Baltic J. Modern Computing, Vol. 10 (2022), No. 3, pp. 560–573
https://doi.org/10.22364/bjmc.2022.10.3.26

Exclusive affine languages*

Abuzer YAKARYILMAZ

[1] Center for Quantum Computer Science, University of Latvia, Rīga, Latvia
[2] QWorld Association, Tallinn, Estonia, https://qworld.net

abuzer.yakaryilmaz@lu.lv

Abstract. We present two fundamental results about the computational power of affine finite automata (AfAs). First, we show that changing the cutpoints does not change the class of languages defined by AfAs. Second, we show that AfAs define more languages than probabilistic and quantum finite automata with exclusive cutpoints.

Keywords: Affine automata, exclusive languages, cutpoints, probabilistic automata, quantum automata

1 Introduction

Finite automata are seen as the most basic computational model, and they have widely been investigated as language recognizer (e.g., Sipser (2013)). A deterministic finite state automaton (DFA) divides the strings generated on a specified alphabet into two sets: the accepted strings and the rejected strings. The accepted strings form a language, and it is called the language recognized by this DFA. The set of all languages recognized by DFAs is called regular languages.

The probabilistic generalization of DFA is probabilistic finite automaton (PFA) (Rabin (1963)). On contrary to a DFA, instead of deciding whether any given input is in the language or not, a PFA assigns to each string an accepting probability in $[0, 1]$. In other words, a string is accepted by a PFA with some probabilities in $[0, 1]$. For a DFA, this accepting probability is either 0 (rejected) or 1 (accepted).

To define a language by a PFA, we use some thresholds called cutpoints in $[0, 1)$. Thus, every PFA P with cutpoint $\lambda \in [0, 1)$ defines a language, which is formed by the strings accepted with probability greater than λ. Remark that the same PFA may define different languages for different cutpoints. The set of languages recognized by

* Part of this manuscript was prepared while Yakaryılmaz was visiting Institute of Theoretical and Applied Informatics, Gliwice, Poland in August 2021.

PFAs with cutpoints is called stochastic languages, which is a superset of regular languages. Remark that real-valued PFAs or PFAs with real-valued cutpoints may define uncomputable languages (Rabin (1963); Paz (1971)).

One basic fact about stochastic languages is that the class does not change when fixing the cutpoint to any value in $(0, 1)$ (Paz (1971)). However, when the cutpoint is fixed to 0, then the set of languages recognized by PFAs is identical to regular languages. Indeed, a PFA can be seen as a nondeterministic finite automaton (NFA) when the cutpoint is fixed to 0, i.e., every transition with non-zero probability is a transition of the NFA.

Besides the default language recognition mode for PFAs with cutpoints, a language can be defined as the collection of the strings accepted by the given PFA with probabilities other than the specified cutpoint. Such language is called exclusive stochastic (Paz (1971)). The class of exclusive stochastic languages is a superset of regular languages but a proper subset of stochastic languages. A half decade open problem is whether the complement set of exclusive stochastic languages is a subset of stochastic languages or not.

The quantum counterpart of PFA is quantum finite automaton (QFA). Due to different physical or mathematical motivations, there are several QFA models in the literature (Ambainis and Yakaryılmaz (2021)). We consider here the most general QFA models. A QFA works based on the principal of quantum mechanics but still it assigns an accepting probability to any given input. Therefore, the framework for PFAs is also used for QFAs. It was shown that the class of languages recognized by QFAs with cutpoints is identical to stochastic languages (Yakaryılmaz and Say (2011)). On the other hand, nondeterministic version of QFAs are more powerful than NFAs, and interestingly the languages recognized by nondeterministic QFAs is identical to exclusive stochastic languages (Yakaryılmaz and Say (2010)). On the other hand, surprisingly, when we use the language recognition mode of the exclusive stochastic languages, QFAs still defines all and only exclusive stochastic languages, not a superset.

The main difference between PFAs and QFAs is that QFAs can have both negative-valued and positive-valued transitions, which may create interference and so some transitions may disappear, i.e., some outcomes may disappear. To have a similar affect classically, affine finite automaton (AfA[3]) was introduced as a quantum-like generalization of PFAs by introducing negative-valued transitions (Díaz-Caro and Yakaryılmaz (2016a)). Such generalization appeared in the literature before (Turakainen (1969)), but the normalization of values based on ℓ_1-norm to calculate the probabilities — similar to the effect of quantum measurement operators — had not been considered before, up to our knowledge.

The computational power of AfAs and their generalizations have been examined and compared with their probabilistic and quantum counterparts in a series of papers: Díaz-Caro and Yakaryılmaz (2016a); Villagra and Yakaryılmaz (2016); Belovs et al. (2017); Hirvensalo et al. (2017); Nakanishi et al. (2017); Ibrahimov et al. (2018); Villagra and Yakaryılmaz (2018); Hirvensalo et al. (2019, 2021); Ibrahimov et al. (2021); Khadieva and Yakaryılmaz (2021); Yakaryılmaz (2021); Nakanishi et al. (2022).

[3] Both PFAs and QFAs are linear system, but AfAs are almost linear system: they evolve linearly but the last weigting operator is non-linear. We use lowercase "f" to indicate this difference.

As the fundamental results, they were shown that AfAs recognize more languages than PFAs and QFAs with cutpoints or with bounded error[4], but nondeterministic AfAs are equivalent to nondeterministic QFAs (Díaz-Caro and Yakaryılmaz (2016a)). In this paper, we present two more fundamental results. The first one is that changing the cutpoints does not change the class of languages defined by AfAs with cutpoints. The second one is that AfAs define more languages than PFA and QFAs with exclusive cutpoints.

We assume the reader is familiar with the basics of automata theory and quantum computation (see e.g., Nielsen and Chuang (2000); Sipser (2013); Say and Yakaryılmaz (2014); Ambainis and Yakaryılmaz (2021)). In the next section, we present the notations and definitions to follow the rest of paper. We present our result about changing cutpoints in Section 3 and our results about exclusive affine languages in Section 4.

Part of this paper appeared in the technical report by Díaz-Caro and Yakaryılmaz (2016b), which had not been peer-reviewed before.

2 Preliminaries

For a given vector v, $v[i]$ is i-th entry, and for a given matrix A, $A[i, j]$ is its (i, j)-th entry. For a numeric value α, $|\alpha|$ is its absolute value, its length to the origin. For a given n-dimensional vector v, $|v|$ is its length in ℓ_1 norm, i.e., $|v| = |v[1]| + \cdots + |v[n]|$.

We denote the alphabet Σ not containing the left and right end-markers (resp.,¢ and \$), and $\widetilde{\Sigma}$ denotes $\Sigma \cup \{¢, \$\}$. The set of all strings including empty string (ε) defined on Σ is represented by Σ^*. For a given string w, $|w|$ is the length of w and $|w|_\sigma$ is the number of symbol σ in w. For any non-empty string w, $w[i]$ is its i-th symbol, where $1 \leq i \leq |w|$. For any automaton M, its accepting probability on the input w is represented as $f_M(w)$.

An affine state is a real-valued column vector with entry summation 1. An affine operator is a real-valued square matrix, each column of which is an affine state. If only non-negative values are allowed, then an affine state turns out be a probabilistic state also called as a stochastic vector, and an affine operator turns out to be a probabilistic operator also called as a stochastic matrix. If only 0s and 1s are allowed, then both the state and operator turn out to be deterministic.

A pure quantum state is a complex-valued column vector with length 1 in ℓ_2-norm. When a pure quantum state is measured, the outcome corresponding to an entry with value $\alpha \in \mathbb{C}$ is observed with probability $|\alpha|^2$. The most basic quantum evolution operator is a unitary matrix, which preserves the length of vectors. In general, a quantum system can be in a mixture of pure states with some probabilities that add up to 1. A quantum evolution operator is a linear mapping between such mixtures. We refer the reader to Nielsen and Chuang (2000) for further details.

[4] We believe that any linear automaton model does not go beyond the regular languages with bounded error, e.g., Jeandel (2007).

2.1 Affine finite automaton

Here we give the formal definition of AfAs. For the definition of PFAs and QFAs, we refer the reader to Paz (1971) and Ambainis and Yakaryılmaz (2021).

An n-state AfA M is a 5-tuple $(S, \Sigma, \{A_\sigma \mid \sigma \in \widetilde{\Sigma}\}, s_1, S_a)$, where

- $S = \{s_1, \ldots, s_n\}$ is the set of states,
- A_σ is the transition matrix for symbol $\sigma \in \widetilde{\Sigma}$ such that $A_\sigma[j,i]$ represents the transition value from s_i to s_j,
- s_1 is the initial state, and
- $S_a \subseteq S$ is the set of accepting state(s).

A given input, say $w \in \Sigma^*$, is read by M as $\tilde{w} = ¢w\$$ symbol by symbol from left to right. The left and right end-markers are used for pre- and post-processing. The initial affine state is $v_0 = (1 \;\; 0 \;\; \cdots \;\; 0)^T$. The affine state after i-th step is

$$v_i = A_{\tilde{w}[i]} v_{i-1},$$

where $1 \leq i \leq |\tilde{w}|$. The final affine state is $v_f = v_{|\tilde{w}|}$. Then, a non-linear operator called weighting is applied, and s_i is observed with probability $\dfrac{|v_f[i]|}{|v_f|}$. The accepting probability of M on w is

$$f_M(w) = \frac{\sum_{s_i \in S_a} |v_f[i]|}{|v_f|}.$$

2.2 Language recognition and language classes

The language $L \subseteq \Sigma^*$ recognized by automaton M with cutpoint $\lambda \in [0,1)$ is defined as

$$L = \{w \in \Sigma^* \mid f_M(w) > \lambda\}.$$

The class of languages recognized by PFAs (resp., QFAs and AfAs) with cutpoints is called stochastic (resp., quantum and affine) languages denoted SL (resp., QAL and AfL). We know by Yakaryılmaz and Say (2011) and Díaz-Caro and Yakaryılmaz (2016a) that

$$\mathsf{SL} = \mathsf{QAL} \subsetneq \mathsf{AfL}.$$

As a special case, by fixing the cutpoint to 0, nondeterministic classical, quantum, and affine automata models are defined (Yakaryılmaz and Say (2010); Díaz-Caro and Yakaryılmaz (2016a)). The class of languages recognized by NFAs (or PFAs with cutpoint 0) is regular languages (Rabin and Scott (1959)), denoted REG. The class of languages recognized by nondeterministic QFAs and AfAs are respectively denoted NQAL and NAfL, and we have the following relation:

$$\mathsf{REG} \subsetneq \mathsf{NQAL} = \mathsf{NAfL} \subsetneq \mathsf{SL}.$$

The language $L \subseteq \Sigma^*$ recognized by automaton M with exclusive cutpoint $\lambda \in [0,1]$ is defined as

$$L = \{w \in \Sigma^* \mid f_M(w) \neq \lambda\}.$$

The class of languages recognized by PFAs (resp., QFAs and AfAs) is called exclusive stochastic (resp., quantum and affine) languages denoted $\mathsf{SL}^{\neq}$ (resp., $\mathsf{QAL}^{\neq}$ and $\mathsf{AfL}^{\neq}$). We know by Yakaryılmaz and Say (2010) that

$$\mathsf{SL}^{\neq} = \mathsf{NQAL} = \mathsf{QAL}^{\neq}.$$

The complement classes of $\mathsf{AfL}^{\neq}$, $\mathsf{QAL}^{\neq}$, and $\mathsf{AfL}^{\neq}$ are respectively $\mathsf{SL}^{=}$, $\mathsf{QAL}^{=}$, and $\mathsf{AfL}^{=}$.

For given automata M_1 and M_2, we say that $L(M_1, \lambda_1)$ is equivalent to $L(M_2, \lambda_2)$, denoted

$$L(M_1, \lambda_1) \equiv L(M_2, \lambda_2),$$

if for any input $w \in \Sigma^*$,

1. $f_{M_1}(w) > \lambda_1 \leftrightarrow f_{M_2}(w) > \lambda_2$,
2. $f_{M_1}(w) = \lambda_1 \leftrightarrow f_{M_2}(w) = \lambda_2$, and
3. $f_{M_1}(w) < \lambda_1 \leftrightarrow f_{M_2}(w) < \lambda_2$.

3 Changing cutpoint

It is a folkloric result that for any given pair of n-state PFA (resp., QFA), say M_1, and a cutpoint $\lambda_1 \in [0, 1]$ and another cutpoint $\lambda_2 \in (0, 1)$, we can define a PFA (resp., QFA) M_2 with $(n + 1)$ states such that

$$L(M_1, \lambda_1) \equiv L(M_2, \lambda_2).$$

For AfAs, we obtain the same result with different state overheads.

Theorem 1. *Let M_1 be an n-state AfA and $\lambda_1 \in (0, 1)$, then for any $\lambda_2 \in (0, 1)$, we can define another AfA M_2 with $(n + 2)$ states such that*

$$L(M_1, \lambda_1) \equiv L(M_2, \lambda_2).$$

Proof. The AfA M_2 is obtained from M_1 by adding two more states and making certain modifications. Let $w \in \Sigma^*$ be the given input, and let v_f be the final vector of M_1 after reading $\tilde{w}$. We can represent v_f as the summation of two orthogonal vectors: $v_f = v_f^a + v_f^r$, where v_f^a is the projection of v_f on the space spanned by the accepting states, i.e., v_f^a is obtained from v_f by setting entries of non-accepting states to zeros, and $v_f^r = v_f - v_f^a$. We define $|A|$ and $|R|$ as the l_1-norms of v_f^a and v_f^r respectively. Remark that $f_M(w) = \dfrac{|A|}{|A| + |R|}$ and,

- if $f_M(w) > \lambda_1$, then $\dfrac{|A|}{|R|} > \dfrac{\lambda_1}{1 - \lambda_1}$,
- if $f_M(w) = \lambda_1$, then $\dfrac{|A|}{|R|} = \dfrac{\lambda_1}{1 - \lambda_1}$, and
- if $f_M(w) < \lambda_1$, then $\dfrac{|A|}{|R|} < \dfrac{\lambda_1}{1 - \lambda_1}$.

Here it is enough to show how to obtain the first inequality:

$$\frac{|A|}{|A|+|R|} > \lambda_1 \implies \frac{1}{\lambda_1} > \frac{|A|+|R|}{|A|} \implies \frac{1}{\lambda_1} > 1 + \frac{|R|}{|A|}$$

$$\implies \frac{1-\lambda_1}{\lambda_1} > \frac{|R|}{|A|} \implies \frac{|A|}{|R|} > \frac{\lambda_1}{1-\lambda_1}.$$

The AfA M_2 follows the same computation of M_1 except that it applies an additional affine transformation on the right end-marker \$, say $A'_{\$}$: The final state of M_2 is

$$u_f = A'_{\$} \begin{pmatrix} v_f \\ 0 \\ 0 \end{pmatrix},$$

and the effect of $A'_{\$}$ (the details are given below) is as follows:

1. each value of the accepting state(s) in v_f is multiplied by $\dfrac{\lambda_2}{\lambda_1}$,
2. each value of the non-accepting state(s) in v_f is multiplied by $\dfrac{1-\lambda_2}{1-\lambda_1}$,
3. the value of the $(n+1)$-th state in u_f is set to $\lambda_2(1-T)$, and
4. the value of the $(n+2)$-th state in u_f is set to $(1-\lambda_2)(1-T)$,

where T is the summation of all entries except the last two in u_f: $T = \sum_{i=1}^{n} u_f[i]$. It is easy to verify that u_f is an affine state:

$$\sum_{i=1}^{n+2} u_f[i] = T + \lambda_2 (1-T) + (1-\lambda_2)(1-T) = 1.$$

The details of $A'_{\$}$ are as follows. The easiest way to obtain the effects in items 1 and 2, the top-left $(n \times n)$-dimensional sub-matrix of $A'_{\$}$ is defined as the diagonal matrix $diag(d_1, \ldots, d_n)$, where d_i is $\dfrac{\lambda_2}{\lambda_1}$ if s_i is an accepting state, and it is $\dfrac{1-\lambda_2}{1-\lambda_1}$ if s_i is a non-accepting state. If $v_f = (x_1 \quad \cdots \quad x_n)^T$ with $x_1 + \cdots + x_n = 1$, then $u_n = (d_1 x_1 \quad \cdots \quad d_n x_n \quad * \quad *)^T$ with $d_1 x_1 + \cdots + d_n x_n = T$. We define $(n+1)$-th row of $A'_{\$}$ as

$$(\lambda_2(1-d_1) \quad \cdots \quad \lambda_2(1-d_n) \quad 1 \quad 0)^T.$$

Then, $(n+1)$-th entry of u_f is calculated as

$$\begin{aligned} \lambda_2(1-d_1)x_1 + \cdots + \lambda_2(1-d_n)x_n &= \lambda_2(x_1 + \cdots + x_n) - \lambda_2(d_1 x_1 + \cdots + d_n x_n) \\ &= \lambda_2 - \lambda_2 T = \lambda_2(1-T). \end{aligned}$$

In the same way, we define $(n+2)$-th row of $A'_{\$}$ as

$$((1-\lambda_2)(1-d_1) \quad \cdots \quad (1-\lambda_2)(1-d_n) \quad 0 \quad 1)^T.$$

Then, $(n+2)$-th entry of u_f is $(1-\lambda_2)(1-T)$. The complete description of $A'_{\$}$ is

$$A'_{\$} = \left(\begin{array}{cccc|cc} d_1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & d_2 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & d_n & 0 & 0 \\ \hline \lambda_2(1-d_1) & \lambda_2(1-d_2) & \cdots & \lambda_2(1-d_n) & 1 & 0 \\ (1-\lambda_2)(1-d_1) & (1-\lambda_2)(1-d_2) & \cdots & (1-\lambda_2)(1-d_n) & 0 & 1 \end{array}\right).$$

It is easy to see that each column summation is 1, and so $A'_{\$}$ is an affine operator.

The accepting states of M_2 is formed by the accepting state(s) of M_1 and the $(n+1)$-th state. For u_f, we similarly define u_f^a and u_f^r. Then, we define $|A'|$ and $|R'|$ as the l_1-norms of u_f^a and u_f^r:

$$|A'| = \frac{\lambda_2}{\lambda_1}|A| + \lambda_2 |1-T| \text{ and } |R'| = \frac{1-\lambda_2}{1-\lambda_1}|R| + (1-\lambda_2)|1-T|.$$

We are ready to verify our construction:

- If $f_{M_1}(w) = \lambda_1$, we have $\dfrac{|A|}{|R|} = \dfrac{\lambda_1}{1-\lambda_1}$. Then, we calculate the following ratio:

$$\frac{|A'|}{|R'|} = \frac{\dfrac{\lambda_2}{\lambda_1}|A| + \lambda_2|1-T|}{\dfrac{1-\lambda_2}{1-\lambda_1}|R| + (1-\lambda_2)|1-T|}$$

  We can replace $|A|$ with $|R|\dfrac{\lambda_1}{1-\lambda_1}$ in the above formula:

$$\frac{|A'|}{|R'|} = \frac{\lambda_2\left(\dfrac{|R|}{1-\lambda_1} + |1-T|\right)}{(1-\lambda_2)\left(\dfrac{|R|}{1-\lambda_1} + |1-T|\right)} = \frac{\lambda_2}{1-\lambda_2}.$$

  That means $f_{M_2}(w) = \lambda_2$.

- If $f_{M_1}(w) > \lambda_1$, we have $\dfrac{|A|}{|R|} > \dfrac{\lambda_1}{1-\lambda_1}$, and so there is $\delta > 0$ satisfying that $|A| = |R|\dfrac{\lambda_1}{1-\lambda_1} + \delta\lambda_2$. Then, we can replace $|A|$ in the same way:

$$\frac{|A'|}{|R'|} = \frac{\lambda_2\left(\dfrac{|R|}{1-\lambda_1} + \delta + |1-T|\right)}{(1-\lambda_2)\left(\dfrac{|R|}{1-\lambda_1} + |1-T|\right)} > \frac{\lambda_2}{1-\lambda_2}.$$

  That means $f_{M_2}(w) > \lambda_2$.

– For the case $f_{M_1}(w) < \lambda_1$, we obtain that $f_{M_2}(w) < \lambda_2$ in the same way by using the equality that $|A| = |R|\dfrac{\lambda_1}{1-\lambda_1} - \delta\lambda_2$ for some $\delta > 0$.

Therefore, $L(M_1, \lambda_1) \equiv L(M_2, \lambda_2)$. □

The construction in the above proof does not work when $\lambda_1 = 0$ or $\lambda_1 = 1$. Therefore, we present different constructions.

Theorem 2. *Let M_1 be an n-state AfA with $k < n$ non-accepting state(s), then for any $\lambda \in (0, 1)$, we can define another AfA M_2 with $(n + k)$ states such that*

$$L(M_1, 0) \equiv L(M_2, \lambda).$$

Proof. Suppose that the first k states of M_1 are non-accepting states: $\{s_1, \ldots, s_k\}$. The AfA M_2 is obtained by modifying M_1. The first k states of M_2 are non-accepting ($\{s_1, \ldots, s_k\}$) and all the others are accepting states ($\{s_{k+1}, \ldots, s_{n+k}\}$). Until reading the right end-marker, M_2 trace the computation of M_1 exactly with the same states. On the right end-marker, M_2 first applies the operator of M_1 and then applies an additional affine operator, say $A'_{\$}$, which (1) multiplies the value of each non-accepting i-th state ($1 \le i \le k$) with $(1 - \lambda)$ and (2) also transfers the values of the non-accepting states to the additional states by multiplying with λ, i.e., the value of the $(n + i)$-th state is set to λ times the value of the i-th state.

For a given input $w \in \Sigma^*$, let v_f and u_f be the final affine states of M_1 and M_2, respectively. Then, they are related as follows:

$$v_f = \begin{pmatrix} x_1 \\ \vdots \\ \dfrac{x_k}{x_{k+1}} \\ \vdots \\ x_n \end{pmatrix} \quad \text{and} \quad u_f = A'_{\$} \begin{pmatrix} x_1 \\ \vdots \\ \dfrac{x_k}{x_{k+1}} \\ \vdots \\ \dfrac{x_n}{0} \\ \vdots \\ 0 \end{pmatrix} = \begin{pmatrix} (1-\lambda)x_1 \\ \vdots \\ \dfrac{(1-\lambda)x_k}{x_{k+1}} \\ \vdots \\ \dfrac{x_n}{\lambda x_1} \\ \vdots \\ \lambda x_k \end{pmatrix} = \begin{pmatrix} u_r \\ \dfrac{u_{a_1}}{u_{a_2}} \end{pmatrix}.$$

It is easy to see that u_f is an affine state and $A'_{\$}$ can be constructed in a straightforward way.

If $f_{M_1}(w) = 0$, u_{a_1} is a zero vector, and so

$$f_{M_2}(w) = \frac{\lambda\,(|x_1| + \cdots + |x_k|)}{|x_1| + \cdots + |x_k|} = \lambda.$$

If $f_{M_1}(w) > 0$, then u_{a_1} is a non-zero vector, and so

$$f_{M_2}(w) = \frac{|x_{k+1}| + \cdots + |x_n| + \lambda\,(|x_1| + \cdots + |x_k|)}{|x_{k+1}| + \cdots + |x_n| + |x_1| + \cdots + |x_k|} > \lambda,$$

where the inequality can be easily verified by multiplying both sides with the denominator. □

Theorem 3. *Let M_1 be an n-state AfA with $k < n$ accepting state(s), then for any $\lambda \in (0,1)$, we can define another AfA M_2 with $(n+k)$ states such that*

$$L(M_1, 1) \equiv L(M_2, \lambda).$$

Proof. We use the same proof with a few modifications. The accepting states of M_1 and M_2 are the same: $\{s_1, \ldots, s_k\}$. For any input $w \in \Sigma^*$, the final states of M_1 and M_2 are respectively as follows:

$$v_f = \begin{pmatrix} x_1 \\ \vdots \\ x_k \\ \hline x_{k+1} \\ \vdots \\ x_n \end{pmatrix} \quad \text{and} \quad u_f = A'_{\$} \begin{pmatrix} x_1 \\ \vdots \\ x_k \\ \hline x_{k+1} \\ \vdots \\ x_n \\ \hline 0 \\ \vdots \\ 0 \end{pmatrix} = \begin{pmatrix} \lambda x_1 \\ \vdots \\ \lambda x_k \\ \hline x_{k+1} \\ \vdots \\ x_n \\ \hline (1-\lambda)x_1 \\ \vdots \\ (1-\lambda)x_k \end{pmatrix} = \begin{pmatrix} u_a \\ \hline u_{r_1} \\ \hline u_{r_2} \end{pmatrix}.$$

If $f_{M_1}(w) = 1$, u_{r_1} is a zero vector, and so

$$f_{M_2}(w) = \frac{\lambda\left(|x_1| + \cdots + |x_k|\right)}{|x_1| + \cdots + |x_k|} = \lambda.$$

If $f_{M_1}(w) < 1$, then u_{r_1} is a non-zero vector, and so

$$f_{M_2}(w) = \frac{\lambda\left(|x_1| + \cdots + |x_k|\right)}{|x_{k+1}| + \cdots + |x_n| + |x_1| + \cdots + |x_k|} < \lambda,$$

where the inequality is trivial. □

4 Exclusive affine languages

We have the following equality for PFAs and QFAs:

$$\mathsf{SL} = \mathsf{QAL} \text{ and } \mathsf{SL}^{\neq} = \mathsf{QAL}^{\neq}.$$

But, QFAs are more powerful then PFAs when the cutpoint is fixed to 0 (nondeterministic recognition mode):

$$\mathsf{REG} \subsetneq \mathsf{NQAL}.$$

On the other hand, by definition we have $\mathsf{NQAL} \subseteq \mathsf{QAL}^{\neq}$, but interestingly this inclusion is not proper as they are identical:

$$\mathsf{NQAL} = \mathsf{QAL}^{\neq} \ (= \mathsf{SL}^{\neq}).$$

AfAs are more powerful than PFAs and QFAs in the case of language recognition with cutpoints. On the other hand, AfAs and QFAs have the same computational power

in nondeterministic language recognition mode (Díaz-Caro and Yakaryılmaz (2016a)). By definition we have $\mathsf{NAfL} \subseteq \mathsf{AfL}^{\neq}$, and the question is whether we can replace the relation "$\subseteq$" with "$\subsetneq$" similar to PFAs or with "$=$" similar to QFAs. We show that AfAs are similar to PFAs in this case.

We work with the complement classes: $\mathsf{SL}^{=}$ and $\mathsf{AfL}^{=}$. We introduce a new language ABS-EQ defined on $\{a, b\}$ such that $w \in$ ABS-EQ if and only if

$$|m - n| + |m - 4n| = |m - 2n| + |m - 3n|, \tag{1}$$

where $|w|_a = m$ and $|w|_b = n$. It is trivial that ABS-EQ contains every string if we do not use the absolute value signs in Equation 1. Therefore, the absolute signs makes language ABS-EQ interesting, which turns out to be an evidence about the power of the weighting operator.

Lemma 1. *Equation 1 is satisfied if and only if $m \geq 4n$ or $m \leq n$.*

Proof. Both m and n are non-negative integers. We check each case one by one.

If $m \geq 4n$, the left side is $m - n + m - 4n = 2m - 5n$ and the right side is $m - 2n + m - 3n = 2m - 5n$, and so the equation is satisfied. If $m \leq n$, then the left side is $n - m + 4n - m = 5n - 2m$ and the right side is $2n - m + 3n - m = 5n - 2m$, and so the equation is satisfied.

If $m \in (n, 4n)$, the left side is $m - n + 4n - m = 3n$. We have three sub-cases for the right side:

- When $n < m \leq 2n$, the right side is $2n - m + 3n - m = 5n - 2m = 3n + 2(n - m) < 3n$. So, the equation is not satisfied.
- When $2n < m \leq 3n$, the right side is $m - 2n + 3n - m = n < 3n$. So, the equation is not satisfied.
- When $3n < m < 4n$, the right side is $m - 2n + m - 3n = 2m - 5n < 2(4n) - 5n = 3n$. So, the equation is not satisfied.

□

We show that ABS-EQ is not in $\mathsf{SL}^{=}$ by using the following fact due to Diêu (1971).

Fact 1 *Let L be a language in $\mathsf{SL}^{=}$. Hence, there exists an n-state PFA P such that $w \in L$ if and only if $f_P(w) = \frac{1}{2}$. Then, for any $x, y, z \in \Sigma^*$,*

$$\text{if } xz, xyz, xy^2z, \ldots, xy^{n-1}z \in L, \text{ we also have } xy^*z \in L.$$

Theorem 4. ABS-EQ $\notin \mathsf{SL}^{=}$.

Proof. Suppose that there exists n-state PFA P as described in the above fact for the language ABS-EQ for $n > 1$. (When $n = 1$, P either accepts all strings or accepts none.) We pick $x = a^{8n}b$, $z = b^n$, and $y = b$. Then, we have the following list (due to

Lemma 1, as long as $|w|_a - 4|w|_b \geq 0$, Equation 1 is satisfied):

$$\begin{array}{llll}
w = xy & = a^{8n}b^{n+1} & \text{is in ABS-EQ since } & |w|_a - 4|w|_b = 4n-4 \geq 0 \\
w = xyz & = a^{8n}b^{n+2} & \text{is in ABS-EQ since } & |w|_a - 4|w|_b = 4n-8 \geq 0 \\
\vdots & \vdots & \vdots & \\
w = xy^jz & = a^{8n}b^{n+j+1} & \text{is in ABS-EQ since } & |w|_a - 4|w|_b = 4n-4j-4 \geq 0 \\
\vdots & \vdots & \vdots & \\
w = xy^{n-1}z & = a^{8n}b^{2n} & \text{is in ABS-EQ since } & |w|_a - 4|w|_b = 0 \geq 0
\end{array}$$

Due to the above fact, $xy^nz = a^{8n}b^{2n+1}$ must be in ABS-EQ but Equation 1 cannot be satisfied for xy^nz:

$$\begin{aligned}
|8n-(2n+1)| + |8n-4(2n+1)| &\stackrel{?}{=} |8n-2(2n+1)| + |8n-3(2n+1)| \\
(6n-1)+4 &\stackrel{?}{=} (4n-2)+(2n-3) \\
6n+3 &\neq 6n-5
\end{aligned}$$

Thus, xy^nz is not in ABS-EQ, which leads us to a contradiction. Therefore, ABS-EQ is not in $\mathsf{SL}^{=}$. □

Now, we present our AfA algorithm for ABS-EQ which only calculates the values inside the absolute values in Equation 1 and the desired decision is followed by the weighting operator.

Theorem 5. ABS-EQ *is in* $\mathsf{AfL}^{=}$.

Proof. We design a 6-state AfA M for ABS-EQ. The initial state is $(1\ 0\ 0\ 0\ 0\ 0)^T$. The operator for $\not\in$ is the identity matrix. Let $w \in \{a,b\}^*$ be the given input and $|w|_a = m$ and $|w|_b = n$. After reading w, the values of m and n are stored in the second and third states:

$$v_{|\not\in w|} = \begin{pmatrix} 1-m-n \\ m \\ n \\ 0 \\ 0 \\ 0 \end{pmatrix},$$

where the updates for three states for symbols a and b are as follows. The value of the second (first) entry is increased (decreased) by 1 when reading an a:

$$\begin{pmatrix} -m'-n' \\ m'+1 \\ n' \end{pmatrix} = \begin{pmatrix} 0 & -1 & -1 \\ 1 & 2 & 1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1-m'-n' \\ m' \\ n' \end{pmatrix}.$$

The value of the third (first) entry is increased (decreased) by 1 when reading a b:

$$\begin{pmatrix} -m'-n' \\ m' \\ n'+1 \end{pmatrix} = \begin{pmatrix} 0 & -1 & -1 \\ 0 & 1 & 0 \\ 1 & 1 & 2 \end{pmatrix} \begin{pmatrix} 1-m'-n' \\ m' \\ n' \end{pmatrix}.$$

On the right end-marker, we have the following operation:

$$v_f = \begin{pmatrix} m-n \\ m-2n \\ m-3n \\ m-4n \\ \frac{1-T}{2} \\ \frac{1-T}{2} \end{pmatrix} = \begin{pmatrix} 0 & 1 & -1 & 1 & 0 & 0 \\ 0 & 1 & -2 & 0 & 1 & 0 \\ 0 & 1 & -3 & 0 & 0 & 1 \\ 0 & 1 & -4 & 0 & 0 & 0 \\ \frac{1}{2} & -\frac{3}{2} & \frac{11}{2} & 0 & 0 & 0 \\ \frac{1}{2} & -\frac{3}{2} & \frac{11}{2} & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1-m-n \\ m \\ n \\ 0 \\ 0 \\ 0 \end{pmatrix},$$

where $T = 4m - 10n$, the summation of the first four entries in v_f. Let the first, fourth, and fifth states be the accepting ones. Then $f_M(w) = \frac{1}{2}$ if and only if

$$|m-n| + |m-4n| + \left|\frac{1-T}{2}\right| = |m-2n| + |m-3n| + \left|\frac{1-T}{2}\right|,$$

which is

$$|m-n| + |m-4n| = |m-2n| + |m-3n|.$$

Therefore, ABS-EQ is a member of $\mathsf{AfL}^{=}$. □

Corollary 1. $\mathsf{SL}^{=} = \overline{\mathsf{NQAL}} = \overline{\mathsf{NAfL}} \subsetneq \mathsf{AfL}^{=}$ *and* $\mathsf{SL}^{\neq} = \mathsf{NQAL} = \mathsf{NAfL} \subsetneq \mathsf{AfL}^{\neq}$.

Thus, similar to PFAs, if we change the cutpoint in $(0, 1)$, the class $\mathsf{AfL}^{\neq}$ does not change, but when setting the cutpoint to 0 or 1, we obtain NAfL, a proper subset of $\mathsf{AfL}^{\neq}$.

The anonymous reviewer pointed out the status of the following languages: For a given set of integers $I = \{a_1, \ldots, a_4, b_1, \ldots, b_4\}$, $L_I \subseteq \{a, b\}^*$ is the language composed by the strings satisfying the following linear equation:

$$|a_1 m + b_1 n| + |a_2 m + b_2 n| = |a_3 m + b_3 n| + |a_4 m + b_4 n|,$$

where m and n are respectively the numbers of as and bs in the strings. Even though it is easy to design an AfA for each such language, it is not trivial to derive some impossibility proofs – at least we do not see any easy generalization. It is clear that for some sets I, L_I can be trivial regular languages such as the empty language. However, for some other cases, the analysis of the above equation can be very complicated. We left open to classify these languages (whether in $\mathsf{SL}^{=}$ or not) as a future work.

Acknowledgment

We thanks the anonymous reviewer for the useful comments. Yakaryılmaz was partially supported by CAPES with grant 88881.030338/2013-01, the ERDF project Nr. 1.1.1.5/19/A/005 "Quantum computers with constant memory", and the project "Quantum algorithms: from complexity theory to experiment" funded under ERDF programme 1.1.1.5.

References

Ambainis, A., Yakaryılmaz, A. (2021). Automata and quantum computing, *in* Pin, J. (ed.), *Handbook of Automata Theory*, European Mathematical Society Publishing House, Zürich, Switzerland, pp. 1457–1493.

Belovs, A., Montoya, J. A., Yakaryılmaz, A. (2017). On a conjecture by Christian Choffrut, *Int. J. Found. Comput. Sci.* **28**(5), 483–502.

Díaz-Caro, A., Yakaryılmaz, A. (2016a). Affine computation and affine automaton, *Computer Science — Theory and Applications*, Vol. 9691 of *LNCS*, Springer, pp. 1–15.

Díaz-Caro, A., Yakaryılmaz, A. (2016b). Affine computation and affine automaton, *Technical Report 1602.04732*, arXiv.

Diêu, P. D. (1971). On a class of stochasic languages, *Mathematical Logic Quarterly* **17**(1), 421–425.

Hirvensalo, M., Moutot, E., Yakaryılmaz, A. (2017). On the computational power of affine automata, *Language and Automata Theory and Applications*, Vol. 10168 of *LNCS*, Springer, pp. 405–417.

Hirvensalo, M., Moutot, E., Yakaryılmaz, A. (2019). Computational limitations of affine automata, *Unconventional Computation and Natural Computation*, Vol. 11493 of *LNCS*, pp. 108–121.

Hirvensalo, M., Moutot, E., Yakaryılmaz, A. (2021). Computational limitations of affine automata and generalized affine automata, *Nat. Comput.* **20**(2), 259–270.

Ibrahimov, R., Khadiev, K., Prūsis, K., Yakaryılmaz, A. (2018). Error-free affine, unitary, and probabilistic OBDDs, *Descriptional Complexity of Formal Systems*, Vol. 10952 of *LNCS*, Springer, pp. 175–187. arXiv:1703.07184.

Ibrahimov, R., Khadiev, K., Prūsis, K., Yakaryılmaz, A. (2021). Error-free affine, unitary, and probabilistic OBDDs, *International Journal of Foundations of Computer Science* **32**(7), 827–847.

Jeandel, E. (2007). Topological automata, *Theory of Computing Systems* **40**(4), 397–407.

Khadieva, A., Yakaryılmaz, A. (2021). Affine automata verifiers, *in* Kostitsyna, I., Orponen, P. (eds), *Unconventional Computation and Natural Computation*, Vol. 12984 of *LNCS*, Springer, pp. 84–100.

Nakanishi, M., Khadiev, K., Prūsis, K., Vihrovs, J., Yakaryılmaz, A. (2017). Exact affine counter automata, *15th International Conference on Automata and Formal Languages*, Vol. 252 of *EPTCS*, pp. 205–218. arXiv:1703.04281.

Nakanishi, M., Khadiev, K., Prūsis, K., Vihrovs, J., Yakaryılmaz, A. (2022). Exact affine counter automata, *Int. J. Found. Comput. Sci.* **33**(3&4), 349–370.

Nielsen, M. A., Chuang, I. L. (2000). *Quantum Computation and Quantum Information*, Cambridge University Press.

Paz, A. (1971). *Introduction to Probabilistic Automata*, Academic Press, New York.

Rabin, M. O. (1963). Probabilistic automata, *Information and Control* **6**, 230–243.

Rabin, M., Scott, D. (1959). Finite automata and their decision problems, *IBM Journal of Research and Development* **3**, 114–125.

Say, A. C. C., Yakaryılmaz, A. (2014). Quantum finite automata: A modern introduction, *Computing with New Resources*, Springer, pp. 208–222.

Sipser, M. (2013). *Introduction to the Theory of Computation, 3rd edition*, Cengage Learning, United States of America.

Turakainen, P. (1969). Generalized automata and stochastic languages, *Proceedings of the American Mathematical Society* **21**, 303–309.

Villagra, M., Yakaryılmaz, A. (2016). Language recognition power and succintness of affine automata, *Unconventional Computation and Natural Computation*, Vol. 9726 of *LNCS*, Springer, pp. 116–129.

Villagra, M., Yakaryılmaz, A. (2018). Language recognition power and succinctness of affine automata, *Natural Computing* **17**(2), 283–293.

Yakaryılmaz, A. (2021). Improved constructions for succinct affine automata, *in* Han, Y., Ko, S. (eds), *Descriptional Complexity of Formal Systems*, Vol. 13037 of *LNCS*, Springer, pp. 188–199.

Yakaryılmaz, A., Say, A. C. C. (2010). Languages recognized by nondeterministic quantum finite automata, *Quantum Information & Computation* **10**(9&10), 747–770.

Yakaryılmaz, A., Say, A. C. C. (2011). Unbounded-error quantum computation with small space bounds, *Information and Computation* **279**(6), 873–892.

Received August 26, 2021 , revised June 8, 2022, accepted August 20, 2022