $2P_3$

The next two theorems show that graphs consisting of 2 pairs of connected vertices and 3 pairs of connected vertices are substantially different.

Theorem 31. *If a graph G contains as a subgraph three vertex-disjoint paths of length 2 ($3P_2$) then there is a continuum of G-computable sets. (see Figure 5).*

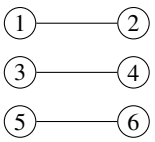

Fig. 5. Graph $3P_2$

Theorem 32. *If a graph G is two vertex-disjoint paths of length 2 ($2P_2$) then every G-computable set is recursive (see Figure 6).*

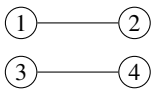

Fig. 6. Graph $2P_2$

7 Conclusion and Discussion

We have considered several types of unconventional finite automata, ranging from two-way probabilistic and alternating finite automata to ultrametric and frequency finite automata. We have also examined two types of unconventional computation – ultrametric query algorithms and structured frequency computation. In all considered areas we have compared how the unconventional models relate to classical models. Although we have proven many new results, there are still a lot of connections to be found.

In the two-way finite automata size complexity theory we have shown a previously unknown relation between alternating and probabilistic automata. More specifically that there is a family of languages that is recognizable by a family of polynomial-size alternating automata, but for every family of fast probabilistic bounded-error two-way automata the sizes of the automata grow superpolynomially. Although in the literature there are several results showing advantages of probabilistic models of automata over non-probabilistic models of automata, this seems to be one of very rare examples where a class of non-probabilistic automata are shown to be more powerful than a class of probabilistic automata. It seems that the result could be strengthened to include also probabilistic automata that can work superpolynomial time or have non-isolated cut-point, but it seems hard to prove it with the current techniques, therefore some new approach might be needed.

We have introduced the ultrametric finite automata. We think that for these the most perspective model is the regulated ultrametric automata as they can recognize exactly the regular languages and a bound on the complexity of simulating them with deterministic automata is given. It would be interesting to find some connections of how they relate to alternating and nondeterministic automata.

We have considered all of the above-mentioned types of automata for the counting problem, i.e., recognizing the one-word unary language $C_n = \{1^n\}$. We have shown optimal constant-size probabilistic and ultrametric regulated automata. However it is still open, if we require the two-way probabilistic automaton to be fast and have the error probability bounded by a constant, can it do any better than one-way automaton, i.e., have less than $O(\log^2 n/ \log\log n)$ states.

We have introduced the two-way frequency finite automata. We have shown that any language recognizable with two-way automaton with k linearly bounded counters is $(n-k, n)$-recognizable by a two-way frequency finite automaton for any $n > k$. This relation shows that many nontrivial languages are $(n-1, n)$-recognizable. However, it is an open question whether there are any other languages recognizable by a frequency automaton. In the work we have shown a sufficient condition for it.

Based on the p-adic numbers we have defined ultrametric query algorithms and ultrametric query complexity similar to probabilistic and quantum query complexity.

However, the unrestricted ultrametric query model seems to be too powerful because such functions as OR_n and XOR_n can be computed with just 1 query and the complexity never exceeds the polynomial degree of the function. Therefore one should try to find a natural restriction for the model that for some functions gives more interesting query complexity bounds such as $O(\log n)$ and $O(\sqrt{n})$. One could also try to devise some lower bound technique and explore what functions are hard for ultrametric query algorithms.

We have introduced the notion of structured frequency algorithms by modifying the definition of frequency computation. Based on finite planes we have shown a structure of size $O(\sqrt{n})$ that allows only recognition of recursive sets and shown that using overlapping structures this size cannot be decreased. It is an open question whether there are some other (non-overlapping) structures of smaller size that allow only recognition of recursive sets.

Acknowledgements

This work has been supported by the European Social Fund within the project "Support for Doctoral Studies at University of Latvia" and by the project 271/2012 from the Latvian Council of Science.

References

Arora, Sanjeev and Boaz Barak (2009). *Computational complexity: a modern approach*. Cambridge University Press.

Austinat, Holger et al. (2005). "Regular frequency computations". In: *Theoretical Computer Science* 330.1. Insightful Theory, pp. 15 –21. ISSN: 0304-3975. DOI: http://dx.doi.org/10.1016/j.tcs.2004.09.007. URL: http://www.sciencedirect.com/science/article/pii/S0304397504006036.

Balodis, Kaspars (2013). "One alternation can be more powerful than randomization in small and fast two-way finite automata". In: *Fundamentals of Computation Theory*. Springer, pp. 40–47.

Balodis, Kaspars (2014). "Counting with probabilistic and ultrametric finite automata". In: *Computing with New Resources*. Springer, pp. 3–16.

Balodis, Kaspars, Jānis Iraids, and Rūsiņš Freivalds (2015). "Structured Frequency Algorithms". In: *Theory and Applications of Models of Computation*. Springer, pp. 50–61.

Balodis, Kaspars et al. (2013). "On the State Complexity of Ultrametric Finite Automata". In: *SOFSEM 2013: Theory and Practice of Computer Science*. Vol. 2, pp. 1–9.

Berman, Piotr and Andrzej Lingas (1977). *On complexity of regular languages in terms of finite automata*. Institute of Computer Science, Polish Academy of Sciences.

Drucker, Andrew and Ronald de Wolf (2009). "Quantum proofs for classical theorems". In: *arXiv preprint arXiv:0910.3376*.

Jērinš, Kārlis et al. (2014). "Ultrametric query algorithms". In: *SOFSEM 2014: Theory and Practice of Computer Science*. Vol. 2, pp. 21–29.

Kapoutsis, Christos A. (2009). "Size complexity of two-way finite automata". In: *International Conference on Developments in Language Theory*. Springer, pp. 47–66.

Kapoutsis, Christos A. (2012). "Minicomplexity". In: *International Workshop on Descriptional Complexity of Formal Systems*. Springer, pp. 20–42.

Kinber, Efim B. (1976). “Frequency computations in finite automata”. In: *Cybernetics and Systems Analysis* 12.2, pp. 179–187.

Královic, Richard (2010). “Complexity classes of finite automata”. PhD thesis. ETH Zürich.

Madore, David A. (2000). *A first introduction to p-adic numbers.* Online. URL: `http://www.madore.org/~david/math/padics.pdf`.

Rose, Gene F. (1960). “An extended notion of computability”. In: *International Congress for Logic, Methodology and Philosophy of Science, Stanford, California*.

Sakoda, William J. and Michael Sipser (1978). “Nondeterminism and the size of two way finite automata”. In: *Proceedings of the tenth annual ACM symposium on Theory of computing*. STOC ’78. San Diego, California, USA: ACM, pp. 275–286. DOI: `10.1145/800133.804357`. URL: `http://doi.acm.org/10.1145/800133.804357`.

Trakhtenbrot, B. A. (1963). “On the frequency computation of functions”. In: *Algebra i Logika* 2.1. In Russian, pp. 25–32.

Received May 31, 2016 , accepted June 4, 2016