

UNCOUNTABLE CLASSICAL AND QUANTUM COMPLEXITY CLASSES[☆]

MAKSIMS DIMITRIJEVS* AND ABUZER YAKARYILMAZ

Abstract. It is known that poly-time constant-space quantum Turing machines (QTMs) and logarithmic-space probabilistic Turing machines (PTMs) recognize uncountably many languages with bounded error (A.C. Cem Say and A. Yakaryılmaz, Magic coins are useful for small-space quantum machines. *Quant. Inf. Comput.* **17** (2017) 1027–1043). In this paper, we investigate more restricted cases for both models to recognize uncountably many languages with bounded error. We show that double logarithmic space is enough for PTMs on unary languages in sweeping reading mode or logarithmic space for one-way head. On unary languages, for quantum models, we obtain middle logarithmic space for counter machines. For binary languages, arbitrary small non-constant space is enough for PTMs even using only counter as memory. For counter machines, when restricted to polynomial time, we can obtain the same result for linear space. For constant-space QTMs, we obtain the result for a restricted sweeping head, known as restarting realtime.

Mathematics Subject Classification. 68Q05, 68Q15, 68Q75.

Accepted November 21, 2018.

1. INTRODUCTION

It is a well-known fact that two-wayness and alternation do not help to recognize a nonregular language for constant-space Turing machines (finite state automata) [24]. When formally defined first time [21], one-way probabilistic finite automata (PFAs) were also shown to recognize all and only regular languages with bounded error. On the other hand, in his seminal paper, Freivalds [15] showed that two-way PFAs can recognize some nonregular languages with bounded error. But, it was shown that two-way PFAs require exponential expected time to recognize nonregular languages [14]. The quantum counterpart of two-way PFAs, two-way quantum finite automata (two-way QFAs), were defined in [18], and it was shown that they can recognize nonregular languages with bounded error in linear time even with one-way head move [25]. What makes these models very powerful is the ability of being in a superposition of tape positions (called quantum head), which can be cleverly used to implement a counter in a very special way (see [32]). Later, two-way QFAs with classical head (2QCFAs) were defined [3], and it was shown that they can also recognize certain nonregular languages in nonlinear but still polynomial expected time.

[☆]A preliminary version appeared as “Maksims Dimitrijevs, Abuzer Yakaryılmaz: Uncountable classical and quantum complexity classes. NCMA 2016: 131-146” [9]. The arXiv number is [1608.00417](https://arxiv.org/abs/1608.00417).

Keywords and phrases: Probabilistic and quantum computation, small-space bounds, unary languages, uncountable classes, counter machines.

Faculty of Computing, University of Latvia, Rīga, Latvia.

* Corresponding author: md09032@lu.lv

It is obvious that there are countably many regular languages since the description of any one-way deterministic finite automaton, which defines a single regular language, is finite. Similarly, computable (regular or not) languages, the ones recognized by Turing machines, form a countable set. On the other hand, all languages form an uncountable set, and so there are uncountably many nonregular languages.

A probabilistic or quantum model can be defined with uncomputable transition values, and so there are uncountably many probabilistic or quantum machines. At this point, it is natural to ask whether they can define an uncountable class. With unbounded error (recognition with cutpoint), unary 2-state QFAs and unary 3-state PFAs¹ define the classes formed by uncountably many languages [22, 23]. (Unary PFAs with 2-states define only a finite number of regular languages [20, 23].) So, the nontrivial question is to determine the minimal bounded-error probabilistic and quantum classes containing uncountably many languages.

Poly-time constant-space quantum Turing machines (QTMs) and logarithmic-space probabilistic Turing machines (PTMs) are known to recognize uncountably many languages with bounded error [1, 7]. In this paper, we investigate more restricted cases for QTMs and PTMs, *i.e.*, using less space, restricted memory types, and restricted input head moves.

We show that double logarithmic space is enough for PTMs on unary languages in sweeping reading mode and logarithmic space for one-way head. On unary languages, for quantum models, we obtain middle logarithmic space for counter machines. For binary languages, arbitrary small non-constant space is enough for PTMs even using only counter as memory. For counter machines, when restricted to polynomial time, we can obtain the same result for linear space. If we focus on middle space without time restrictions, then we can improve the space bound to logarithmic for sweeping counter machines. For constant-space QTMs, we obtain the result for a restricted sweeping head, known as restarting realtime.

In the next section, we give the required background with a short introduction to quantum operators, and then we present our results in Section 3 under four subsections. We first present the results for PTMs that are pedagogically easy to follow (Sect. 3.1). Then, we restrict the model to use counter as a memory (Sect. 3.2). After this, we also check some restrictions on the head movement (Sect. 3.3). Lastly, we present our quantum results (Sect. 3.4). We close the paper by listing our results, some recent related results, and related open problems in Section 4.

We refer the reader to [9] for the conference version of the paper. Here, we slightly improve the content, include the omitted proofs for quantum results, and add a new result on middle logarithmic-space sweeping probabilistic counter automata (Thm. 3.12).

2. BACKGROUND

We assume that the reader is familiar with the basics of complexity theory and automata theory. We refer the reader to [19] for a complete reference on quantum computation, to [6] for a pedagogical introduction to quantum finite automata (QFAs), and to [4] for a comprehensive chapter on QFAs.

Throughout the paper, $\#$ denotes the blank symbol, ε denotes the empty string, Σ not containing $\$$ (the left end-marker) and $\$$ (the right end-marker) denotes the input alphabet, $\tilde{\Sigma}$ is the set $\Sigma \cup \{\$, \#\}$, Γ not containing $\#$ denotes the work tape alphabet, $\tilde{\Gamma}$ is the set $\Gamma \cup \{\#\}$, and Σ^* is the set of all strings obtained from the symbols in Σ including the empty string. We order the elements of Σ^* lexicographically and then represent the i th element by $\Sigma^*(i)$ where the first value $\Sigma^*(1)$ is the empty string. We fix n as the length of any given input.

Each model has a read-only one-way infinite input tape with a single head, on which the given input w is placed as $\tilde{w} = \$w\$$. All the remaining tape cells are filled with blank symbols. At the beginning of the computation, the input head is placed on the left end-marker. Each machine is designed to guarantee that the input head never visits outside \tilde{w} . A work tape is a two-way infinite tape with a single head, where each tape cell is indexed with an integer. At the beginning of the computation, all cells of a work tape are filled with symbols $\#$, and the head is placed on the cell indexed by zero.

¹As another “probabilistic” but unconventional model, ultrametric automata can also define uncountably many languages with 2 states [8].

A deterministic Turing machine (DTM) D having an input tape and a work tape is a 7-tuple

$$D = (S, \Sigma, \Gamma, \delta, s_1, s_a, s_r),$$

where S is the set of finite internal states, $s_1 \in S$ is the initial state, $s_a \in S$ and $s_r \in S$ ($s_a \neq s_r$) are the accepting and rejecting states, respectively, and δ is the transition function

$$\delta: S \times \tilde{\Sigma} \times \tilde{\Gamma} \rightarrow S \times \tilde{\Gamma} \times \{\leftarrow, \downarrow, \rightarrow\} \times \{\leftarrow, \downarrow, \rightarrow\}$$

that governs the behavior of D as follows: when D is in state $s \in S$, reads symbol $\sigma \in \tilde{\Sigma}$ on the input tape, and reads symbol $\gamma \in \tilde{\Gamma}$ on the work tape, it follows the transition

$$\delta(s, \sigma, \gamma) = (s', \gamma', d_i, d_w), \quad (2.1)$$

and then the state becomes $s' \in S$, γ' is written on the cell under the work head, and then the positions of input and work heads are updated with respect to $d_i \in \{\leftarrow, \downarrow, \rightarrow\}$ and $d_w \in \{\leftarrow, \downarrow, \rightarrow\}$, respectively, where “ \leftarrow ” (“ \downarrow ” and “ \rightarrow ”) means the head moves one cell to the left (the head does not move and the head moves one cell to the right). The computation starts in state s_1 , and the computation is terminated and the given input is accepted (rejected) if D enters s_a (s_r). The set of strings accepted by D forms a language, say $L \subseteq \Sigma^*$, and it is said that L is recognized by D .

The space used by D on a given input is the number of all cells visited on the work tape during the computation.

If the input head is not allowed to move to the left, then it is called “one-way”, and then the model is denoted as 1DTM. If we remove the work tape (and all related components in the formal definition and in the transition function) of a DTM/1DTM, we obtain a two-way/one-way deterministic finite automaton (2DFA/1DFA).

A counter is a special type of memory containing only the integers. Its value is set to zero at the beginning. During the computation, its status (whether its value is zero or not) can be read like reading blank symbol or not on the work tape, and then its value is incremented or decremented by 1 or not changed like the position update of work head. A deterministic counter automaton (2DCA) is a 2DFA with a counter. The space used (on the counter) by a 2DCA is the maximum absolute value of the counter during the computation. Remark that the value of the counter can be stored on a binary work tape with logarithmic amount of the space. We remark that a counter can also be seen as a unary work tape with certain specifications.

A PTM is a generalization of a DTM such that it can make random choices according to some probability distributions. Thus, a PTM can do more than one transition in each step, but each choice can be realized only with some probabilities. The number of choices and their realization probabilities are determined by the current state and the symbols read on the tapes, and the summation of the probabilities must be 1 to have a well-formed probabilistic systems. Thus, a PTM can follow different paths during the computation and so the input is accepted with some probabilities. Remark that all probabilistic models in this paper halt either absolutely or with probability 1. In the latter case, the running time is expected. Since the space usage of a PTM can be different on the different paths, we take the maximum value.

The language L is said to be recognized by a PTM with error bound ϵ ($0 \leq \epsilon < \frac{1}{2}$) if every member of L is accepted with probability at least $1 - \epsilon$ and every non-member of L ($w \notin L$) is accepted with probability not exceeding ϵ .

One-way PTM (1PTM) is defined similarly to 1DTM. A PTM/1PTM without a work tape is a two-way/one-way probabilistic finite automaton (2PFA/1PFA). A probabilistic counter automaton (2PCA) is a 2PFA with a counter.

A quantum system with m states ($Q = \{q_1, \dots, q_m\}$) forms an m -dimensional Hilbert space (\mathcal{H}^m), which is a complex vector space with inner product, spanned by $\{|q_1\rangle, \dots, |q_m\rangle\}$, where $|q_j\rangle$ is a column vector with zero

entries except the j th entry that is 1. A quantum state of the system is a norm-1 vector in \mathcal{H}^m :

$$|v\rangle = \alpha_1|q_1\rangle + \cdots + \alpha_m|q_m\rangle, \quad \sum_{j=1}^m |\alpha_j|^2 = 1,$$

where α_j is a complex number and represents the amplitude of the system being in $|q_j\rangle$, and the probability of system being in $|q_j\rangle$ is given by $|\alpha_j|^2$.

The quantum system evolves by unitary operators, also known as norm preserving operators, represented by unitary matrices. Let U be a unitary operator (matrix). Then its (l, j) th entry represents the transition amplitude from $|q_j\rangle$ to $|q_l\rangle$, where $1 \leq j, l \leq n$. After applying U , the new state is

$$|v'\rangle = U|v\rangle = \alpha'_1|q_1\rangle + \cdots + \alpha'_m|q_m\rangle, \quad \sum_{j=1}^m |\alpha'_j|^2 = 1.$$

In order to retrieve information from the system, measurement operators are applied. We use a simple one called projective measurement, say P . Formally P is composed by $k \geq 1$ elements $\{P_1, \dots, P_k\}$. Each P_i is a zero-one diagonal (nonzero) matrix and $P_1 + \cdots + P_k = I$. So, P is designed to decompose \mathcal{H}^m into k orthogonal subspaces and P_j projects any vector to its subspace, where $1 \leq j \leq k$. After applying P to the system when in $|v\rangle$, the system collapses to one of the subspaces and so the new quantum state lies only in this subspace. The vector

$$|\tilde{v}_j\rangle = P_j|v\rangle$$

is the projection of the quantum state to the j th space, and so the probability of observing the system in this subspace is given by $p_j = \|\tilde{v}_j\|^2$. If this happens ($p_j > 0$), the new quantum state is

$$|v_j\rangle = \frac{|\tilde{v}_j\rangle}{\sqrt{p_j}}.$$

The vector $|\tilde{v}_j\rangle$ is called unnormalized state vector, and tracing quantum systems by such vectors can make the calculations simpler.

Now we give the definition of two-way quantum finite automaton with classical head, known as two-way finite automaton with quantum and classical states (2QCFA) [3], which can use unitary operators and projective measurements on the quantum part. Formally, a 2QCFA M is an 8-tuple

$$M = (S, Q, \Sigma, \delta, s_1, q_1, s_a, s_r),$$

where, different from a classical model, Q is the set of quantum states, q_1 is the initial quantum state, and the transition function δ is composed by δ_q governing the quantum part and δ_c governing the classical part. The computation is governed classically. At the beginning of the computation, the classical part is initialized and the state of the quantum part is set to $|q_1\rangle$. In each step, the current classical state and scanned symbol determines a quantum operator, either a unitary operator or a projective measurement, that is applied to the quantum register. After getting the new quantum state, the classical part is updated. If the quantum operator is unitary, then the classical part is updated like a 2DFA. If the quantum operator is a measurement, then the outcome is processed classically, that is, the next state and head movement is determined by the current classical state, the measurement outcome, and the scanned symbol. When entering s_a (s_r), the computation halts and the input is accepted (rejected).

A (strict) realtime version of 2QCFA (rtQCFA) [33] moves its head one square to the right in each step, halts the computation after reading the right end-marker, and applies one unitary operator and then measurement operator for the quantum part in each step.

A 2QCFA with a counter (2QCCA) is a 2QCFA augmented with a classical counter, where the classical part can access a counter.

A two-way model is called sweeping if the direction of the head can be changed only on the end-markers. So, the input is read from the left to the right, then right to left, and then left to right, and so on. A very restricted version of sweeping models is restarting realtime models (see [29, 31] for the details of restarting concept): the models have an additional state s_i such that, immediately after entering s_i , the overall computation is terminated and all computations start from the initial configuration. In this paper, we focus on restarting rtQCFA's.

We denote the set of integers \mathbb{Z} and the set of positive integers \mathbb{Z}^+ . The set \mathcal{I} is the set of all subsets of \mathbb{Z}^+ :

$$\mathcal{I} = \{I \mid I \subseteq \mathbb{Z}^+\}.$$

Remark that the cardinality of \mathbb{Z} or \mathbb{Z}^+ is \aleph_0 (countably many) and the cardinality of \mathcal{I} is \aleph_1 (uncountably many) like the set of real numbers (\mathbb{R}).

The membership of each positive integer in any $I \in \mathcal{I}$ can be represented as a binary probability value:

$$p_I = 0.x_101x_201x_301 \cdots x_i01 \cdots, \quad x_i = 1 \leftrightarrow i \in I.$$

Quantumly, we can use a different representation, originally given in [1]. The membership of each positive integer in any $I \in \mathcal{I}$ can be represented as a single rotation on \mathbb{R}^2 with the angle:

$$\theta_I = 2\pi \sum_{i=1}^{\infty} \left(\frac{x_i}{8^{i+1}} \right), \quad \begin{array}{l} x_i = 1, \text{ if } i \in I \\ x_i = -1, \text{ if } i \notin I \end{array}.$$

3. MAIN RESULTS

We start with PTMs. Then, we focus on 2PCAs and PTMs with restricted head movements. Lastly, we present our quantum results.

3.1. Probabilistic Turing machines

It is known that poly-time PTMs can recognize uncountably many languages with bounded error by using uncomputable transition probabilities [7]. The k th bit in the decimal expansion of the probability that a given biased coin will land heads can be estimated by a procedure that involves tossing that coin for a number of times that is exponential in k . Given any unary language L on the alphabet $\{a\}$, and a coin which lands heads with probability $0.x$, where x is an infinite sequence of digits whose k th member encodes whether the k th unary string is in L , the language $\{a^{4^k} \mid a^k \in L\}$ is recognized by a PTM with bounded error. The machine in this construction uses logarithmic space.

In this section, we improve this result and show that bounded-error probabilistic models can recognize uncountably many languages with less resources. We start with a technical lemma.

Lemma 3.1. *Let $x = x_1x_2x_3 \cdots$ be an infinite binary sequence. If a biased coin lands on head with binary probability value $p = 0.x_101x_201x_301 \dots$, then the value x_k can be determined with probability at least $\frac{3}{4}$ after 64^k coin tosses.*

Proof. Let X be the random variable denoting the number of heads after 64^k coin flips. The expected value of X is $E[X] = p \cdot 64^k$. The value of x_k is equal to $(3 \cdot k - 2)$ th bit in $E[X]$. If $|X - E[X]| \leq 8^k$ we still have the correct x_k since in $E[X]$ ($= x_101x_201x_301 \cdots x_k01 \cdots$) x_k01 is followed by $3k$ bits and if we add a number in

the interval $[-8^k, 8^k]$ to $E[X]$, we can get a number between

$$x_1 0 1 x_2 0 1 x_3 0 1 \cdots x_k 0 0 \cdots \text{ and } x_1 0 1 x_2 0 1 x_3 0 1 \cdots x_k 1 0 \cdots .$$

By using this fact with Chebyshev's inequality, we can conclude that

$$Pr[|X - E[X]| \geq 8^k] \leq \frac{p \cdot (1-p) \cdot 64^k}{(8^k)^2} = \frac{p \cdot (1-p) \cdot 64^k}{64^k} = p \cdot (1-p),$$

where the function $p \cdot (1-p)$ is parabolic and its global maximum is $\frac{1}{4}$, *i.e.*, $p \cdot (1-p) \leq \frac{1}{4}$ for any chosen probability p .

Therefore, by outputting the $(3k-2)$ th digit of the counter value that keeps the number of heads after 64^k coin tosses, we can correctly guess x_k with the probability at least $\frac{3}{4}$. \square

Now, we show that $O(\log \log n)$ space is enough to recognize uncountably many languages.

Theorem 3.2. *Poly-time bounded-error unary PTMs can recognize uncountably many languages in $O(\log \log n)$ space.*

Proof. For our purpose, we define languages based on the following unary language given by Alt and Mehlhorn in 1975 [2]:

$$\text{AM75} = \{a^n \mid n > 0 \text{ and } F(n) \text{ is a power of } 2\},$$

where $F(n) = \min\{i \mid i \text{ does not divide } n\} \in \{2, 3, 4, \dots\}$. It is known that $O(\log \log n)$ -space DTMs can recognize AM75. It is clear that the following language

$$\text{AM75}' = \{a^n \mid n > 0 \text{ and } F(n) \text{ is a power of } 64\}$$

can also be recognized by $O(\log \log n)$ -space DTMs. For the sake of completeness, we provide the details of the algorithm (see also [24]).

Assume that the input is $w = a^n$ for some $n > 0$. In order to check if a number k written in binary on the work tape divides n , we can use $O(\log k)$ space for binary values of k that form a counter, and then we can check whether $n \bmod k$ is equal to zero or not. In order to compute $F(n)$, we can check each $k = 2, 3, 4, \dots$ in order to determine the first k such that $n \bmod k \neq 0$. It is known that (see Lem. 4.1.2(d) in [24]), $F(n) < c \cdot \log n$ for some constant c . Therefore, we use $O(\log \log n)$ space to find $F(n)$. Remark that when the number $F(n)$ is found, it is written on the work tape, and it is easy to check whether this number is a power of 64, *i.e.*, it must start with 1 and should be followed by only zeros, and the number of zeros must be a multiple of 6 ($64 = 2^6$).

For any $I \in \mathcal{I}$, we can define a corresponding language:

$$\text{AM75}'(I) = \{a^n \mid a^n \in \text{AM75}' \text{ and } \Sigma^*(\log_{64} F(n)) \in I\}.$$

(For the definition of $\Sigma^*(\cdot)$, please see the second paragraph of Sect. 2.)

For any input a^n , we can deterministically check whether $a^n \in \text{AM75}'$ by using the above algorithm. If not, the input is rejected. Otherwise, we continue with a probabilistic procedure. Remark that the work tape can still contain the binary value of $F(n)$ that is 64^m for some positive integers m in the beginning of the probabilistic procedure.

We use a biased coin landing on head with probability p_I encoding the memberships of positive integers in I as described before.

By definition, we know that $a^n \in \text{AM75}'(I)$ if and only if $m \in I$. So, if we compute the value of x_m correctly, we are done. Since the work tape contains the value of 64^m , we can toss this biased coin 64^m times and count

the number of heads. Due to Lemma 3.1, we know that we can correctly compute x_m with probability at least $\frac{3}{4}$. Here the number of heads is kept in binary and we check the $(3m - 2)$ th bit of the result after finishing the all coin tosses. By executing the probabilistic procedure a few more times, the success probability can be increased. Remark that the space used on the work tape does not exceed $O(\log \log n)$ and so the running time is polynomial in n .

The cardinality of the set of all subsets of positive integers is uncountably many and so the cardinality of the following set

$$\{\text{AM75}'(\mathbb{I}) \mid I \subseteq \mathbb{Z}^+\}$$

is also uncountably many, each element of which is recognized by a poly-time bounded-error unary PTM using $O(\log \log n)$ space. \square

With polynomial expected time, we cannot do better since it was proven that polynomial-time PTMs using $o(\log \log n)$ space can recognize only regular languages even with unrestricted transition probabilities [14].

On the other hand, with super-polynomial expected time, PTMs can recognize nonregular binary languages even with constant space [15]. Here we show that PTMs can recognize uncountably many binary languages with arbitrary small non-constant space, and we leave open the case of constant space. Regarding unary languages, we know that constant-space PTMs and $o(\log \log n)$ -space 1PTMs can recognize only regular languages [16, 17], and, up to our knowledge, it is still open whether PTMs can recognize a unary nonregular language with $o(\log \log n)$ space.

For our purpose, we use a fact given by Freivalds in [15], after a slight modification in order to keep the input alphabet binary: for any binary language $L \subseteq \{0, 1\}^*$, we define another language $\text{LOG}(L)$ as follows:

$$\text{LOG}(L) = \{0(1w_1)0^{2^1} (1w_2)0^{2^2} (1w_3)0^{2^3} \dots 0^{2^{m-1}} (1w_m)0^{2^m} \mid w = w_1 \dots w_m \in L\}.$$

Fact 3.3. [15] If a binary language L is recognized by a bounded-error PTM in space $s(n)$, then the binary language $\text{LOG}(L)$ is recognized by a bounded-error PTM in space $\log(s(n))$.

Theorem 3.4. For any $I \in \mathcal{I}$, the language $\text{LOG}(\text{AM75}'(\mathbb{I}))$ can be recognized by a bounded-error PTM in space $O(\log \log \log(n))$.

Proof. It follows from Theorem 3.2 and Fact 3.3. \square

Similarly, we can conclude that the language $\text{LOG}^k(\text{AM75}'(\mathbb{I}))$ for $k > 1$ can be recognized by a bounded-error PTM in space $O(\log^{k+2}(n))$.

Corollary 3.5. The cardinality of languages recognized by bounded-error PTMs with arbitrary small non-constant space bound is uncountably many.

3.2. Probabilistic counter machines

In this section, we present some results for 2PCAs. Remark that any $s(n)$ -space counter can be simulated by $\log(s(n))$ -space work tape.

It is easy for a 2PCA to check whether any specific part of the input has length of 64^k for some $k > 0$, and so, they can easily toss a biased coin for 64^k times, and then count the number of heads on the counter. However, it is not trivial to read certain digits of the result on the counter, and so we use a clever trick here.

Theorem 3.6. Bounded-error linear-time (linear-space) 2PCAs can recognize uncountably many languages.

Proof. We start with the definition of a new language:

$$\text{DIMA} = \{0^{2^0} 10^{2^1} 10^{2^2} 1 \dots 10^{2^{3k+1}} 110^{2^{3k+2}} 110^{2^{3k+3}} 1 \dots 10^{2^{6k}} \mid k > 0\}.$$

Remark that each member is composed by $(6k + 1)$ zero-blocks separated by single 1's except two special separators "11" that are used as the markers to indicate the $(3k + 3)$ th block, the length of which is 2^{3k+2} .

The language DIMA can be recognized by a 2DCA, say D . First it checks that the input starts with a single 0 and then ends with some 0's, all separators are 1's except two of them, which are "11" and consecutive, and the number of zero-blocks is $6k + 1$ for some $k > 0$. For all these checks, D can use only its internal states. Then, by using its counter, it can check that the length of each zero-block (except the first one) is double of the length of previous block. Similarly, it can check the equality of the number of zero-blocks before the first "11" and the number of zero-blocks after the first "11" plus 3, *i.e.*, $3k + 2$ versus $(3k - 1) + 3$. If one of these checks fails, then the input is rejected immediately. Otherwise, it is accepted. Remark that D can finish its computation in linear time and the counter value never exceeds the input length.

For any $I \in \mathcal{I}$, we define a new corresponding language:

$$\text{DIMA}(I) = \{w \in \{0, 1\}^* 10^m \mid m > 0, w \in \text{DIMA}, \text{ and } \Sigma^*(\log_{64} m) \in I\}.$$

(For the definition of $\Sigma^*(\cdot)$, please see the second paragraph of Sect. 2.) For any such I , we can construct a 2PCA recognizing $\text{DIMA}(I)$, say R_I , as desired. The machine R_I checks whether any given input, say w , is in DIMA deterministically by using D . If the input is not rejected by D , we continue with a probabilistic procedure. Since the last zero-block has the length of $m = 64^k$, by reading this block R_I can toss 64^k biased coins that land on head with probability p_I . The number of heads is counted on the counter. Similar to the proof of Theorem 3.2, the only remaining task is to determine the $(3k - 2)$ th bit of the binary value of the counter, which is x_k . The bit x_k in $E[X] = p_I \cdot 64^k$ is followed by $3k + 2$ bits.

The number of heads on the counter, say C , can be written as a binary number as follows:

$$C = \sum_{i=0}^{6k} a_i 2^i = a_{6k} 2^{6k} + \dots + a_{3k+2} 2^{3k+2} + a_{3k+1} 2^{3k+1} + \dots + a_1 2 + a_0,$$

where each $a_i \in \{0, 1\}$. Remark that x_k is corresponding to a_{3k+2} , *i.e.*, $3k + 2 = 6k - (3k - 1) + 1$. We can rewrite C as

$$C = B_1 2^{3k+3} + a_{3k+2} 2^{3k+2} + B_0 = B_1 2^{3k+3} + C',$$

where B_0 and B_1 are integers, $B_0 < 2^{3k+2}$, and $C' = a_{3k+2} 2^{3k+2} + B_0$.

After tossing-coin part, R_I moves its head to the second symbol of the first "11" and then the automaton enters a loop. In each iteration, the head moves to the next separator on the right by reading 2^{3k+2} 0's and then comes back by reading the same amount of 0's. In each iteration, R_I tries to subtract 2^{3k+2} , twice of (2^{3k+3}) .

If $C' = 0$, then R_I hits to the zero value on the counter when the head is at the starting position of the loop. This means $a_{3k+2} = x_k = 0$, and so the input is rejected by the automaton R_I . If $C' \neq 0$, then R_I hits to the zero value on the counter, say in the j th iteration $j = 0, 1, \dots$, when the head is not at the starting position of the loop. The value of counter is C' before starting the j th iteration, and there are two cases, $x_k = 1$ or $x_k = 0$. If $x_k = 1$, R_I hits to the zero value on the counter only after reading the first 2^{3k+2} 0's. In this case, the input is accepted. Otherwise, R_I hits to the zero value on the counter before finishing to read the first 2^{3k+2} 0's. Then, the input is rejected.

It is clear that the value of the counter never exceeds the length of the input. Moreover, both deterministic and probabilistic parts finish in linear time. \square

By relaxing the time constraint, we can obtain similar results for arbitrary small non-constant space on the counter like PTMs.

Theorem 3.7. *For any $I \subseteq \mathcal{I}$, the language $\text{LOG}(\text{DIMA}(I))$ can be recognized by a bounded-error 2PCA that uses $O(\log(n))$ space on the counter.*

Proof. Let R'_I be our desired 2PCA. The definition of $\text{LOG}(\text{DIMA}(\mathbb{I}))$ is

$$\{0(1w_1)0^{2^1}(1w_2)0^{2^2}(1w_3)0^{2^3} \dots 0^{2^{m-1}}(1w_m)0^{2^m} \mid w = w_1 \dots w_m \in \text{DIMA}(\mathbb{I})\}.$$

The automaton can deterministically check whether the input is of the form $0(1\{0,1\}0^+)^+$. If not, the input is rejected. If so, we can assume that the input is of the form

$$0^+(1w_1)0^+(1w_2)0^+(1w_3)0^+ \dots 0^+(1w_m)0^+$$

and the computation continues.

If we are sure that each zero block (except the first one) has double length of the previous zero block, R'_I executes R_I on $w = w_1w_2 \dots w_m$ by giving the same answer as R_I , and so we are done. In such case, R_I uses linear space on the counter in m , which is logarithm of the input length.

It is clear that if we use the counter to compare the length of zero blocks in regular way, then the value of counter cannot be sub-linear. On the other hand, as shown by Freivalds [15], 2PFAs can make such a sequence (unary) equality checks with high probability (see Lem. 2 in [15]). (The only drawback is that 2PFAs require exponential expected time for these checks [14].)

Thus, after the first deterministic check, R'_I determines the well form of zero blocks with high probability without using its counter. If the zero blocks are well formed, it calls R_I on w . Otherwise, the input is rejected. \square

Similarly, we can conclude that the language $\text{LOG}^k(\text{DIMA}(\mathbb{I}))$ for $k > 1$ can be recognized by a bounded-error 2PCA that uses $O(\log^k(n))$ space on the counter.

Corollary 3.8. *The cardinality of languages recognized by bounded-error 2PCAs with arbitrary small non-constant space bound is uncountably many.*

3.3. One-way and sweeping probabilistic machines

Here we present the results when having further restrictions.

Theorem 3.9. *Linearithmic-time bounded-error one-way unary PTMs can recognize uncountably many languages in $O(\log n)$ space.*

Proof. We start with the definition of language UPOWER64 :

$$\text{UPOWER64} = \{0^{2^{6k}} \mid k > 0\}.$$

This language is recognized by 1DTMs in $O(\log n)$ space, where n is the length of the input. A binary counter on the work tape is used to count the number of zeros in the input. This can be done in a straightforward way. For each input symbol, the value of the counter is increased by 1. Remark that any update on the counter can be done in $O(\log n)$ steps. Once the whole input is read, the counter is checked whether it is a power of 64, *i.e.*, it must start with 1 and should be followed by only zeros and the number of zeros must be a multiple of 6 ($64 = 2^6$). The overall running time is $O(n \log n)$.

As in Theorem 3.2, for any $I \in \mathcal{I}$, we can define a corresponding language:

$$\text{UPOWER64}(\mathbb{I}) = \{0^n \mid 0^n \in \text{UPOWER64} \text{ and } \Sigma^*(\log_{64} n) \in I\}.$$

(For the definition of $\Sigma^*(\cdot)$, please see the second paragraph of Sect. 2.) We again use a biased coin landing on head with probability p_I . When we read the input and count the number of zeros, we can in parallel toss the biased coin and count the number of heads in a second counter on the work tape. After reading the whole input, for the inputs in UPOWER64 , the decision is given by checking the $(3k - 2)$ th bit of the second counter. This additional probabilistic procedure does not change the runtime and space asymptotically. \square

The algorithm given in the proof of Theorem 3.2 for the language

$$\text{AM75}'(\mathcal{I}) = \{a^n \mid a^n \in \text{AM75}' \text{ and } \Sigma^*(\log_{64} F(n)) \in I\}$$

does not need to change the direction of head on the a 's. So, we can call that PTM sweeping.

Corollary 3.10. *Polynomial-time bounded-error sweeping unary PTMs can recognize uncountably many languages in $O(\log \log n)$ space.*

Theorem 3.11. *Bounded-error linear-space sweeping 2PCAs can recognize uncountably many languages in subquadratic time.*

Proof. We modify the algorithms given in the proof of Theorem 3.6. Remark that the algorithms given there run in linear time. Here the algorithms run in super-linear time. First, we show how to deterministically recognize the language DIMA in sweeping reading mode, *i.e.*,

$$\text{DIMA} = \{0^{2^0} 10^{2^1} 10^{2^2} 1 \dots 10^{2^{3k+1}} 110^{2^{3k+2}} 110^{2^{3k+3}} 1 \dots 10^{2^{6k}} \mid k > 0\}.$$

With one pass (reading the input from the left end-marker to the right end-marker), the input is checked without using counter whether having the following form

$$01(0^+1)^+110^+11(0^+1)^+0^+,$$

and the number of 0-blocks is $6k + 1$ for some $k > 0$. Moreover, for a member, the number of 0-blocks before the first “11” is $3k + 2$, and the number of 0-blocks after the second “11” is $3k - 2$. Therefore, by using the counter, we can check that the number of 0-blocks before the first “11” is 4 more than the number of 0-blocks after the second “11”. If any of these checks fails, then the input is immediately rejected.

In the second pass (reading the input from the right end-marker to the left end-marker), it is checked that, for each $0 < i \leq 3k$, $(2i + 1)$ th 0-block has twice more zeros than $(2i)$ th 0-block.

In the third pass (reading the input from the left end-marker to the right end-marker), it is checked that, for each $0 < i \leq 3k$, $(2i - 1)$ th 0-block has twice less zeros than $(2i)$ th 0-block.

Thus, in three passes, DIMA can be recognized by a sweeping PCA.

Then, as in the proof of Theorem 3.6, for any $I \in \mathcal{I}$, we consider the language:

$$\text{DIMA}(\mathcal{I}) = \{w \in \{0, 1\}^* 10^m \mid m > 0, w \in \text{DIMA}, \text{ and } \Sigma^*(\log_{64} m) \in I\}.$$

If the given input is in DIMA, then we continue with the probabilistic procedure. (Otherwise, the input is rejected.) We perform the same walk as in the proof of Theorem 3.6, but, due to sweeping reading mode, each walk can be done from one end-marker to the other end-marker. But the presence of symbols “11” allows us to follow the same procedure only with slowdown. The running time is $O(2^{3k})O(2^{6k}) = O(2^{9k})$ and it is super-linear and subquadratic in the length of input. To be more precise, the running time is $O(n\sqrt{n})$ (where n is the length of the input). \square

Some algorithms can be space sufficient only for the members. That is known as recognition with middle space [24]. The standard space usage is then called as recognition with strong space. Here we show that sweeping 2PCAs can recognize uncountably many languages also for middle logarithmic space.

Theorem 3.12. *Middle logarithmic-space sweeping 2PCAs can recognize uncountably many languages with bounded error.*

Proof. We will construct a 2PCA, say P , recognizing $\text{LOG}(\text{DIMA}(\mathcal{I}))$ for $I \in \mathcal{I}$, where any member is of the form

$$0(1w_1)0^{2^1}(1w_2)0^{2^2}(1w_3)0^{2^3} \dots 0^{2^{m-1}}(1w_m)0^{2^m}, \quad (3.1)$$

for $w = w_1 \dots w_m \in \text{DIMA}(\mathbf{I})$.

First, in a single pass, the input is deterministically checked whether it is of the form $0(1\{0, 1\}0^+)^+$. If not, it is rejected immediately. Otherwise, P continues with assumption that the input is

$$0^{j_0}(1w_1)0^{j_1}(1w_2)0^{j_2}(1w_3)0^{j_3} \dots 0^{j_{m-1}}(1w_m)0^{j_m},$$

where $m > 0$ and $j_0 = 1$.

Then, P checks, for each $i = 1, \dots, m$, whether j_i is twice of j_{i-1} or not. For such a check, we can use the 2PFA algorithm given by Freivalds, say M , in [15] (*i.e.*, Lems. 1 and 2). The 2PFA M executes another 2PFA algorithm for equality check, say M_{EQ} , as a subroutine on each pair $0^{j_{i-1}}(1w_i)0^{j_i}$ many times, where $1 \leq i \leq m$. Once this check is successfully done (with high probability), then the determination whether $w \in \text{DIMA}(\mathbf{I})$ can be done by P in sweeping mode as described in the proof of Theorem 3.11 by using logarithmic space.

Therefore, the only tricky part is to execute the tasks done by M in sweeping mode. The 2PFA M_{EQ} can read each pair already in sweeping mode, *i.e.*, it reads the pair in an infinite loop and in sweeping mode, and the loop is terminated with probability 1 in exponential expected time. Thus, M can also read whole input in sweeping mode if it knows the pair on which M_{EQ} is executed in every pass. By using the counter, the active pair can be easily traced: for i th pair, the counter has the value of $i - 1$ when on the left end-marker. By reading the counter value, the head is placed on the first symbol of the i th pair, and the value of counter is set to zero. Then, by going to the right end-marker, the value of the counter is set to $m - i - 1$. By reading the counter value, the head is placed on the last symbol of the i th pair, and the value of the counter is set to zero. Then, by going to the left end-marker, the value of the counter is set to $i - 1$ again. When the active pair is changed to the next one, the values of the counter are set to i and $m - i - 2$ on the left and right end-markers, respectively. Hence, the overall tasks done by M can be easily executed in sweeping mode.

It is clear that for members the value of the counter never exceeds $O(m)$, and so the space used by P is logarithmic in the length of the input. However, for non-members there is no such a guarantee. \square

3.4. Quantum models

For any $I \in \mathcal{I}$, we can compute the membership of the positive integer j in I by using the technique described in [1, 7]. We call it Procedure ADH.

The qubit spanned by $\{|q_1\rangle, |q_2\rangle\}$ is set to $|q_1\rangle$. Then, it is rotated with angle $\theta_I 8^j$ times, which leaves the quantum state having angle

$$8^j \cdot 2\pi \sum_{i=1}^{\infty} \left(\frac{x_i}{8^{i+1}} \right) = \pi \left(\frac{x_j}{4} \right) + 2\pi \sum_{i=j+1}^{\infty} \left(\frac{x_i}{8^{i-j+1}} \right)$$

from the initial position. After an additional rotation by $\frac{\pi}{4}$, the final angle from $|q_1\rangle$ is $\frac{\pi}{2} + \delta$ if $x_j = 1$ ($j \in I$) and it is δ if $x_j = -1$ ($j \notin I$), where δ is sufficiently small such that the probability of the qubit being in $|q_2\rangle$ ($|q_1\rangle$) is bigger than 0.98 if $j \in I$ ($j \notin I$).

Say and Yakaryılmaz [7] presented a bounded-error poly-time 2QCFA algorithm, say M , for

$$\text{POWER-EQ} = \{aba^7ba^{7 \cdot 8}ba^{7 \cdot 8^2}ba^{7 \cdot 8^3}b \dots ba^{7 \cdot 8^m} \mid m \geq 0\}.$$

Remark that every member has 8^{m+1} a 's for some $m \geq 0$. It is clear that for any member of POWER-EQ having 8^{m+1} a 's and for any $I \in \mathcal{I}$, M can be modified, say M_I , in order to determine whether m is in I or not by using

Procedure ADH with high probability. So, M_I can recognize the following language with bounded error [7]

$$\text{POWER-EQ}(I) = \{w \in \{a, b\}^* \mid w \in \text{POWER-EQ and } \log_8(|w|_a) \in I\}.$$

Then, we can conclude that 2QCFAs can recognize uncountably many languages with bounded error.

Procedure ADH can be trivially implemented by an rtQCFA having a single qubit, say R_I for $I \in \mathcal{I}$. So, if we show that POWER-EQ is recognized by a restarting rtQCFA, say R , then, we can conclude that restarting rtQCFAs can recognize uncountably many languages. Since R can execute R_I in parallel to its original algorithm, *i.e.*, R and R_I are tensorred such that if R is in the restarting state, then all computation is restarted; otherwise, the input is accepted if and only if both R and R_I give the decision of “accepting”. The obtained restarting rtQCFA gives its decisions with bounded error. We refer the reader how to [29] for the technical details to obtain a restarting bounded-error rtQCFA by tensoring two bounded-error restarting rtQCFAs, where the results are given for general realtime QFA models, but it can be obtained for rtQCFAs in the same way since general realtime QFA models and rtQCFAs can simulate each other exactly.

Theorem 3.13. *The language POWER-EQ can be recognized by a restarting rtQCFA R with bounded error.*

Proof. The quantum part of R has 9 states ($\{|q_1, \dots, q_9\rangle\}$) but only the first three of them are used to process useful information. Before each unitary operation, the quantum state has always zeros in the entries coming after these significant first three entries

$$(\alpha_1 \ \alpha_2 \ \alpha_3 \ 0 \ \cdots \ 0)^T.$$

After we apply a unitary operator U , we obtain a new quantum state. Then, we make some measurements such that if the system is in $\text{span}\{|q_4\rangle, \dots, |q_9\rangle\}$, then the computation is always restarted. So, if the computation is not restarted, the new quantum state has always zeros for the last six entries.

$$(\alpha'_1 \ \alpha'_2 \ \alpha'_3 \ 0 \ \cdots \ 0)^T.$$

Sometimes the measurement operator can also affect the first three states that will be specified later.

Each unitary operator is a (9×9) -dimensional unitary matrix. However, the significant parts are the top-left (3×3) -dimensional matrices due to the measurement operators. So, we can trace the computation only by a 3-dimensional vector and (3×3) -dimensional matrices.

We describe our algorithm with integer matrices with a real coefficient $0 < l < 1$:

$$lA = l \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}.$$

Remark that lA is the top-left corner of a unitary matrix, and all the other entries can be filled arbitrarily providing that the matrix is unitary. For our purpose, first we define our (3×3) -dimensional matrices A 's, which do the main tasks, and then complete the missing parts of unitary matrices by picking a fixed l for all A 's. We refer the reader to [27, 30] for the details of how to pick nonnegative real $l < 1$ and fill the missing parts of unitary matrices.

After a measurement operator, we can obtain more than one unnormalized state vector having norm less than 1. Depending on the measurement outcome, the system collapses into one of them and then the corresponding unnormalized state vector is normalized (norm-1 vector). On the other hand, since the probabilities can be calculated directly from the entries of unnormalized state vectors, we trace the computation with unnormalized state vectors.

After all these technical descriptions, we can give the details of our quantum algorithm. (QFA algorithms based on such assumptions have been presented before (*e.g.*, see [26]).)

The quantum part is in state $|v_0\rangle = (1 \ 0 \ 0)^T$ at the beginning. If the input does not start with aba^7b , then it is rejected deterministically. Otherwise, by reading 7 a 's, the quantum part is set to

$$|\widetilde{v}_7\rangle = l^7 \begin{pmatrix} 1 \\ 7 \cdot 56 \\ 0 \end{pmatrix} = \left(l \begin{pmatrix} 1 & 0 & 0 \\ 56 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \right)^7 \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}.$$

If the input is aba^7b , then it is accepted deterministically.

In the remaining part, we assume that the input is of the form $aba^7b(a^+b)^+$. Otherwise, the input is rejected deterministically. Remark that, after each quantum step, the computation is restarted with some probability, and so the computation in a single round can reach to the end-marker only with a very small (exponentially small in the input length) probability.

At the beginning of each block of a 's, say the i th block, the quantum state is

$$|\widetilde{v}_{t_{i-1}}\rangle = l^{t_{i-1}} \begin{pmatrix} 1 \\ 8S_{i-1} \\ 0 \end{pmatrix},$$

where

- $|\widetilde{v}_{t_{i-1}}\rangle$ is the non-halting *unnormalized* quantum state vector,
- the first block ($i = 1$) refers the first a 's after aba^7b ,
- S_{i-1} is the number of a 's in the previous block with $S_0 = 7$, and
- $t_{i-1} = S_0 + S_1 + \dots + S_{i-1} + i - 1$ is the number of unitary operators applied until that step, *i.e.*, 7 unitary operators are applied on the input aba^7b (the other quantum operators on aba^7 can be assumed as identity operator), and, then for each symbol after aba^7b , a unitary operator is applied ($i - 1$ refers the number of b 's).

Remark that the expected number of a 's in the i th block is already written as the amplitude of the $|q_2\rangle$, *i.e.*, for any member, the number of a 's in the i th block is 8 times of the number of a 's in the previous block.

During reading the i th block, the number of a 's is counted and kept as the amplitude of $|q_3\rangle$:

$$l^{t_{i-1}+j} \begin{pmatrix} 1 \\ 8S_{i-1} \\ j \end{pmatrix} = l \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} l^{t_{i-1}+j-1} \begin{pmatrix} 1 \\ 8S_{i-1} \\ j-1 \end{pmatrix}.$$

After reading all a 's in the block and just before reading b , the quantum state is

$$l^{t_{i-1}} \begin{pmatrix} 1 \\ 8S_{i-1} \\ S_i \end{pmatrix}.$$

Then, we obtain the following vector after reading b and just before the measurement:

$$l^{t_i} \begin{pmatrix} 1 \\ 8 \cdot S_i \\ S_i - 8S_{i-1} \end{pmatrix} = l \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 8 \\ 0 & -1 & 1 \end{pmatrix} l^{t_{i-1}} \begin{pmatrix} 1 \\ 8S_{i-1} \\ S_i \end{pmatrix}.$$

After this, the measurement operator, in addition to the previously described standard behavior, also checks whether the system is in $\text{span}\{|q_1\rangle, |q_2\rangle\}$ or $\text{span}\{|q_3\rangle\}$. In the latter case, the input is rejected, and the

computation continues, otherwise. Here we have two cases:

- If $S_i \neq 8S_{i-1}$, then $S_i - 8S_{i-1}$ is a nonzero integer, and so, the input is rejected with probability at least l^{2t_i} .
- If $S_i = 8S_{i-1}$, then the input is rejected with zero probability.

That is, for any non-member, the input is rejected after a block with some nonzero probability. For each member, on the other hand, the input is never rejected until the end of the computation.

At the end of the computation, the state before reading the right end-marker is

$$l^{t_m} \begin{pmatrix} 1 \\ 8 \cdot S_m \\ 0 \end{pmatrix}$$

if there are m blocks of a 's. Then, we obtain the following quantum state after reading the right end-marker

$$l^{t_m+1} \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = l \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} l^{t_m} \begin{pmatrix} 1 \\ 8 \cdot S_m \\ 0 \end{pmatrix},$$

and the input is accepted if $|q_1\rangle$ is observed. Then, the input is accepted with probability l^{2t_m+2} , which is clearly at least l^2 times of any possible rejecting probability before.

Now, we can analyze a single round of R , the period from the initial configuration to give a decision or to restart the computation, and then calculate the overall probabilities on the given input.

Any member is accepted with an exponentially small but non-zero probability (l^{2t_k+2}), and it is rejected with zero probability. So, in exponential expected time, the input is accepted with probability 1.

Any non-member, on the other hand, is again accepted with a very small non-zero probability, but it is also rejected with a probability sufficiently bigger than the accepting probability. So, in exponential expected time, the input is rejected with probability $\frac{R}{A+R}$ that is at least $\frac{R}{l^2R+R} = \frac{1}{1+l^2} > \frac{1}{2}$, where A and R are the accepting and rejecting probabilities, respectively, in a single round (see [28] for the details of calculating the overall rejecting probability). We can pick l arbitrarily small, and so the rejecting probability can be arbitrarily close to 1. \square

Corollary 3.14. *Exponential expected time restarting rtQCFAs can recognize uncountably many languages with bounded error.*

It is still open whether rtQCFAs can recognize a nonregular language with bounded error in polynomial time.

Now, we will consider middle space complexity results. We know that 2QCCAs can recognize the following nonregular unary language with bounded error in middle logarithmic space [5]:

$$\text{UPOWER2} = \{a^{2^n} \mid n \geq 0\}.$$

Here, the base-2 is not essential, and it can be replaced with any integer bigger than 2. Therefore, 2QCCAs can also recognize

$$\text{UPOWER8} = \{a^{8^n} \mid n \geq 0\}$$

with bounded error in middle logarithmic space (by slightly modifying the algorithm for UPOWER2). Moreover, for any $I \in \mathcal{I}$, 2QCCAs recognize the following language

$$\text{UPOWER8}(I) = \{a^{8^n} \mid n - 1 \in I\}$$

with bounded error in middle logarithmic space, *i.e.*, we first determine whether the input is of the form a^{8^n} with high probability. If so, we call Procedure ADH, which does not use the counter, to determine whether $(n - 1)$ is in I or not with high probability.

Theorem 3.15. *Unary middle logarithmic-space 2QCCAs can recognize uncountably many languages with bounded error.*

4. CONCLUDING REMARKS

We identify new small space and time bounds for bounded-error probabilistic and quantum models that can recognize uncountably many languages. We list below our positive cases with the related open cases.

- Unary languages:
 - Poly-time $O(\log \log n)$ -space sweeping PTMs (open for $o(\log \log n)$ -space)
 - Linearithmic-time $O(\log n)$ -space 1PTMs
 - Middle $O(\log n)$ -space 2QCCAs (open for better space bounds and/or poly-time)
- Binary languages:
 - $\omega(1)$ -space 2PCAs (open for $O(1)$ -space (or equivalently 2PFAs))
 - Linear-time $O(n)$ -space 2PCAs (open for poly-time $o(n)$ -space)
 - Restarting rtQCFAs (open for polynomial-time)
 - Subquadratic-time $O(n)$ -space sweeping PCAs
 - Middle $O(\log n)$ -space sweeping PCAs

We have extended our research with other restricted bounded-error models. In [10, 13], we focus on real-time PTMs. We show that unary $O(\log n)$ -space and binary $O(\log \log n)$ -space realtime PTMs can recognize uncountably many languages with bounded error. We also show that realtime probabilistic automata with two (resp., one) counters can recognize uncountably many unary (resp., binary) languages, and these bounds for the number of counters are tight.

In [12], we focus on probabilistic recognition and verification of all languages. We show that PTMs can recognize all unary and binary languages in linear and exponential space, respectively, and that PTMs can verify all unary and binary languages in $O(\log n)$ -space and $O(n)$ -space, respectively. In [12], we also show that PFAs can verify uncountably many unary and binary languages in quadratic expected time and linear time, respectively.

In [11], we focus on postselecting probabilistic recognizers and verifiers. We show that postselecting realtime PFA verifiers can verify uncountably many unary languages, and we also show that $\omega(1)$ -space postselecting realtime PCAs can recognize uncountably many binary languages. Since postselecting realtime models are more restricted than sweeping models, the last result is also valid for sweeping PCAs.

Acknowledgements. We thank to anonymous reviewers for their very helpful comments. Dimitrijevs was partially supported by University of Latvia projects AAP2016/B032 “Innovative information technologies” and ZD2018/20546 “For development of scientific activity of Faculty of Computing”. Yakaryılmaz was partially supported by ERC Advanced Grant MQC.

REFERENCES

- [1] L.M. Adleman, J. DeMarrais and M.-D.A. Huang, Quantum computability. *SIAM J. Comput.* **26** (1997) 1524–1540.
- [2] H. Alt and K. Mehlhorn, A language over a one symbol alphabet requiring only $O(\log \log n)$ space. *ACM SIGACT* **7** (1975) 31–33.
- [3] A. Ambainis and J. Watrous, Two-way finite automata with quantum and classical states. *Theor. Comput. Sci.* **287** (2002) 299–311.
- [4] A. Ambainis and A. Yakaryılmaz, Automata and quantum computing. Technical Report. Preprint [arXiv:1507.01988](https://arxiv.org/abs/1507.01988) (2015).
- [5] Z. Bednářová, V. Geffert, K. Reinhardt and A. Yakaryılmaz, New results on the minimum amount of useful space. *Int. J. Found. Comput. Sci.* **27** (2016) 259–282.

- [6] A.C. Cem Say and A. Yakaryılmaz, Quantum finite automata: A modern introduction, in *Computing with New Resources*. Vol. 8808 of *Lect. Notes Comput. Sci.* Springer, Cham (2014) 208–222.
- [7] A.C. Cem Say and A. Yakaryılmaz, Magic coins are useful for small-space quantum machines. *Quantum Inf. Comput.* **17** (2017) 1027–1043.
- [8] M. Dimitrijevs, Capabilities of ultrametric automata with one, two, and three states, in *Theory and Practice of Computer Science*. Vol. 9587 of *Lect. Notes Comput. Sci.* Springer, Berlin, Heidelberg (2016) 253–264.
- [9] M. Dimitrijevs and A. Yakaryılmaz, Uncountable classical and quantum complexity classes, in *Eighth Workshop on Non-Classical Models of Automata and Applications (NCMA 2016)*. Vol. 321. Austrian Computer Society, Australia, books@ocg.at (2016) 131–146.
- [10] M. Dimitrijevs and A. Yakaryılmaz, Uncountable realtime probabilistic classes, in *Descriptional Complexity of Formal Systems*. Vol. 10316 of *Lect. Notes Comput. Sci.* Springer, Berlin, Heidelberg (2017) 102–113.
- [11] M. Dimitrijevs and A. Yakaryılmaz, Postselecting probabilistic finite state recognizers and verifiers, in *Tenth Workshop on Non-Classical Models of Automata and Applications (NCMA 2018)*. Vol. 332. Austrian Computer Society, Australia, books@ocg.at (2018) 65–82.
- [12] M. Dimitrijevs and A. Yakaryılmaz, Probabilistic verification of all languages. Technical Report. Preprint [arXiv: 1807.04735](https://arxiv.org/abs/1807.04735) (2018).
- [13] M. Dimitrijevs and A. Yakaryılmaz, Recognition of uncountably many languages with one counter, in *Tenth Workshop on Non-Classical Models of Automata and Applications (NCMA 2018)*, Short Papers. (2018) 7–13.
- [14] C. Dwork and L. Stockmeyer, A time complexity gap for two-way probabilistic finite-state automata. *SIAM J. Comput.* **19** (1990) 1011–1123.
- [15] R. Freivalds, Probabilistic two-way machines, in *Proceedings of the International Symposium on Mathematical Foundations of Computer Science*. Springer-Verlag, London (1981) 33–45.
- [16] J. Kaņeps, Regularity of one-letter languages acceptable by 2-way finite probabilistic automata, in *FCT'91*. Springer, Berlin, Heidelberg (1991) 287–296.
- [17] J. Kaņeps and R. Freivalds, Minimal nontrivial space complexity of probabilistic one-way Turing machines, in *MFCS'90*. Vol. 452 of *Lect. Notes Comput. Sci.* Springer, Berlin, Heidelberg (1990) 355–361.
- [18] A. Kondacs and J. Watrous, On the power of quantum finite state automata, in *FOCS'97*. IEEE Computer Society Washington, DC (1997) 66–75.
- [19] M.A. Nielsen and I.L. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, Cambridge (2000).
- [20] A. Paz, *Introduction to Probabilistic Automata*. Academic Press, New York (1971).
- [21] M.O. Rabin, Probabilistic automata. *Inf. Control* **6** (1963) 230–243.
- [22] A.M. Shur and A. Yakaryılmaz, Quantum, stochastic, and pseudo stochastic languages with few states, in *UCNC 2014*. Vol. 8553 of *Lect. Notes Comput. Sci.* Springer, Cham (2014) 327–339.
- [23] A.M. Shur and A. Yakaryılmaz, More on quantum, stochastic, and pseudo stochastic languages with few states. *Nat. Comput.* **15** (2016) 129–141.
- [24] A. Szepietowski, *Turing Machines with Sublogarithmic Space*. Springer-Verlag, Berlin, Heidelberg (1994).
- [25] A. Yakaryılmaz, Superiority of one-way and realtime quantum machines. *RAIRO: ITA* **46** (2012) 615–641.
- [26] A. Yakaryılmaz, Public qubits versus private coins, in *The Proceedings of Workshop on Quantum and Classical Complexity*, ECCC:TR12-130. University of Latvia Press, Riga (2013) 45–60.
- [27] A. Yakaryılmaz and A.C. Cem Say, Languages recognized by nondeterministic quantum finite automata. *Quantum Inf. Comput.* **10** (2010) 747–770.
- [28] A. Yakaryılmaz and A.C. Cem Say, Succinctness of two-way probabilistic and quantum finite automata. *Discrete Math. Theor. Comput. Sci.* **12** (2010) 19–40.
- [29] A. Yakaryılmaz and A.C. Cem Say, Probabilistic and quantum finite automata with postselection. (A preliminary version of this paper appeared in the *Proceedings of Randomized and Quantum Computation (Satellite Workshop of MFCS and CSL 2010)*. (2010) 14–24.) Technical Report. Preprint [arXiv: 1102.0666](https://arxiv.org/abs/1102.0666) (2011).
- [30] A. Yakaryılmaz and A.C. Cem Say, Unbounded-error quantum computation with small space bounds. *Inf. Comput.* **279** (2011) 873–892.
- [31] A. Yakaryılmaz and A.C. Cem Say, Proving the power of postselection. *Fundam. Inform.* **123** (2013) 107–134.
- [32] A. Yakaryılmaz, R. Freivalds, A.C. Cem Say and R. Agadzanyan, Quantum computation with write-only memory. *Nat. Comput.* **11** (2012) 81–94.
- [33] S. Zheng, D. Qiu, L. Li and J. Gruska, One-way finite automata with quantum and classical states, in *Languages Alive*. Vol. 7300 of *Lect. Notes Comput. Sci.* Springer, Berlin, Heidelberg (2012) 273–290.