Multi-Letter Quantum Finite Automata: Decidability of the Equivalence and Minimization of States

D. Qiu[1,2,3] and L. Li[1] and X. Zou[1] and P. Mateus[2] and J. Gruska[4]

[1]Department of Computer Science,
Sun Yat-sen University, Guangzhou, 510006, China

[2]SQIG–Instituto de Telecomunicações,
Departamento de Matemática, Instituto Superior Técnico,
TULisbon, Av. Rovisco Pais 1049-001, Lisbon, Portugal

[3]The State Key Laboratory of Computer Science,
Institute of Software, Chinese Academy of Sciences, Beijing 100080, China

[4]Faculty of Informatics, Masaryk University, Brno, Czech Republik

February 2011

Abstract

Multi-letter *quantum finite automata* (QFAs) are quantum variants of the *one-way multi-head finite automata* (J. Hromkovič, Acta Informatica 19 (1983) 377-384). It has been shown that this new one-way QFAs (multi-letter QFAs) can accept with no error some regular languages, for example $(a + b)^*b$, that are not acceptable by QFAs of Moore and Crutchfield [20] as well as Kondacs and Watrous [16]. In this paper, we study the decidability of the equivalence and minimization problems of multi-letter QFAs. Two new results presented in this paper are the following ones: (1) Given a k_1-letter QFA $\mathcal{A}_1$ and a k_2-letter QFA $\mathcal{A}_2$ over the same input alphabet Σ, they are equivalent if and only if they are $(n^2m^{k-1}-m^{k-1}+k)$-equivalent, where $m = |\Sigma|$ is the cardinality of Σ, $k = \max(k_1, k_2)$, and $n = n_1+n_2$, with n_1 and n_2 being numbers of states of $\mathcal{A}_1$ and $\mathcal{A}_2$, respectively. When $k = 1$, this result implies the decidability of equivalence of measure-once QFAs [20]. (It is worth mentioning that our technical method is essentially different from the previous ones used in the literature.) (2) A polynomial-time $O(m^{2k-1}n^8 + km^kn^6)$ algorithm is designed to determine the equivalence of any two multi-letter QFAs (see Theorems 2 and 3). Observe also that time complexity is expressed here in terms of the number of states of the multi-letter QFAs and k can be seen as a constant. (3) It is shown that the states minimization problem of multi-letter QFAs is decidable in EXPSPACE.

1 Introduction

Models of automata used to play a very important role in the (classical) theoretical computer science. They were very helpful in finding power and limitations of various computation features and in developing a very powerful theory of computational complexity. Models of finite automata, as the key model behind the intuitively very important idea of the real-time and finite memory computations, helped also to develop various design techniques for classical processors. All that is already a sufficient reason to assume that quantum versions of them can be expected to play an important role in the theory of quantum information processing. In addition, a vision of the most simple quantum processor as the one performing quantum actions on classical inputs leads also naturally to the models of quantum finite automata. The final reason for assuming that quantum finite automata could be of large importance is the fact that quantum memory seems to be very expensive and it is therefore of paramount importance to know what can be achieved with limited amounts of quantum resources.

Interest in quantum computation and information has steadily increased since Shor's quantum algorithm for factoring integers in polynomial time [30] and Grover's algorithm for searching in unsorted database of size n with only $O(\sqrt{n})$ accesses [9]. These algorithms are nice, fast, but require for a realization of very powerful quantum computers and that seems to be still a long term goal. It is therefore clear that there is a need to study "small-size" quantum processors using variations of the models that have shown their relevance in the classical cases [10].

There is a variety of models of quantum finite automata (QFAs) (for instance, [20, 16, 2, 5, 4, 1, 11]). All of them can be seen as models of quantum processors in a similar way as various models of classical finite automata serve as models of classical processors with finite memory. The first two main models of QFAs were introduced independently by Moore and Crutchfield [20], as well as by Kondacs and Watrous [16], using different measurement modes, and they were intensively investigated also by others.

There are several ways how to categorize models of QFAs. Two main ones are according to the "head moves" and quantum resources and tools they operate with. According to the head moves, the main models are those that heads always move and always in the same direction (so-called one-way QFA). As natural modifications of that are models that heads can move in both directions (so called two-way QFA), or in one-direction only but sometimes they may be stationary (so-called 1.5 way QFA introduced by Amano and Iwama [1])–in the last two cases, however, models are not really with finite memory.

On the other hand, by considering the number of times the measurement is performed during a computation, 1QFAs have two different forms: *measure-once* 1QFAs (MO-1QFAs) proposed by Moore and Crutchfield [20], and, *measure-many* 1QFAs (MM-1QFAs) studied first by Kondacs and Watrous [16]. MM-1QFAs are strictly more powerful than MO-1QFAs [2, 4] (the language a^*b^* can be accepted by MM-1QFAs with bounded error but not by any MO-1QFA with bounded error). Due to the unitarity of quantum physics and finite memory of finite automata, both MO-1QFAs and MM-1QFAs can accept, with respect

to the so-called bounded error acceptance, only proper subclasses of regular languages with bounded error (see, e.g., [16, 2, 5, 4, 15]). Indeed, it was shown that the regular language $(a+b)^*b$ cannot be accepted by any MM-1QFA with bounded error [16][1].

Belovs, Rosmanis, and Smotrovs [3] proposed a new model of one-way QFAs. Namely, so-called multi-letter QFAs, that should be thought of as a quantum counterpart of *one-way multi-head finite automata* [12, 14]. Roughly speaking, a k-letter QFA is not limited to see only one, the just-incoming input letter, but can see several–up to k–of the earlier received letters as well. Therefore, the quantum state transition at each step depends on the last k letters received. Otherwise such automata work as MO-1QFAs already discussed.

By $\mathcal{L}(QFA_k)$ we denote the class of languages accepted with bounded error by k-letter QFAs. Any given k-letter QFA can be simulated by some $(k+1)$-letter QFA. Qiu and Yu [25] have proved that $\mathcal{L}(QFA_k) \subsetneq \mathcal{L}(QFA_{k+1})$ for $k = 1, 2, ...$, where the inclusion $\subsetneq$ is proper. Therefore, $(k+1)$-letter QFAs are computationally more powerful than k-letter QFAs. Belovs et al. [3] have already showed that the language $(a+b)^*b$ can be accepted by a 2-letter QFA but, as proved in [16], it cannot be accepted by any MM-1QFA with bounded error. Therefore, multi-letter QFAs can accept some regular languages that cannot be accepted neither by MM-1QFA nor by MO-1QFA.

To determine the equivalence for automata is an important issue in the theory of computation (see, e.g., [21, 31]). Two computing models over the same input alphabet Σ are defined to be n-equivalent if and only if the accepting probabilities for all input strings of length not more than n are equal, while they are defined to be equivalent if and only if the accepting probabilities for all input strings are equal.

Concerning the problem of determining the equivalence for QFAs, the simplest case—MO-1QFAs—was dealt with in [5, 20]. For quantum sequential machines (QSMs), Qiu and Li [23, 17] gave a method for determining whether or not any two given QSMs are equivalent. This method can be used to determine to determining the equivalence between any two MO-1QFAs. For the equivalence problem of MM-1QFAs, inspired by [31] and [4], Li and Qiu [18] presented a polynomial-time algorithm for determining whether or not any two given MM-1QFAs are equivalent. Recently, Qiu and Yu [25] proved that a k_1-letter QFA $\mathcal{A}_1$ and a k_2-letter QFA $\mathcal{A}_2$ over one-letter input alphabet $\Sigma = \{\sigma\}$ are equivalent if and only if they are $(n_1+n_2)^4 + k - 1$-equivalent, where n_1 and n_2 are numbers of states of $\mathcal{A}_1$ and $\mathcal{A}_2$, respectively, and $k = \max(k_1, k_2)$. However, it seems that the decidability method from [25] cannot be applied to the equivalence problem of multi-input alphabet. Therefore, a new method is needed to determine the equivalence of multi-letter QFAs for an arbitrary input alphabet Σ. (This will be discussed in more details in Remark 3.9.)

In this paper, we study the equivalence of multi-letter QFAs for any input alphabet $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_m\}$. More specifically, we prove that given any k_1-letter QFA $\mathcal{A}_1$ and k_2-letter QFA $\mathcal{A}_2$ over the input alphabet $\Sigma =$

[1]The case of the so-called unbounded error acceptance was explored in depth by Yakaryilmaz and Cem Say [33, 34].

$\{\sigma_1, \sigma_2, \ldots, \sigma_m\}$, they are equivalent if and only if they are $((n_1+n_2)^2 m^{k-1} - m^{k-1} + k)$-equivalent, where n_1 and n_2 are numbers of states of $\mathcal{A}_1$ and $\mathcal{A}_2$, respectively, and $k = \max(k_1, k_2)$. As a corollary, when $k = 1$, we obtain that given any MO-1QFAs $\mathcal{A}_1$ and $\mathcal{A}_2$ over the input alphabet $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_m\}$, they are equivalent if and only if they are $(n_1+n_2)^2$-equivalent. In addition, when $m = 1$, we have that given any k_1-letter QFAs $\mathcal{A}_1$ and k_2-letter $\mathcal{A}_2$ over the input alphabet $\Sigma = \{\sigma\}$, they are equivalent if and only if they are $((n_1+n_2)^2 + k - 1)$-equivalent.

Observe that if we determine the equivalence of multi-letter QFAs by checking all strings x with $|x| \leq n^2 m^{k-1} - m^{k-1} + k$, then the time complexity is exponential, i.e., $O(n^6 m^{n^2 m^{k-1} - m^{k-1} + 2k - 1})$, where $|x|$ is the length of x, and $n = n_1 + n_2$. In this paper, we design a polynomial-time $O(m^{2k-1} n^8 + k m^k n^6)$ algorithm for determining the equivalence of any two multi-letter QFAs. Furthermore, we prove that the state minimization problem concerning multi-letter QFAs is decidable in EXPSPACE.

The rest of the paper is organized as follows. In Section 2, we recall the definition of multi-letter QFAs and other related definitions, and some related results are reviewed. In Section 3, we give a condition (mentioned before) for two multi-letter QFAs to be equivalent. Then, in Section 4, we design a polynomial-time $O(m^{2k-1} n^8 + k m^k n^6)$ algorithm for determining the equivalence of any two multi-letter QFAs. Afterwards, in Section 5, we prove that the state minimization problem for multi-letter QFAs is decidable in EXPSPACE. Finally, in Section 6, we address some related issues for further consideration.

The usual technique [20, 5, 17, 18, 19] to handle the equivalence problem for QFAs is to simulate QFAs by so-called bilinear machines [26, 21, 18, 19], and to make use of the method that has been developed to deal with the equivalence problem for these machines. However, it does not seem that this technique can be used also to deal with the equivalence problem of multi-letter QFAs, and therefore we have to develop a new technique to deal with the equivalence problem of multi-letter QFAs.

In general, symbols will be explained when they first appear. For a matrix A, we use A^* and $A^\dagger$ to denote its conjugate and conjugate transpose, respectively.

2 Preliminaries

According to a quantum mechanical principle, to each closed quantum system[2] $\mathcal{S}$ a Hilbert space $\mathcal{H}_\mathcal{S}$ is associated and states of $\mathcal{S}$ correspond to vectors of the norm one of $\mathcal{H}_\mathcal{S}$. In case $\mathcal{H}_\mathcal{S}$ is an n-dimensional vector space, then it has a basis consisting of n mutually orthogonal vectors and we will usually denote these vectors and the basis by

$$\{|i\rangle\}_{i=1}^n.$$

In such a case any vector of $\mathcal{H}_\mathcal{S}$ can be uniquely expressed as the superposition:

$$|\psi\rangle = \sum_{i=1}^{n} \alpha_i |i\rangle \tag{1}$$

[2]That is to a quantum system that has no interaction with an environment.

where α_i's are complex numbers, called probability amplitudes, that satisfy the so-called normalization condition $\sum_{i=1}^{n} |\alpha_i|^2 = 1$. If such a state is measured with respect to the above basis, then the state collapses to one of the states $|i\rangle$ and to a particular state $|i_0\rangle$ with the probability $|\alpha_{i_0}|^2$ and, if this is the case, then i_0 is the outcome we receive into the classical world.

Each evolution of a finite n dimensional quantum system is specified by a unitary $n \times n$ matrix U and in each (discrete) step such an evolution changes any state $|\phi\rangle$ into the state $U|\phi\rangle$.

To extract some information from a quantum state $|\psi\rangle$ a measurement has to be performed. We will consider here only projective measurements that are defined by a set $\{P_m\}$ of so-called projective operators/matrices, where indices m refer to the potential classical outcomes of measurements, with the property:

$$P_i P_j = \begin{cases} P_i & \text{if } i = j, \\ 0 & \text{if } i \neq j. \end{cases} \tag{2}$$

The operators/projectors should satisfy the following completeness condition:

$$\sum_{i=1}^{n} P_i = I. \tag{3}$$

In case a state $|\psi\rangle$ is measured with respect to such a set of projective operators $\{P_m\}$, then the classical outcome m is obtained with the probability

$$p(m) = \|P_m \psi\rangle\|^2, \tag{4}$$

and the state $|\psi\rangle$ "collapse" into the state

$$\frac{P_m \psi\rangle}{\sqrt{p(m)}}. \tag{5}$$

A projective measurement $\{P_m\}$ is usually specified by an *observable* M, a Hermitian matrix that has a so called spectral decomposition

$$M = \sum_{m} m P_m \tag{6}$$

where m are mutually different eigenvalues of M and each P_m is a projector into the space of eigenvectors associated to the eigenvalue m.

Now we briefly review some definitions and related properties that will be used in the sequel. For details, we can refer to [3, 25]. First we recall k-letter deterministic finite automata (k-letter DFAs).

Definition 2.1 [[3]] A k-letter *deterministic finite automaton* (k-letter DFA) is defined by a quintuple $(Q, Q_{acc}, q_0, \Sigma, \gamma)$, where Q is a finite set of states, $Q_{acc} \subseteq Q$ is the set of accepting states, $q_0 \in Q$ is the initial state, Σ is a finite input alphabet, and γ is a transition function that maps $Q \times T^k$ to Q, where $T = \{\Lambda\} \bigcup \Sigma$ and $\Lambda \notin \Sigma$ denotes the empty letter, and $T^k \subset T^*$ consists of all strings of length k.

We describe now the computing process of a k-letter DFA on an input string x in Σ^*, where $x = \sigma_1\sigma_2\cdots\sigma_n$. A k-letter DFA has a tape which contains the letter Λ in its first $k-1$ positions followed by the input string x. The automaton starts in the initial state q_0 and has k reading heads which initially are on the first k positions of the tape (clearly, the kth head reads σ_1 and the other heads read Λ). In each step, symbols read by k heads and the current internal state determine a new internal state and all reading heads move in parallel by one symbol to the right. This process ends when the last input symbol is read by a head. The input string x is accepted if and only if at the end of the computation the automaton enters an accepting state.

Clearly, k-letter DFAs are not more powerful than DFAs. Indeed, the family of languages accepted by k-letter DFA for any $k \geq 1$, is exactly that of regular languages.

A *group finite automaton* (GFA) [5] is a DFA whose state transition function, say δ, satisfies the condition that for any input symbol σ, $\delta(\cdot, \sigma)$ is a one-to-one map on the state set, i.e., a permutation on the state set.

Definition 2.2 [[3]] A k-letter DFA $(Q, Q_{acc}, q_0, \Sigma, \gamma)$ is called a k-letter group finite automaton (k-letter GFA) if and only if for any string $x \in T^k$ the function $\gamma_x(q) = \gamma(q, x)$ is a bijection from Q to Q.

Now we recall the definition of multi-letter QFAs [3].

Definition 2.3 [[3]] A k-letter QFA $\mathcal{A}$ is defined by a quintuple

$$\mathcal{A} = (Q, Q_{acc}, |\psi_0\rangle, \Sigma, \mu)$$

where Q is a set of states, $Q_{acc} \subseteq Q$ is the set of accepting states, $|\psi_0\rangle$ is the initial quantum state from $\mathcal{H}_Q$, Σ is a finite input alphabet, and μ is a function that assigns a unitary transition matrix U_w on $\mathbb{C}^{|Q|}$ to each string $w \in (\{\Lambda\} \cup \Sigma)^k$.

A k-letter QFA $\mathcal{A}$ works in the same way as an MO-1QFA [20, 5], except that it applies unitary transformations corresponding not only to the last letter but to the last k letters received (like a k-letter DFA). When $k = 1$, it is exactly an MO-1QFA. According to [3], all languages accepted by k-letter QFAs with bounded error are regular languages for any k.

Let us now determine the probability $P_{\mathcal{A}}(x)$ that a k-letter QFA $\mathcal{A} = (Q, Q_{acc}, |\psi_0\rangle, \Sigma, \mu)$ accepts an input string $x = \sigma_1\sigma_2\cdots\sigma_m$. From the definition of k-letter QFA it follows that for any $w \in (\{\Lambda\} \cup \Sigma)^k$, $\mu(w)$ is a unitary matrix. The definition of the mapping μ can be used to assign, in a usual way, a unitary matrix to any string $x = \sigma_1\sigma_2\cdots\sigma_m \in \Sigma^*$. By $\overline{\mu}$ (induced by μ) we will denote such a map from Σ^* to the set of all $|Q| \times |Q|$ unitary matrices. More specifically, $\overline{\mu}$ is induced by μ in the following way: For $x = \sigma_1\sigma_2\cdots\sigma_m \in \Sigma^*$,

$$\overline{\mu}(x) = \begin{cases} \mu(\Lambda^{k-1}\sigma_1)\mu(\Lambda^{k-2}\sigma_1\sigma_2)\cdots\mu(\Lambda^{k-m}x), & \text{if } m < k, \\ \mu(\Lambda^{k-1}\sigma_1)\mu(\Lambda^{k-2}\sigma_1\sigma_2)\cdots\mu(\sigma_{m-k+1}\sigma_{m-k+2}\cdots\sigma_m), & \text{if } m \geq k, \end{cases} \tag{7}$$

which specifies the computing process of $\mathcal{A}$ for an input string x.

Let P_{acc} denote the projection operator on the subspace spanned by Q_{acc}. Using the above notations it holds:

$$P_{\mathcal{A}}(x) = \|\langle\psi_0|\overline{\mu}(x)P_{acc}\|^2. \tag{8}$$

3 Equivalence problem for multi-letter quantum finite automata

The equivalence problem for multi-letter QFAs is defined as follows:

Definition 3.1 A k_1-letter QFA $\mathcal{A}_1$ and a k_2-letter QFA $\mathcal{A}_2$ over an input alphabet Σ are said to be equivalent (resp. t-equivalent) if $P_{\mathcal{A}_1}(w) = P_{\mathcal{A}_2}(w)$ for any $w \in \Sigma^*$ (resp. for any w with $|w| \leq t$).

Let $\mathbb{C}^n$ denote the Euclidean space consisting of all n-dimensional complex vectors. For a subset S of $\mathbb{C}^n$, span$S$ will denote the subspace spanned by S. For any given k-letter QFA $\mathcal{A} = (Q, Q_{acc}, |\psi_0\rangle, \Sigma, \mu)$ and any integer j we denote $\mathbb{F}(j) = \text{span}\{\langle\psi_0|\overline{\mu}(x) : x \in \Sigma^*, |x| \leq j\}$, $j = 1, 2, \cdots$. That is, $\mathbb{F}(j)$ is a subspace of $\mathbb{C}^{|Q|}$, spanned by $\{\langle\psi_0|\overline{\mu}(x) : x \in \Sigma^*, |x| \leq j\}$, where $|Q|$ denotes the the number of states of Q.

Lemma 3.2 *Let $\mathcal{A} = (Q, Q_{acc}, |\psi_0\rangle, \Sigma, \mu)$ be a k-letter QFA, where $\Sigma = \{\sigma_i : i = 1, 2, \cdots, m\}$. Then there exists an integer $i_0 \leq (n-1)m^{k-1} + k$ such that, for any $i \geq i_0$, $\mathbb{F}(i) = \mathbb{F}(i_0)$, where n is number of states of Q.*

Proof. Let us denote

$$\Sigma^{(k-1)} = \{x : x \in \Sigma^*, |x| = k-1\}, \tag{9}$$

and

$$\mathbb{G}(l, w) = \text{span}\{\langle\psi_0|\overline{\mu}(xw) : x \in \Sigma^*, |x| \leq l\} \tag{10}$$

for any $w \in \Sigma^{(k-1)}$ and any integer $l \in \{0, 1, 2, \cdots\}$. In addition, let us denote

$$\mathbb{H}(l) = \oplus_{w\in\Sigma^{(k-1)}}\mathbb{G}(l, w) \tag{11}$$

for any $l \in \{0, 1, 2, \cdots\}$, where $\oplus_{w\in\Sigma^{(k-1)}}\mathbb{G}(l, w)$ is the direct sum of $\mathbb{G}(l, w)$ for all $w \in \Sigma^{(k-1)}$. It is clear that

$$\mathbb{G}(l, w) \subseteq \mathbb{G}(l+1, w), \text{ for any } w \in \Sigma^{(k-1)} \text{ and any integer } l \in \{0, 1, 2, \cdots\}, \tag{12}$$

and therefore

$$\mathbb{H}(l) \subseteq \mathbb{H}(l+1), \text{ for all integers } l \in \{0, 1, 2, \cdots\}. \tag{13}$$

Since $\mathbb{G}(l, w)$ is a subspace of $\mathbb{C}^n$, we obtain that

$$1 \leq \dim(\mathbb{G}(l, w)) \leq n, \tag{14}$$

for any $w \in \Sigma^{(k-1)}$ and any $l \in \{0, 1, 2, \cdots\}$, where $\dim(\mathbb{G}(l, w))$ denotes the dimension of $\mathbb{G}(l, w)$. Furthermore, by the definition of direct sum, we have

$$m^{k-1} \leq \dim(\mathbb{H}(l)) \leq nm^{k-1} \tag{15}$$

for any $l \in \{0, 1, 2, \cdots\}$. Therefore, by (13) there exists $l_0 \leq (n-1)m^{k-1} + 1$ such that

$$\mathbb{H}(l_0) = \mathbb{H}(l_0 + 1). \tag{16}$$

Equivalently,

$$\mathbb{G}(l_0, w) = \mathbb{G}(l_0 + 1, w), \text{ for any } w \in \Sigma^{(k-1)}. \tag{17}$$

Let $i_0 = l_0 + (k-1) \leq (n-1)m^{k-1} + k$. Now, we prove by induction that, for any $i \geq i_0$, $\mathbb{F}(i) = \mathbb{F}(i_0)$.

Base step. When $i = i_0$, it is clear that $\mathbb{F}(i) = \mathbb{F}(i_0)$.

Induction step. Suppose $\mathbb{F}(j) = \mathbb{F}(i_0)$, for some $j \geq i_0$. Our purpose is to prove that $\mathbb{F}(j+1) = \mathbb{F}(i_0)$. For any $w \in \Sigma^{(j+1)}$, we denote $w = \sigma_1\sigma_2\cdots\sigma_{l_0+1}\sigma_{l_0+2}\cdots\sigma_{i_0}\sigma_{i_0+1}\cdots\sigma_j\sigma_{j+1}$ and let $w_0 = \sigma_{l_0+2}\cdots\sigma_{i_0}\sigma_{i_0+1}$. Clearly, $w_0 \in \Sigma^{(k-1)}$, and $\langle\psi_0|\overline{\mu}(\sigma_1\sigma_2\cdots\sigma_{l_0+1}\sigma_{l_0+2}\cdots\sigma_{i_0}\sigma_{i_0+1}) = \langle\psi_0|\overline{\mu}(\sigma_1\sigma_2\cdots\sigma_{l_0+1}w_0) \in \mathbb{G}(l_0+1, w_0)$.

Since $\mathbb{H}(l_0) = \mathbb{H}(l_0+1)$, i.e., $\mathbb{G}(l_0, w) = \mathbb{G}(l_0+1, w)$ for any $w \in \Sigma^{(k-1)}$, we obtain that $\langle\psi_0|\overline{\mu}(\sigma_1\sigma_2\cdots\sigma_{l_0+1}w_0) \in \mathbb{G}(l_0, w_0)$. Therefore, $\langle\psi_0|\overline{\mu}(\sigma_1\sigma_2\cdots\sigma_{l_0+1}w_0)$ can be linearly represented by the vectors of $\mathbb{G}(l_0, w_0)$. As a result, there exist a finite index set Γ and $x_\gamma \in \{x : x \in \Sigma^*, \ |x| \leq l_0\}$ as well as complex numbers p_γ with $\gamma \in \Gamma$, such that

$$\langle\psi_0|\overline{\mu}(\sigma_1\sigma_2\cdots\sigma_{l_0+1}w_0) = \sum_{\gamma\in\Gamma} p_\gamma\langle\psi_0|\overline{\mu}(x_\gamma w_0). \tag{18}$$

Therefore, we have

$$\begin{aligned}
\langle\psi_0|\overline{\mu}(w) &= \langle\psi_0|\overline{\mu}(\sigma_1\sigma_2\cdots\sigma_{l_0+1}w_0\sigma_{i_0+2}\cdots\sigma_{j+1}) \\
&= \langle\psi_0|\overline{\mu}(\sigma_1\sigma_2\cdots\sigma_{l_0+1}w_0)\mu(w_0\sigma_{i_0+2})\cdots\mu(\sigma_{j-k+2}\cdots\sigma_{j+1}) \\
&= \sum_{\gamma\in\Gamma} p_\gamma\langle\psi_0|\overline{\mu}(x_\gamma w_0)\mu(w_0\sigma_{i_0+2})\cdots\mu(\sigma_{j-k+2}\cdots\sigma_{j+1}) \\
&= \sum_{\gamma\in\Gamma} p_\gamma\langle\psi_0|\overline{\mu}(x_\gamma\sigma_{l_0+2}\cdots\sigma_{i_0}\sigma_{i_0+1}\cdots\sigma_j\sigma_{j+1}) \in \mathbb{F}(j).
\end{aligned}$$

Consequently, $\langle\psi_0|\overline{\mu}(w) \in \mathbb{F}(j)$, and we get that $\mathbb{F}(j+1) \subseteq \mathbb{F}(j)$. On the other hand, $\mathbb{F}(j) \subseteq \mathbb{F}(j+1)$ always holds. Hence, we obtain that $\mathbb{F}(j+1) = \mathbb{F}(j) = \mathbb{F}(i_0)$. The proof is completed. □

With the same proof method as that for Lemma 3.2 we can obtain the following lemma.

Lemma 3.3 *For $\Sigma = \{\sigma_i : i = 1, 2, \cdots, m\}$ and a k-letter QFA*

$$\mathcal{A} = (Q, Q_{acc}, |\psi_0\rangle, \Sigma, \mu),$$

there exists an integer $i_0 \leq (n^2-1)m^{k-1}+k$ *such that, for any* $i \geq i_0$, $\mathbb{E}(i) = \mathbb{E}(i_0)$, *where* n *is number of states of* Q, $\mathbb{E}(j) = span\{\langle\psi_0| \otimes (\langle\psi_0|)^* \overline{\nu}(x) : x \in \Sigma^*, |x| \leq j\}$ *for* $j = 1, 2, \cdots$, *and* $\overline{\nu}(x) = \overline{\mu}(x) \otimes \overline{\mu}(x)^*$, *where* $*$ *denotes the conjugate operation.*

In order to deal with the equivalence problem of multi-letter QFAs, we need one more lemma for which we need the following notation:

For $\Sigma = \{\sigma_1, \sigma_2, \cdots, \sigma_m\}$, a k_1-letter QFA $\mathcal{A}_1 = (Q_1, Q_{acc}^{(1)}, |\psi_0^{(1)}\rangle, \Sigma, \mu_1)$ and a k_2-letter QFA $\mathcal{A}_2 = (Q_2, Q_{acc}^{(2)}, |\psi_0^{(2)}\rangle, \Sigma, \mu_2)$, let $P_{acc}^{(1)}$ and $P_{acc}^{(2)}$ denote the projection operators on the subspaces spanned by $Q_{acc}^{(1)}$ and $Q_{acc}^{(2)}$, respectively. For any string $x \in \Sigma^*$, we set $\overline{\mu}(x) = \overline{\mu}_1(x) \oplus \overline{\mu}_2(x)$ and $P_{acc} = P_{acc}^{(1)} \oplus P_{acc}^{(2)}$, $Q_{acc} = \{|\eta_1\rangle \oplus |\eta_2\rangle : |\eta_1\rangle \in Q_{acc}^{(1)} \text{ and } |\eta_2\rangle \in Q_{acc}^{(2)}\}$. In addition, we denote $|\eta_1\rangle = |\psi_0^{(1)}\rangle \oplus \mathbf{0_2}$ and $|\eta_2\rangle = \mathbf{0_1} \oplus |\psi_0^{(2)}\rangle$, where $\mathbf{0_1}$ and $\mathbf{0_2}$ represent zero column vector of dimensions n_1 and n_2, respectively.

Lemma 3.4 *The automata* $\mathcal{A}_1$ *and* $\mathcal{A}_2$ *defined above are equivalent if and only if*

$$(\langle\eta_1|(\langle\eta_1|)^* - \langle\eta_2|(\langle\eta_2|)^*)\overline{\nu}(x) \sum_{p_j \in Q_{acc}} (|p_j\rangle(|p_j\rangle)^*) = 0 \tag{19}$$

for all strings $x \in \Sigma^+$, *where* $\overline{\nu}(x) = \overline{\mu}(x) \otimes \overline{\mu}(x)^*$.

Proof. Denote, for any string $x \in \Sigma^*$,

$$P_{\eta_1}(x) = \|\langle\eta_1|\overline{\mu}(x)P_{acc}\|^2 \tag{20}$$

and

$$P_{\eta_2}(x) = \|\langle\eta_2|\overline{\mu}(x)P_{acc}\|^2. \tag{21}$$

It holds that

$$\begin{aligned} P_{\eta_1}(x) &= \|\langle\eta_1|\overline{\mu}(x)P_{acc}\|^2 \\ &= \langle\eta_1|\overline{\mu}(x)P_{acc}P_{acc}^{\dagger}\overline{\mu}(x)^{\dagger}|\eta_1\rangle \\ &= \langle\eta_1|\overline{\mu}(x)P_{acc}\overline{\mu}(x)^{\dagger}|\eta_1\rangle \\ &= \langle\psi_0^{(1)}|\overline{\mu}_1(x)P_{acc}^{(1)}\overline{\mu}_1(x)^{\dagger}|\psi_0^{(1)}\rangle \\ &= P_{\mathcal{A}_1}(x), \end{aligned} \tag{22}$$

and similarly,

$$P_{\eta_2}(x) = P_{\mathcal{A}_2}(x). \tag{23}$$

Therefore,

$$P_{\mathcal{A}_1}(x) = P_{\mathcal{A}_2}(x) \tag{24}$$

if and only if

$$P_{\eta_1}(x) = P_{\eta_2}(x) \tag{25}$$

for any string $x \in \Sigma^*$. On the other hand, we have

$$\begin{aligned} P_{\eta_1}(x) &= \|\langle\eta_1|\overline{\mu}(x)P_{acc}\|^2 \\ &= \sum_{p_j\in Q_{acc}} |\langle\eta_1|\overline{\mu}(x)|p_j\rangle|^2 \\ &= \sum_{p_j\in Q_{acc}} \langle\eta_1|\overline{\mu}(x)|p_j\rangle(\langle\eta_1|\overline{\mu}(x)|p_j\rangle)^* \\ &= \sum_{p_j\in Q_{acc}} \langle\eta_1|(\langle\eta_1|)^*(\overline{\mu}(x)\otimes(\overline{\mu}(x))^*)|p_j\rangle(|p_j\rangle)^* \\ &= \langle\eta_1|(\langle\eta_1|)^*(\overline{\mu}(x)\otimes(\overline{\mu}(x))^*)\sum_{p_j\in Q_{acc}} |p_j\rangle(|p_j\rangle)^*. \end{aligned} \tag{26}$$

Similarly,

$$P_{\eta_2}(x) = \langle\eta_2|(\langle\eta_2|)^*(\overline{\mu}(x)\otimes(\overline{\mu}(x))^*)\sum_{p_j\in Q_{acc}} |p_j\rangle(|p_j\rangle)^*. \tag{27}$$

Therefore, Eq. (25) holds if and only if

$$\begin{aligned} &\langle\eta_1|(\langle\eta_1|)^*(\overline{\mu}(x)\otimes(\overline{\mu}(x))^*)\sum_{p_j\in Q_{acc}} |p_j\rangle(|p_j\rangle)^* \\ = \;&\langle\eta_2|(\langle\eta_2|)^*(\overline{\mu}(x)\otimes(\overline{\mu}(x))^*)\sum_{p_j\in Q_{acc}} |p_j\rangle(|p_j\rangle)^* \end{aligned} \tag{28}$$

for any $x \in \Sigma^*$.

Denote again

$$\overline{\nu}(x) = \overline{\mu}(x)\otimes\overline{\mu}(x)^*. \tag{29}$$

Clearly, $\overline{\nu}(x)$ is an $n^2 \times n^2$ complex square matrix. Then, the equivalence between $\mathcal{A}_1$ and $\mathcal{A}_2$ depends on whether or not the following equation holds for any $x \in \Sigma^*$:

$$\langle\eta_1|(\langle\eta_1|)^*\overline{\nu}(x)\sum_{p_j\in Q_{acc}} |p_j\rangle(|p_j\rangle)^* = \langle\eta_2|(\langle\eta_2|)^*\overline{\nu}(x)\sum_{p_j\in Q_{acc}} |p_j\rangle(|p_j\rangle)^*, \tag{30}$$

i.e.,

$$(\langle\eta_1|(\langle\eta_1|)^* - \langle\eta_2|(\langle\eta_2|)^*)\overline{\nu}(x)\sum_{p_j\in Q_{acc}} |p_j\rangle(|p_j\rangle)^* = 0. \tag{31}$$

□

With the above lemmas we are ready to prove the main theorem.

Theorem 3.5 *The automata $\mathcal{A}_1$ and $\mathcal{A}_2$ defined above are equivalent if and only if they are $(n^2m^{k-1} - m^{k-1} + k)$-equivalent, where $k = \max(k_1, k_2)$ and $n = n_1 + n_2$, with n_i being number of states in Q_i, $i = 1, 2$.*

Proof. Denote $\mathbb{D}(j) = \text{span}\{(\langle\eta_1|(\langle\eta_1|)^* - \langle\eta_2|(\langle\eta_2|)^*)\overline{\nu}(x) : x \in \Sigma^*, |x| \leq j\}$ for $j = 1, 2, \cdots$. Here $\mathbb{D}(j)$ is a subspace of $\mathbb{C}^{n^2}$. It follows from Lemma 3.3 that there exists an $i_0 \leq (n^2-1)m^{k-1}+k$ such that

$$\mathbb{D}(i) = \mathbb{D}(i_0) \tag{32}$$

for all $i \geq i_0$. Eq. (32) implies that, for any $x \in \Sigma^*$ with $|x| > (n^2-1)m^{k-1}+k$, $(\langle\eta_1|(\langle\eta_1|)^* - \langle\eta_2|(\langle\eta_2|)^*)\overline{\nu}(x)$ can be linearly represented by some vectors in $\{(\langle\eta_1|(\langle\eta_1|)^* - \langle\eta_2|(\langle\eta_2|)^*)\overline{\nu}(y) : y \in \Sigma^* \text{ and } |y| \leq (n^2-1)m^{k-1}+k\}$.

Consequently, if Eq. (19) holds for all x with $|x| \leq (n^2-1)m^{k-1}+k$, then so does it for all x with $|x| > (n^2-1)m^{k-1}+k$. This completes the proof of this theorem. □

From Theorem 3.5 we can get a sufficient and necessary condition for the equivalence of MO-1QFAs ($k = 1$), formulated in the following corollary, which was also presented by Li and Qiu [19].

Corollary 3.6 *For $\Sigma = \{\sigma_1, \sigma_2, \cdots, \sigma_m\}$, an MO-1QFA*

$$\mathcal{A}_1 = (Q_1, Q_{acc}^{(1)}, |\psi_0^{(1)}\rangle, \Sigma, \mu_1)$$

and another MO-1QFA

$$\mathcal{A}_2 = (Q_2, Q_{acc}^{(2)}, |\psi_0^{(2)}\rangle, \Sigma, \mu_2)$$

are equivalent if and only if they are $(n_1+n_2)^2$-equivalent, where n_i is number of states of Q_i, $i = 1, 2$.

Remark 3.7 Let us analyze the complexity of the algorithm sketched in the proof of Theorem 3.5. Similarly as in [31], we assume that all inputs consist of complex numbers whose real and imaginary parts are rational numbers and that each arithmetic operation on rational numbers can be done in a constant time. If we denote $n = n_1 + n_2$, then we can check in time $O(in^4)$ whether or not Eq. (19) holds for $x \in \Sigma^*$ with $|x| = i$. Because the length of those x that need to be used to check in Eq. (19) is at most $(n^2-1)m^{k-1}+k$, the time complexity for checking whether the two multi-letter QFAs are equivalent is $O(n^4(m + 2m^2 + ... + ((n^2-1)m^{k-1}+k)m^{n^2m^{k-1}-m^{k-1}+k}))$, that is $O(n^6 m^{n^2m^{k-1}-m^{k-1}+2k-1})$.

From Theorem 3.5 we can also obtain a sufficient and necessary condition for the equivalence of multi-letter QFAs over the one letter input alphabet, and we describe it by the following corollary (see also Qiu and Yu [25]).

Corollary 3.8 *For $\Sigma = \{\sigma\}$, a k_1-letter QFA $\mathcal{A}_1 = (Q_1, Q_{acc}^{(1)}, |\psi_0^{(1)}\rangle, \Sigma, \mu_1)$ and a k_2-letter QFA $\mathcal{A}_2 = (Q_2, Q_{acc}^{(2)}, |\psi_0^{(2)}\rangle, \Sigma, \mu_2)$ are equivalent if and only if they are (n^2+k-1)-equivalent, where $k = \max(k_1, k_2)$ and $n = n_1 + n_2$, with n_i being the number of states of Q_i, $i = 1, 2$.*

Remark 3.9 An essential difference for determining the equivalence of k-letter QFAs between "multi-letter input alphabet" and "single-letter input alphabet" is in the following.

When a k-letter QFA $\mathcal{A} = (Q, Q_{acc}, |\psi_0\rangle, \Sigma, \mu)$ with multi-letter input alphabet (i.e., $|\Sigma| \geq 2$) is considered, we can not always obtain

$$\langle\psi_0|\overline{\mu}(x\sigma) = \sum_{\gamma\in\Gamma} c_\gamma \langle\psi_0|\overline{\mu}(y_\gamma\sigma) \tag{33}$$

from

$$\langle\psi_0|\overline{\mu}(x) = \sum_{\gamma\in\Gamma} c_\gamma \langle\psi_0|\overline{\mu}(y_\gamma) \tag{34}$$

where $x, y_\gamma \in \Sigma^*$, $\sigma \in \Sigma$, and c_γ are complex numbers. Furthermore, we can not obtain

$$\langle\psi_0|\overline{\mu}(xz) = \sum_{\gamma\in\Gamma} c_\gamma \langle\psi_0|\overline{\mu}(y_\gamma z) \tag{35}$$

from Eq. (34) where $z \in \Sigma^*$.

However, we can obtain Eq. (33) from Eq. (34) for one-letter input alphabet. In general, if Eq. (33) can be obtained from Eq. (34), we must let the strings of the last $k-1$ letters of $y_\gamma, \gamma \in \Gamma$, be the same as those of x. Therefore, we need to divide $\{x : x \in \Sigma^* \text{ and } |x| \geq k-1\}$ into m^{k-1} classes by the last $k-1$ letters in our paper.

4 Polynomial-time equivalence of multi-letter QFAs

According to our analysis in Remark 3.7, we need exponential time for checking whether or not two multi-letter QFAs are equivalent if Eq. (19) is checked for all strings $x \in \Sigma^*$ with $|x| \leq (n^2-1)m^{k-1}+k$. In this section, our goal is to design a polynomial-time algorithm for determining the equivalence between any two multi-letter QFAs.

We will use notation from Lemma 3.4 and Theorem 3.5 and its proof. In addition, let us denote $\langle\eta| = \langle\eta_1|(\langle\eta_1|)^* - \langle\eta_2|(\langle\eta_2|)^*$ and $|P_{acc}\rangle = \sum_{p_j\in Q_{acc}} |p_j\rangle(|p_j\rangle)^*$. With this notation, Eq. (19) is equivalent to

$$\langle\eta|\overline{\nu}(x)|P_{acc}\rangle = 0. \tag{36}$$

For $w \in \Sigma^{(k-1)}$, $\mathbb{G}(l, w) = span\{\langle\eta|\overline{\nu}(xw) : x \in \Sigma^*, |x| \leq l\}$. It follows from the proof of Lemma 1 that there exists $l_0 \leq (n^2-1)m^{k-1}+1$ such that, for any $w \in \Sigma^{(k-1)}$ and any $l \geq l_0$, $\mathbb{G}(l, w) = \mathbb{G}(l_0, w)$. Since $\mathbb{G}(l_0, w)$ is a subspace of $\mathbb{C}^{n^2}$, $\dim(\mathbb{G}(l_0, w)) \leq n^2$.

If we can find in polynomial time a basis, denoted by $\mathcal{B}(w)$, of the subspace $\mathbb{G}(l_0, w)$, for all $w \in \Sigma^{(k-1)}$, then any element in $\cup_{w\in\Sigma^{(k-1)}}\mathbb{G}(l_0, w)$ can be linearly represented by these elements in $\cup_{w\in\Sigma^{(k-1)}}\mathcal{B}(w)$. Therefore, to determine whether or not Eq. (19) holds, it suffices to check Eq. (19) for all elements in $\cup_{w\in\Sigma^{(k-1)}}\mathcal{B}(w) \bigcup \{\langle\eta|\overline{\nu}(x) : x \in \cup_{l=0}^{k-2}\Sigma^{(l)}\}$.

4.1 A polynomial-time algorithm to find a basis of $\mathbb{G}(l_0, w)$

At first, we give some useful definitions. For $\Sigma = \{\sigma_1, \sigma_2, \cdots, \sigma_m\}$, we define a *strict order* [28] " $<$ " on Σ^+ ($\Sigma^+ = \Sigma^* \setminus \{\epsilon\}$). For any $x_1, x_2 \in \Sigma^+$,

$$x_1 < x_2 \text{ iff } |x_1| < |x_2| \text{ or } (|x_1| = |x_2|, x_1 = y\sigma_i z_1, x_2 = y\sigma_j z_2 \text{ and } i < j), \quad (37)$$

where $\sigma_i, \sigma_j \in \Sigma$ and $y, z_1, z_2 \in \Sigma^*$. Based on the strict order $<$, we define a *partial order* [28] " $\leq$ " on Σ^+. For any $\forall x_1, x_2 \in \Sigma^+$,

$$x_1 \leq x_2 \text{ iff } x_1 < x_2 \text{ or } x_1 = x_2. \quad (38)$$

The partial order $\leq$ can be expanded to Σ^* if we set $\epsilon \leq x$ for all $x \in \Sigma^*$. Clearly, Σ^* is a well-ordered set [28], i.e., every nonempty subset of Σ^* contains a least element.

In the following we design a polynomial-time algorithm to search for $\mathcal{B}(w)$ for all $w \in \Sigma^{(k-1)}$, which is described in the proof of the following theorem.

Theorem 4.1 *For a k_1-letter QFA $\mathcal{A}_1 = (Q_1, Q_{acc}^{(1)}, |\psi_0^{(1)}\rangle, \Sigma, \mu_1)$ and a k_2-letter QFA $\mathcal{A}_2 = (Q_2, Q_{acc}^{(2)}, |\psi_0^{(2)}\rangle, \Sigma, \mu_2)$, where $\Sigma = \{\sigma_1, \sigma_2, \cdots, \sigma_m\}$, there exists a polynomial time $O(m^{2k-1}n^8 + km^k n^6)$ algorithm searching for $\bigcup_{w\in\Sigma^{(k-1)}} \mathcal{B}(w)$, where, as above, $\mathcal{B}(w)$ is a base of subspace $\mathbb{G}(l_0, w)$, $l_0 \leq (n^2-1)m^{k-1}+1$, and $n = n_1 + n_2$, with n_i being number of states of Q_i, $i = 1, 2$.*

Proof. Let $\Sigma_<^{(k)}$ be defined as $\Sigma^{(k)}$ but it is further required that the elements of $\Sigma^{(k)}$ are arranged according to the order $<$ from the smallest σ_1^k to the largest σ_m^k, where σ^k means the string $\underbrace{\sigma\sigma\cdots\sigma}_{k}$. In addition, let us denote $x\Sigma_< = \{x\sigma_1, x\sigma_2, \ldots, x\sigma_m\}$ for any $x \in \Sigma^*$.

We outline the process of searching for $\mathcal{B}(w)$, for all $w \in \Sigma^{(k-1)}$. Initially, we let $\langle\eta|\overline{\nu}(w) \in \mathcal{B}(w)$ for $w \in \Sigma^{(k-1)}$, and let us define *queue* to be $\Sigma_<^{(k)}$. Let us decompose any x from *queue* as $x = yw_i$ for some $y \in \Sigma^*$ and $w_i \in \Sigma^{(k-1)}$, and further check whether or not $\langle\eta|\overline{\nu}(x) \in span\mathcal{B}(w_i)$. If $\langle\eta|\overline{\nu}(x) \notin span\mathcal{B}(w_i)$, then add $\langle\eta|\overline{\nu}(x)$ to $\mathcal{B}(w_i)$ and add the elements of $x\Sigma_<$ to *queue* from small element to large one sequentially. By repeating the above process, we take another element x' from *queue* and then determine $x' = y'w_j$ for some $y' \in \Sigma^*$ and $w_j \in \Sigma^{(k-1)}$, and further check whether or not $\langle\eta|\overline{\nu}(x') \in span\mathcal{B}(w_j)$. Let us continue this process, until *queue* is empty. Since the number of the elements in $\bigcup_{w\in\Sigma^{(k-1)}} \mathcal{B}(w)$ is at most n^2m^{k-1}, the number of the elements in *queue* is at most n^2m^k. In addition, from the proof of Lemma 1, we know that $\mathbb{G}(l_0, w) = \mathbb{G}(l_0 + i, w)$ for any $i \geq 0$. Therefore, if x belongs to the *queue*, then $k - 1 \leq |x| \leq l_0 + k - 1 \leq (n^2-1)m^{k-1} + k$. Therefore, the above process will always terminate.

The above process is more formally described in Figure 1. Its time analysis will be given below.

We first prove that $\mathcal{B}(w)$ found out by the algorithm is exactly a base of $\mathbb{G}(l_0, w)$. Indeed, it follows form the algorithm that the vectors in $\mathcal{B}(w)$ are linearly independent. Therefore, we only need to prove that, for every

$w \in \Sigma^{(k-1)}$, all vectors in $\mathbb{G}(l_0, w)$ can be linearly expressed by vectors in $\mathcal{B}(w)$. To show that, it suffices to prove that, for any $x \in \Sigma^*$ with $x = x_0 w$, $\langle \eta | \overline{\nu}(x)$ can be linearly expressed by vectors in $\mathcal{B}(w)$. We proceed by induction on the partial order $\leq$ on Σ^*.

(1) *Basis.*

The case $x_0 = \varepsilon$ is handled by the first step of the algorithm. Therefore $\langle \eta | \overline{\nu}(x)$ can be linearly expressed by the vectors in $\mathcal{B}(w)$.

Figure 1. Algorithm (I) for searching for $\mathcal{B}(w)$ for all $w \in \Sigma^{(k-1)}$.

Input: $\mathcal{A}_1 = (Q_1, Q_{acc}^{(1)}, |\psi_0^{(1)}\rangle, \Sigma, \mu_1)$ and $\mathcal{A}_2 = (Q_2, Q_{acc}^{(2)}, |\psi_0^{(2)}\rangle, \Sigma, \mu_2)$

Set $\mathcal{B}(w) = \{\langle \eta | \overline{\nu}(w)\}$, for $w \in \Sigma^{(k-1)}$;
queue $\leftarrow \Sigma_<^{(k)}$;
while *queue* is not empty **do**
begin take an element x from *queue*;
  if $x = yw$ for some $y \in \Sigma^*$ and $\langle \eta | \overline{\nu}(x) \notin span\mathcal{B}(w)$
  then
  begin add $\langle \eta | \overline{\nu}(x)$ to $\mathcal{B}(w)$;
    add all elements of $x\Sigma_<$ to *queue* from the smallest $x\sigma_1$ to the largest $x\sigma_m$ sequentially;
  end;
end

(2) *Induction.*

Assume that for any $y > \epsilon$, whenever $x_0 < y$, $\langle \eta | \overline{\nu}(x_0 w)$ can be linearly expressed by the vectors in $\mathcal{B}(w)$. We prove that $\langle \eta | \overline{\nu}(yw)$ can be linearly expressed by the vectors in $\mathcal{B}(w)$ as well.

(i) If $x = yw$ is in the *queue*, then it follows directly from the algorithm that $\langle \eta | \overline{\nu}(x)$ can be linearly expressed by the vectors in $\mathcal{B}(w)$.

(ii) If $x = yw$ is not in the *queue*, then we know that there exist $w' \in \Sigma^{(k-1)}$ and $x_1, x_2 \in \Sigma^*$ with $|x_2| \geq 1$ such that

$$x = yw = x_1 w' x_2, \tag{39}$$

and

$$\langle \eta | \overline{\nu}(x_1 w') = \sum_{\gamma \in \Gamma} p_\gamma \langle \eta | \overline{\nu}(y_\gamma w'), \tag{40}$$

for some set of indices Γ, where $p_\gamma \in \mathbb{C}$, $y_\gamma \in \Sigma^*$, and $y_\gamma < x_1$ for all $\gamma \in \Gamma$. Therefore,

$$\langle \eta | \overline{\nu}(x) = \sum_{\gamma \in \Gamma} p_\gamma \langle \eta | \overline{\nu}(y_\gamma w' x_2) \tag{41}$$

and $y_\gamma w' x_2 < x_1 w' x_2 = x = yw$. Let $y_\gamma w' x_2 = y'_\gamma w$. Then $y'_\gamma < y$. Therefore, by the inductive assumption, we know that $\langle \eta | \overline{\nu}(y'_\gamma w)$ can be linearly expressed by vectors from $\mathcal{B}(w)$. Consequently, by Eq. (41) we obtain that $\langle \eta | \overline{\nu}(x)$ can be linearly expressed by vectors from $\mathcal{B}(w)$.

(3) *Conclusion.*

For any $\langle \eta | \overline{\nu}(x) \in \mathbb{G}(l_0, w)$, $\langle \eta | \overline{\nu}(x)$ can be linearly expressed by vectors from $\mathcal{B}(w)$. Hence, $\mathcal{B}(w)$ is a base of $\mathbb{G}(l_0, w)$ for every $w \in \Sigma^{(k-1)}$.

Complexity of the algorithm. Let us assume that all inputs consist of complex numbers whose real and imaginary parts are rational numbers and that each arithmetic operation on rational numbers can be done in a constant time. Note that in time $O(|x|n^4)$ we can compute $\langle\eta|\overline{\nu}(x)$. Let us recall that time $O(n^3)$ is needed to verify whether a set of n-dimensional vectors is linearly independent needs [8]. Time complexity to check whether or not $\langle\eta|\overline{\nu}(x) \in span\mathcal{B}(w)$ is therefore $O(n^6)$.

Because the basis $\mathcal{B}(w)$ has at most n^2 elements, $\bigcup_{w\in\Sigma^{(k-1)}}\mathcal{B}(w)$ has at most n^2m^{k-1} elements. Every element produces at most m valid child nodes. Therefore, we need to visit at most $O(n^2m^k)$ nodes in the *queue*. At every visited node x the algorithm does three things: (i) to calculate $\langle\eta|\overline{\nu}(x)$, what requires time $O(|x|n^4)$; (ii) to find the string w composed of the last $k-1$ letters of x, what requires time $O(k)$; (iii) to verify whether or not the n^2-dimensional vector $\langle\eta|\overline{\nu}(x)$ is linearly independent of the set $\mathcal{B}(w)$, what requires time $O(n^6)$ according to [8]. In addition, from the proof of Lemma 1, we know that $\mathbb{G}(l_0, w) = \mathbb{G}(l_0 + i, w)$ for any $i \geq 0$ and any $w \in \Sigma^{(k-1)}$. Therefore, if x belongs to the *queue*, then $k \leq |x| \leq l_0 + k - 1 \leq (n^2 - 1)m^{k-1} + k$.

Hence, the worst time complexity is $O(n^2m^k(n^6m^{k-1} + kn^4 + k + n^6))$, that is, $O(m^{2k-1}n^8 + km^kn^6)$. □

4.2 A polynomial-time algorithm for determining the equivalence between multi-letter QFAs

By virtue of Algorithm (I), we can determine the equivalence between any two multi-letter QFAs in polynomial time.

Theorem 4.2 *Let Σ, a k_1-letter QFA $\mathcal{A}_1$, and a k_2-letter QFA $\mathcal{A}_2$ be the same as in Theorem 2. Then there exists a polynomial-time $O(m^{2k-1}n^8 + km^kn^6)$ algorithm to determine whether or not $\mathcal{A}_1$ and $\mathcal{A}_2$ are equivalent.*

Proof. Denote $\mathcal{B}_0 = \{\langle\eta|\overline{\nu}(x) : x \in \bigcup_{l=0}^{k-2}\Sigma^{(l)}\}$. By Algorithm (I), in time $O(m^{2k-1}n^8 + km^kn^6)$ we can find $\bigcup_{w\in\Sigma^{(k-1)}}\mathcal{B}(w)$. Since $\mathcal{B}(w)$ is a basis of $\mathbb{G}(l_0, w)$, we know that every vector in $\{\langle\eta|\overline{\nu}(x) : x \in \Sigma^*, |x| \geq 1\}$ can be linearly expressed by a finite number of vectors in $\bigcup_{w\in\Sigma^{(k-1)}}\mathcal{B}(w)\bigcup\mathcal{B}_0$. Therefore, we have the following algorithm.

Figure 2. Algorithm (II) for determining the equivalence of multi-letter QFAs.

Input: $\mathcal{A}_1 = (Q_1, Q_{acc}^{(1)}, |\psi_0^{(1)}\rangle, \Sigma, \mu_1, k_1)$ and $\mathcal{A}_2 = (Q_2, Q_{acc}^{(2)}, |\psi_0^{(2)}\rangle, \Sigma, \mu_2, k_2)$.
Step 1:

Using Algorithm (I) find $\mathcal{B}(w)$ for all $w \in \Sigma^{(k-1)}$;
compute $\mathcal{B}_0$;

Step 2:

If $\forall\langle\psi| \in \bigcup_{w\in\Sigma^{(k-1)}}\mathcal{B}(w)\bigcup\mathcal{B}_0$, $\langle\psi|P_{acc}\rangle = 0$ **then** return ($\mathcal{A}_1$ and $\mathcal{A}_2$ are equivalent)
else return the x for which $\langle\eta|\overline{\nu}(x)|P_{acc}\rangle \neq 0$;

Complexity of the algorithm. To compute $\mathcal{B}_0$ time $O(m^{k-2}n^4)$ is needed. Therefore, by Algorithm (I), in Step 1, time $O(m^{2k-1}n^8 + km^k n^6)$ is needed.

Provided $\langle\eta|\overline{\nu}(x)$ has been computed, in time $O(n^2)$ we can decide whether $\langle\eta|\overline{\nu}(x)P_{acc} \neq 0$. Since the number of elements in $\mathcal{B}_0 \cup \bigcup_{w\in\Sigma^{(k-1)}} \mathcal{B}(w)$ is at most $O(m^{k-1}n^2)$, the worst time complexity in Step 2 is $O(m^{k-1}n^4)$.

In summary, the worst time complexity of Algorithm (II) is $O(m^{2k-1}n^8 + km^k n^6)$. □

5 Decidability of the state minimization problem of multi-letter QFAs

The important state minimization problem for DFA and PFA has already been investigated in depth [13, 26, 21] and solved in an elegant way for DFA. It is natural to try to explore this problem also for models of QFAs. Recently, Mateus and Qiu [22] proved that the state minimization of PFA and MO-1QFA is decidable, using Renegar's algorithm [27]. In this section, motivated by the idea from [22], we consider the state minimization problem of multi-letter QFAs, and show its solvability. Namely, that there exists an algorithm to output a minimal k-letter QFA equivalent to a given k-letter QFA $\mathcal{A}$. This result will be based, on one side, on the decidability of the equivalence problem discussed above and, on the other side, on the decidability of the theory of real ordered fields [6, 7, 27].

The decision problem for the existential theory of the reals [27] is the problem of deciding if the set $\mathbb{S} = \{x \in \mathbb{R}^n : \mathbf{P}(x)\}$ is non-empty, where $\mathbf{P}(x)$ is a predicate which is a Boolean function of atomic predicates either of the form $f_i(x) \geq 0$ or $f_j(x) > 0$, f's being real polynomials. For this decision problem, it is important to know three parameters: the number of atomic predicates m (i.e., the number of polynomials), the number of variables n, the highest degree d among all the atomic predicates forming $\mathbf{P}(x)$.

Canny [7] developed a PSPACE algorithm in n, m, d for the above problem, but its time complexity is very high. Later, Renegar [27] designed an algorithm of time complexity $(md)^{O(n)}$. Renegar's algorithm has asymptotically optimal time complexity. Furthermore, to find a sample of $\mathbb{S}$ requires $\tau d^{O(n)}$ space if all coefficients of the atomic predicates use at most τ space (see [6], page 518). Let us summarize these results in the following theorem.

Theorem 5.1 ([7, 27, 6]) *To decide whether the set $\mathbb{S} = \{x \in \mathbb{R}^n : \mathbf{P}(x)\}$ is non-empty, where $\mathbf{P}(x)$ is a predicate which is a Boolean function of atomic predicates either of the form $f_i(x) \geq 0$ or $f_j(x) > 0$, with f's being real polynomials, can be done in PSPACE in n, m, d, where n is the number of variables, m is the number of atomic predicates, d is the highest degree among all atomic predicates of $\mathbf{P}(x)$. Moreover, there exists an algorithm of time complexity $(md)^{O(n)}$ for this problem. To find a sample of $\mathbb{S}$ requires $\tau d^{O(n)}$ space if all coefficients of the atomic predicates use at most τ space.*

We will use the above theorem to deal with the state minimization of multi-letter QFAs. Since QFAs are usually defined over the field of complex numbers,

we need to transform the problem over the field of complex numbers to that over real numbers. That will be based on the following observation.

Remark 5.2 Any complex number $z = x+yi$ is determined by two reals x and y, and any complex polynomial $f(z)$ with $z \in \mathbb{C}^n$ can be equivalently written as $f(z) = f_1(x, y) + if_2(x, y)$ where $(x, y) \in \mathbb{R}^{2n}$ is the real representation of z, and f_1 and f_2 are real polynomials. Thus, the set $\mathbb{S}^{'}$ defined over the filed of complex numbers with n complex variables and m complex polynomials can be equivalently described by $\mathbb{S}$ over the field of real numbers with $2n$ real variables and $2m$ real polynomials.

Now we are in the position to deal with the minimization problem of multi-letter QFAs. We prove the following theorem.

Theorem 5.3 *The state minimization problem of multi-letter QFAs is decidable in EXPSPACE.*

Proof. Given a k-letter QFA $\mathcal{A} = (Q, Q_{acc}, |\psi_0\rangle, \Sigma, \mu)$, the goal is to find another k-letter QFA $\mathcal{A}' = (Q', Q'_{acc}, |\psi'_0\rangle, \Sigma, \mu')$ that is equivalent to $\mathcal{A}$ and has the smallest number of states from all k-letter automata equivalent to A.

For a given k-letter QFA $\mathcal{A}$ with $|Q| = n$ we define the set

$$\mathbb{S}^{(n')}_{\mathcal{A},\Sigma} = \{\mathcal{A}' : \mathcal{A}' \text{ is a k-letter QFA equivalent to } \mathcal{A} \text{ over } \Sigma \text{ with } n' \text{ states}\}. \tag{42}$$

The minimization algorithm is now depicted in Figure 3.

Figure 3. Algorithm (III) for the minimization of multi-letter QFAs.

Input: a k-letter QFA $\mathcal{A}$ with n states
output: a minimal k-letter QFA $\mathcal{A}^{'}$ equivalent to $\mathcal{A}$
Step 1:

For $i = 1$ to n
If ($\mathbb{S}^{(i)}_{\mathcal{A},\Sigma}$ is not empty) return $\mathcal{A}' =$ sample $\mathbb{S}^{(i)}_{\mathcal{A},\Sigma}$

Step 2:

return $\mathcal{A}' = \mathcal{A}$

To verify the algorithm, we will use Theorem 5.1 to show that both problem: to decide if $\mathbb{S}^{(i)}_{\mathcal{A},\Sigma}$ is non-empty and to find a sample of $\mathbb{S}^{(i)}_{\mathcal{A},\Sigma}$ are decidable/solvable. By Theorem 5.1, it is sufficient to show that $\mathbb{S}^{(n')}_{\mathcal{A},\Sigma}$ can be described by some polynomial equations and/or polynomial inequations.

Let $\mathcal{A}' = (Q', Q'_{acc}, |\psi'_0\rangle, \Sigma, \mu') \in \mathbb{S}^{(n')}_{\mathcal{A},\Sigma}$. Suppose $|\psi'_0\rangle = [x_1, x_2, \cdots, x_{n'}]^T$ where T denotes the transpose operation. Then

$$\sum_{i=1}^{n'} x_i x_i^* = 1 \tag{43}$$

where $*$ denotes the conjugate operation. Due to the analysis mentioned in Remark 5.2 we can use two real polynomial equations with $2n'$ real variables to describe that $|\psi'_0\rangle$ is a unit vector in $\mathbb{C}^{n'}$.

For any $w \in (\{\Lambda\} \cup \Sigma)^k$, $\mu(w)$ is an $n' \times n'$ unitary matrix. Suppose that $\mu(w) = [y_{ij}(w)]$ and therefore

$$[y_{ij}(w)] \times [y_{ij}(w)]^\dagger = I \tag{44}$$

where $\dagger$ denotes the conjugate transpose operation. Thus we can use $2n'^2$ real polynomial equations with $2n'^2$ real variables to describe that $\mu(w)$ is a unitary matrix. Note that to present $\mathcal{A}'$ we should describe $\mu(w)$ for every $w \in (\{\Lambda\} \cup \Sigma)^k$ (more exactly, for $w \in \Lambda^i \Sigma^{k-i}, i = k-1, k-2, \cdots, 0$). Thus, the number of $\mu(w)$'s is

$$N = |\Sigma|^1 + |\Sigma|^2 + \cdots + |\Sigma|^k. \tag{45}$$

The accepting states set Q'_{acc} can be characterized by an n'-dimensional vector $|\eta_{acc}\rangle = (z_1, z_2, \ldots, z_{n'})$ with entries 0 or 1, where $z_i = 1$ means that the state q_i is an accepting state, and $z_i = 0$ means that the state q_i is a rejecting state. We will say that $|\eta_{acc}\rangle$ is the characteristic vector of the accepting set Q'_{acc}. Thus, the accepting set Q'_{acc} can be described by n' real variables with polynomial equations such as

$$z_i = 1 \text{ or } z_i = 0. \tag{46}$$

Since $\mathcal{A}'$ is equivalent to $\mathcal{A}$, for each $x \in \Sigma^*$ with $|x| \leq ((n+n')^2 m^{k-1} - m^{k-1} + k)$, by Theorem 3.5 we have the following equation:

$$\|\langle\psi'_0|\overline{\mu'}(x)P'_{acc}\|^2 = \|\langle\psi_0|\overline{\mu}(x)P_{acc}\|^2 \tag{47}$$

where $\overline{\mu}(x)$ and $\overline{\mu}(x)'$ are associated with $\mathcal{A}$ and $\mathcal{A}'$, and are products of $|x|$ unitary matrices, as defined in Eq. (7). Equivalently, we can represent the accepting probability of $\mathcal{A}'$ on an input x as

$$\|\langle\psi'_0|\overline{\mu'}(x)P'_{acc}\|^2 = \sum_{i=1}^{n'} |\langle\psi'_0|\overline{\mu'}(x)|z_i\rangle|^2 \tag{48}$$

$$= \langle\psi'_0| \otimes \langle\psi'_0|^* \overline{\mu'}(x) \otimes \overline{\mu'}(x)^* \sum_{i=1}^{n'} |z_i\rangle \otimes |z_i\rangle^* \tag{49}$$

where $|z_i\rangle$ is an n'-dimensional column vector with the ith element being the value of the ith element of $|\eta_{acc}\rangle = (z_1, z_2, \ldots, z_{n'})$ and others being 0's. It can be seen that Eq. (49) always has a real value and, furthermore, it can be represented by a real polynomial with variables that describe $|\psi'_0\rangle$, $\mu(w)$ and Q'_{acc}. It is also easy to see that the degree of this polynomial is $2(|x|+2)$. Thus, for each $x \in \Sigma^*$ with $|x| \leq ((n+n')^2 m^{k-1} - m^{k-1} + k)$, where $m = |\Sigma|$, Eq. (47) can be described by a real polynomial equation, since the left side, as we have shown, is a real polynomial, and the right side is a fixed value for the given k-letter QFA $\mathcal{A}$ (of course, some time is need to compute this value for the given $\mathcal{A}$). Note that to describe the fact that $\mathcal{A}'$ and $\mathcal{A}$ are equivalent, the total number of polynomial equations needed is

$$P = |\Sigma|^1 + |\Sigma|^2 + \cdots + |\Sigma|^{(n+n')^2 m^{k-1} - m^{k-1} + k}. \tag{50}$$

The above statements and analysis can now be summarized as follows: for a given k-letter QFA $\mathcal{A}$ over an input alphabet Σ, any k-letter QFA $\mathcal{A}' \in \mathbb{S}_{\mathcal{A},\Sigma}^{(n')}$ that is equivalent to $\mathcal{A}$ can be represented by a vector $x \in \mathbb{R}^{2Nn'^2+3n'}$, where N is given by Eq. (45), with real polynomial equations from Eqs. (43), (44), (46), (47). The total number of the polynomial equations needed is

$$M = 2 + 2Nn'^2 + 2n' + P. \tag{51}$$

The highest degree in these equations is

$$d = 2((n+n')^2 m^{k-1} - m^{k-1} + k + 2). \tag{52}$$

Thus, by Theorem 5.1, for every $n' \leq n$ there exists an algorithm to decide if $\mathbb{S}_{\mathcal{A},\Sigma}^{(n')}$ is not empty and time cost of that is (where $m = |\Sigma|$)

$$(Md)^{O(2Nn'^2+3n')} = O\left(\left(n^2 m^{n^2 m^{k-1}+k-1}\right)^{m^k n^2}\right). \tag{53}$$

If we view m, k as constants, then time complexity is $O((2c)^{n^4})$. Furthermore, if the set $\mathbb{S}_{\mathcal{A},\Sigma}^{(n')}$ is non-empty, then there exists an algorithm to find a sample of $\mathbb{S}_{\mathcal{A},\Sigma}^{(n')}$ in space

$$\tau d^{O(2Nn'^2+3n')} = O\left(\tau\left(n^2 m^{k-1}\right)^{m^k n^2}\right). \tag{54}$$

If we view m, k as constants, then the space complexity is $O(\tau 2^{n^4})$.

Therefore, the procedures described in Figure 3 can be used to find a minimal k-letter QFA equivalent to a given k-letter QFA. By that we have completed the proof of Theorem 5. □

Since for the case $k = 1$ k-letter QFAs are exactly MO-1QFAs, we have that the state minimization problem for MO-1QFAs is decidable in EXPSPACE.

6 Concluding remarks and some open problems

Because the method to address the equivalence problem for multi-letter QFAs with single-symbol input alphabet in [25] does not seem to be applied to the case of multi-symbol input alphabet directly, we have solved the decidability of the equivalence problem of multi-letter QFAs with a different method. More exactly, we have proved that any two automata, a k_1-letter QFAs $\mathcal{A}_1$ and a k_2-letter $\mathcal{A}_2$ over the input alphabet $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_m\}$, are equivalent if and only if they are $(n^2 m^{k-1} - m^{k-1} + k)$-equivalent, where $n = n_1 + n_2$, n_1 and n_2 are numbers of states of $\mathcal{A}_1$ and $\mathcal{A}_2$, respectively, and $k = \max(k_1, k_2)$. By that, we have also obtained decidability of equivalence problem for multi-letter QFAs over one letter alphabet $\Sigma = \{\sigma\}$ and decidability of equivalence problem for MO-1QFAs.

However, if the equivalence of multi-letter QFAs is determined by checking all strings x with $|x| \leq n^2 m^{k-1} - m^{k-1} + k$, then the time complexity is exponential. Therefore, we have designed another polynomial-time $O(m^{2k-1} n^8 + k m^k n^6)$

algorithm for determining the equivalence of any two multi-letter QFAs. Furthermore, we have proved that the state minimization problem for multi-letter QFAs is decidable in EXPSPACE.

Finally, it is worth pointing out that although we have given an upper bound on the length of strings to be verified for determining whether two multi-letter QFAs are equivalent, the optimality of the upper bound has not been discussed, and this is left for further consideration. Another open problem is to explore the relation between the state complexity of multi-letter QFAs compared with MO-1QFAs for accepting some languages (for example, unary regular languages [29, 32]).

Acknowledgments

This work was partially supported by the National Natural Science Foundation (No. 60873055, 61073054), the Natural Science Foundation of Guangdong Province of China (No. 10251027501000004), the Fundamental Research Funds for the Central Universities (No. 10lgzd12,11lgpy36), the Research Foundation for the Doctoral Program of Higher School of Ministry of Education (Nos. 20100171110042, 20100171120051), the Program for New Century Excellent Talents in University (NCET) of China, the China Postdoctoral Science Foundation project (Nos. 20090460808, 201003375), and the project of SQIG at IT, funded by FCT and EU FEDER projects projects QSec PTDC/EIA/67661/2006, AMDSC UTAustin/MAT/0057/2008, NoE Euro-NF, and IT Project QuantTel.

References

[1] M. Amano, K. Iwama, Undecidability on Quantum Finite Automata, in: Proceedings of the 31st Annual ACM Symposium on Theory of Computing, Atlanta, Georgia, USA, 1999, pp. 368-375.

[2] A. Ambainis, R. Freivalds, One-way quantum finite automata: strengths, weaknesses and generalizations, in: Proceedings of the 39th Annual Symposium on Foundations of Computer Science, IEEE Computer Society Press, Palo Alfo, California, USA, 1998, pp. 332-341. Also quant-ph/9802062, 1998.

[3] A. Belovs, A. Rosmanis, and J. Smotrovs, Multi-letter Reversible and Quantum Finite Automata, in: Proceedings of the 13th International Conference on Developments in Language Theory (DLT'2007), Lecture Notes in Computer Science, Vol. 4588, Springer, Berlin, 2007, pp. 60-71.

[4] A. Bertoni, C. Mereghetti, B. Palano, Quantum Computing: 1-Way Quantum Automata, in: Proceedings of the 9th International Conference on Developments in Language Theory (DLT'2003), Lecture Notes in Computer Science, Vol. 2710, Springer, Berlin, 2003, pp. 1-20.

[5] A. Brodsky, N. Pippenger, Characterizations of 1-way quantum finite automata, *SIAM Journal on Computing* 31 (2002) 1456-1478. Also quant-ph/9903014, 1999.

[6] S. Basu, R. Pollack, and M.-F. Roy, Algorithms in Real Algebraic Geometry, 2nd Edition, Springer, 2006.

[7] J. Canny, Some algebraic and geometric computations in PSPACE, in STOC '88: Proceedings of the twentieth annual ACM symposium on Theory of computing, ACM, New York, USA, 1988, pp. 460-469.

[8] D.K. Faddeev, V.N. Faddeeva, Computational Methods of Linear Algebra, Freeman, San Francisco, 1963.

[9] L.K. Grover, A fast quantum mechanical algorithm for database search, in: Proceedings of the 28th Annual ACM Symposium on Theory of Computing, Philadelphia, Pennsylvania, USA, 1996, pp. 212-219.

[10] J. Gruska, Quantum Computing, McGraw-Hill, London, 1999.

[11] M. Hirvensalo, Improved Undecidability Results on the Emptiness Problem of Probabilistic and Quantum Cut-Point Languages, in: SOFSEM'2007, Lecture Notes in Computer Science, Vol. 4362, Springer, Berlin, 2007, pp. 309-319.

[12] J. Hromkovič, One-way multihead deterministic finite automata, *Acta Informatica* 19 (1983) 377-384.

[13] J.E. Hopcroft, J.D. Ullman, Introduction to Automata Theory, Languages, and Computation, Addision-Wesley, New York, 1979.

[14] O.H. Ibarra, C.E. Kim, On 3-Head Versus 2-Head Finite Automata, *Acta Informatica* 4 (1975) 193-200.

[15] E. Jeandel, Topological Automata, *Theory of Computing Systems* 40 (2007) 397-407.

[16] A. Kondacs, J. Watrous, On the power of finite state automata, in: Proceedings of the 38th IEEE Annual Symposium on Foundations of Computer Science, Miami Beach, Florida, USA, 1997, pp. 66-75.

[17] L.Z. Li, D.W. Qiu, Determination of equivalence between quantum sequential machines, *Theoretical Computer Science* 358 (2006) 65-74.

[18] L.Z. Li, D.W. Qiu, Determining the equivalence for one-way quantum finite automata, *Theoretical Computer Science* 403 (2008) 42-51.

[19] L.Z. Li, D.W. Qiu, A note on quantum sequential machines, *Theoretical Computer Science* 410 (2009) 2529-2535.

[20] C. Moore, J.P. Crutchfield, Quantum automata and quantum grammars, *Theoretical Computer Science* 237 (2000) 275-306. Also quant-ph/9707031, 1997.

[21] A. Paz, Introduction to Probabilistic Automata, Academic Press, New York, 1971.

[22] P. Mateus, D.W. Qiu, On the complexity of minimizing probabilistic and quantum automata. Manuscript.

[23] D.W. Qiu, Characterization of Sequential Quantum Machines, *International Journal of Theoretical Physics* 41 (2002) 811-822.

[24] D.W. Qiu, L.Z. Li, An overview of quantum computation models: quantum automata, *Frontiers of Computer Science in China* 2 (2)(2008) 193-207.

[25] D.W. Qiu, S. Yu, Hierarchy and equivalence of multi-letter quantum finite automata, *Theoretical Computer Science* 410 (2009) 3006-3017. Also arXiv: 0812.0852, 2008.

[26] M. O. Rabin, Probabilistic Automata, *Information and Control* 6 (3) (1963) 230-245.

[27] J. Renegar, A faster PSPACE algorithm for deciding the existential theory of the reals, in: Proc. 29th IEEE Annu. Symp. on Foundations of Computer Science, 1988, pp. 291-295.

[28] S. Roman, Lattices and ordered sets, Springer, New York, 2008.

[29] G. Rozenberg, A. Salomaa (Eds.), Handbook of Formal Languages, Vol. 1, Springer-Verlag, Berlin, 1997.

[30] P.W. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, SIAM Journal on Computing 26 (5) (1997) 1484-1509.

[31] W.G. Tzeng, A Polynomial-time Algorithm for the Equivalence of Probabilistic Automata, *SIAM Journal on Computing* 21 (2) (1992) 216-227.

[32] S. Yu, Regular Languages, In: G. Rozenberg, A. Salomaa (Eds.), Handbook of Formal Languages, Springer-Verlag, Berlin, 1998, pp. 41-110.

[33] A. Yakaryilmaz, A. C. Cem Say, Languages recognized with unbounded error by quantum finite automata, In Proceedings of the 4th Computer Science Symposium in Russia, Lecture Notes in Comput. Sci. 5675, Springer-Verlag, Berlin, 2009, pp. 356-367.

[34] A. Yakaryilmaz, A. C. Cem Say, Unbounded-error quantum computation with small space bounds, arXiv:1007.3624, 2010.