Determining the equivalence for one-way quantum finite automata

Lvzhou Li, Daowen Qiu *

Department of Computer Science, Zhejiang University, Hangzhou 310027, People’s Republic of China

A R T I C L E   I N F O

Article history:
Received 25 April 2007
Received in revised form 25 October 2007
Accepted 21 March 2008
Communicated by M. Hirvensalo

Keywords:
Quantum computing
Quantum finite automata
Equivalence

A B S T R A C T

Two quantum finite automata are equivalent if for any input string $x$ the two automata accept $x$ with equal probability. In this paper, we first focus on determining the equivalence for one-way quantum finite automata with control language (CL-1QFAs) defined by Bertoni et al., and then, as an application, we address the equivalence problem for measure-many one-way quantum finite automata (MM-1QFAs) introduced by Kondacs and Watrous. More specifically, we obtain that:

(i) Two CL-1QFAs $A_1$ and $A_2$ with control languages (regular languages) $L_1$ and $L_2$, respectively, are equivalent if and only if they are $(c_1n_1^2 + c_2n_2^2 - 1)$-equivalent, where $n_1$ and $n_2$ are the numbers of states in $A_1$ and $A_2$, respectively, and $c_1$ and $c_2$ are the numbers of states in the minimal DFAs that recognize $L_1$ and $L_2$, respectively. Furthermore, if $L_1$ and $L_2$ are given in the form of DFAs, with $m_1$ and $m_2$ states, respectively, then there exists a polynomial-time algorithm running in time $O((m_1n_1^2 + m_2n_2^2)^4)$ that takes as input $A_1$ and $A_2$ and determines whether they are equivalent.

(ii) (As an application of item (i)): Two MM-1QFAs $A_1$ and $A_2$ with $n_1$ and $n_2$ states, respectively, are equivalent if and only if they are $(3n_1^2 + 3n_2^2 - 1)$-equivalent. Furthermore, there is a polynomial-time algorithm running in time $O((3n_1^2 + 3n_2^2)^4)$ that takes as input $A_1$ and $A_2$ and determines whether $A_1$ and $A_2$ are equivalent.

© 2008 Elsevier B.V. All rights reserved.

1. Introduction

Over the past two decades, quantum computing has attracted wide attention in the academic community [16,27]. To a certain extent, this was motivated by the exponential speed-up of Shor’s quantum algorithm for factoring integers in polynomial time [31] and afterwards Grover’s algorithm of searching in a database of size $n$ with only $O(\sqrt{n})$ accesses [14]. As we know, these algorithms are based on quantum Turing machines or quantum circuits that seem to be complicated to implement using today’s experiment technology. (Note that quantum Turing machines and quantum circuits, as two important models of quantum computation, were proved to be equivalent by Yao [36], and the former are a base for quantum complexity theory such as in references [1,12,32].) Therefore, it is natural to consider simpler models of quantum computation.

Classically, finite automata (FAs), as one of the simplest models of computation, have been deeply studied [19]. Then, as a quantum variant of FAs, quantum finite automata (QFAs) were developed and have received extensive attention from the academic community. QFAs were first introduced independently by Moore and Crutchfield [26], as well as by Kondacs

* This research was supported by the National Natural Science Foundation (Nos. 90303024, 60573006), the Research Foundation for the Doctoral Program of Higher School of Ministry of Education (No. 20050558015), and Program for New Century Excellent Talents in University (NCET) of China.

* Corresponding author. Tel.: +86 02038458089; fax: +86 02084037549.
E-mail address: issqdw@mail.sysu.edu.cn (D. Qiu).

0304-3975/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2008.03.021
and Watrous [23], and then they were intensively investigated by others [2–11,20]. QFAs can be mainly divided into two kinds: one-way quantum finite automata (1QFAs) whose tape heads move only one cell to right at each evolution, and two-way quantum finite automata (2QFAs), in which the tape heads are allowed to move towards right or left, or to be stationary. (Notably, Amano and Iwama [6] dealt with an intermediate form called 1.5QFAs, whose tape heads are allowed to move right or to be stationary, and, particularly, they showed that the emptiness problem for this restricted model is undecidable.)

Furthermore, by means of the measurement times in a computation, 1QFAs have two fashions: measure-once 1QFAs (MO-1QFAs) proposed by Moore and Crutchfield [26], and, measure-many 1QFAs (MM-1QFAs) studied first by Kondacs and Watrous [23]. In addition, in terms of the kind of measurement allowed, both MO-1QFAs and MM-1QFAs allow only very restricted measurement: MO-1QFAs allow projective measurement with only two results: acceptance and rejection; MM-1QFAs allow projective measurement with only three results: acceptance, rejection and continuation. As we know, measurement is an important operation in quantum computation and quantum information. Then in Refs. [4,10,5], some more general quantum models were proposed and characterized, in which any projective measurement was allowed as a valid intermediate computational step. Particularly, Bertoni et al. [10] characterized a model called one-way quantum finite automata with control language (CL-1QFAs). We will detail it later on.

In addition to QFAs, there are some other types of finite-like quantum automata that are being developed, such as quantum push-down automata (QPDAs) [18], quantum one-counter automata [35], and quantum sequential machines (QSMs) [15,29]. Some interesting results have been obtained on these models, and we do not detail them here.

So far, work on QFAs has mainly focused on characterizing the language recognized by QFAs and comparing them with their classical analogues (finite automata and probabilistic finite automata [30,28]). We briefly state some main results obtained. The class of languages recognized by CL-1QFAs with bounded error probabilities is strictly bigger than that by MM-1QFAs which, in turn, recognize the class of languages strictly bigger than that by MO-1QFAs. However, all of them recognize only the subclass of regular languages with bounded error probabilities [26,23,11,10]. Also, the class of languages recognized by MM-1QFAs with bounded error probabilities is not closed under the binary Boolean operations [11,10]. Concerning 2QFAs, an exciting result was obtained by Kondacs and Watrous [23] that some 2QFA can recognize non-regular language $L_{eq} = \{a^i b^i | n > 0\}$ with one-sided error probability in linear time, which cannot be attained by the classical analogues (even by two-way probabilistic automata).

Although more and more problems concerning the models of quantum computation have been clarified, there are still some fundamental problems left open. One of these problems is to determine the equivalence for these models. As we know, determining the equivalence for computing models is a very important issue in the theory of classical computation. For instance, both the references [19,28] were devoted to this issue and good results were obtained. Concerning the problem of determining the equivalence for QFAs, there exists only little work [11,21] that deals with the simplest case—MO-1QFAs. For the equivalence problem of MM-1QFAs, Koshiba [21] had a try, but we found that the method in [21] does not work for MM-1QFAs. Lately, we have learned from [22] that Koshiba also mentioned at the ISAAC2001 conference that his method has a flaw. Thus, in fact, the problem for MM-1QFAs is still left open. To our knowledge, there seems to be no more related work on this problem, except for some work on QSMs [24].

In this paper, we first focus on determining the equivalence between CL-1QFAs, and then, as an application, we address the equivalence problem for MM-1QFAs. Sufficient and necessary conditions for deciding the equivalence are obtained.

Also, we present some polynomial-time algorithms to judge the equivalence for CL-1QFAs and MM-1QFAs, respectively. The remainder of this paper is organized as follows. Some models and related definitions are introduced in Section 2. Section 3 is the main part of the paper which will deal with the problem stated above. In Section 3.1, we introduce some definitions and related results that will be used in the later subsections. Then we deal with the equivalence problems for CL-1QFAs and MM-1QFAs in Sections 3.2 and 3.3, respectively. Finally, some concluding remarks are made in Section 4.

2. Preliminaries

2.1. Some notation on linear algebra and quantum mechanics

As usual, for non-empty set $\Sigma$, by $\Sigma^*$ we mean the set of all finite length strings over $\Sigma$, and by $\Sigma^n$ we mean the set of all strings over $\Sigma$ with length $n$. For $u \in \Sigma^*$, $|u|$ denotes the length of $u$; for example, if $u = x_1x_2 \ldots x_m \in \Sigma^*$ where $x_i \in \Sigma$, then $|u| = m$. For set $S$, $|S|$ denotes the cardinality of $S$.

Let $\mathbb{C}$ denote the set of all complex numbers, $\mathbb{R}$ the set of all real numbers, and $\mathbb{C}^{n \times m}$ the set of all $n \times m$ matrices having entries in $\mathbb{C}$. Given two matrices $A \in \mathbb{C}^{n \times m}$ and $B \in \mathbb{C}^{p \times q}$, their Kronecker product is the $np \times mq$ matrix, defined as

$$A \otimes B = \begin{bmatrix} A_{11}B & \cdots & A_{1m}B \\ \vdots & \ddots & \vdots \\ A_{n1}B & \cdots & A_{nm}B \end{bmatrix}.$$  

For the above matrices $A$ and $B$, if $m = p$, then $A$ multiplying $B$, denoted by $AB$, is allowed. When $A$ multiplying $C$ and $B$ multiplying $D$ are allowed, we get $(A \otimes B)(C \otimes D) = AC \otimes BD$. Matrix $M \in \mathbb{C}^{n \times n}$ is said to be unitary if $MM^\dagger = M^\dagger M = I$, where $\dagger$ denotes the conjugate-transpose operation. $M$ is said to be Hermitian if $M = M^\dagger$. For $n$-dimensional row vector
when we analyze the time complexity of the algorithms determining the equivalence of

\[ \| x \| = \left( \sum_{i=1}^{n} x_i^* x_i \right)^{1/2} \]

where the symbol * denotes conjugate operation. Unitary matrices preserve the norm, i.e., \( \| M \| = \| x \| \) for each \( x \in \mathbb{C}^{1 \times n} \) and unitary matrix \( M \in \mathbb{C}^{n \times n} \). An \( n \)-dimensional row vector \( a = (a_1, a_2, \ldots, a_n) \) is said to be stochastic if \( a_i \geq 0 \) (\( i = 1, 2, \ldots, n \)), and \( \sum_{i=1}^{n} a_i = 1 \); in particular, \( a \) is called a degenerate stochastic vector if one of the entries is 1 and the others 0’s. A matrix is said to be stochastic if its each row is a stochastic vector.

We would refer the reader to [16,27] for a thorough treatment on the postulates of quantum mechanics, and here we just briefly introduce some notation to be used in this paper. For a quantum system with a finite basic state set \( Q = \{ q_1, \ldots, q_n \} \), every basic state \( q_i \) can be represented by an \( n \)-dimensional row vector \( \langle q_i \rangle = (0 \ldots 1 \ldots 0) \) having only 1 at the \( i \)th entry. At any time, the state of this system is a superposition of these basic states and can be represented by a row vector \( \langle \phi \rangle = \sum_{i=1}^{n} c_i \langle q_i \rangle \) with \( c_i \in \mathbb{C} \) and \( \sum_{i=1}^{n} |c_i|^2 = 1 \). If we want to get some information from a quantum system, then we should make a measurement on it. Here we consider projective measurement (von Neumann measurement). A projective measurement is described by an observable that is a Hermitian matrix \( \Theta = c_1 P_1 + \cdots + c_s P_s \), where \( c_i \) is its eigenvalue and, \( P_i \) is the projector onto the eigenspace corresponding to \( c_i \). In this case, the projective measurement of \( \Theta \) has result set \( \{ c_i \} \) and projector set \( \{ P_i \} \).

We assume that the operations of addition and multiplication on two complex numbers can all be done in constant time, which will be used in Section 3 when we analyze the time complexity of the algorithms determining the equivalence between QFAs.

2.2 Classical computing models

Firstly, we give a mathematical model which is not an actual computing model but generalizes many classical computing models, which will play a foundational role in this paper.

Definition 1. A bilinear machine (BLM) over the alphabet \( \Sigma \) is a four-tuple \( \mathcal{M} = (S, \pi, [M(\sigma)]_{\sigma \in \Sigma}, \eta) \), where \( S \) is a finite state set with \( |S| = n \), \( \pi \in \mathbb{C}^{1 \times n} \), \( \eta \in \mathbb{C}^{n \times 1} \) and \( M(\sigma) \in \mathbb{C}^{n \times n} \) for \( \sigma \in \Sigma \).

Associated to a BLM \( \mathcal{M} \), the word function \( f_\mathcal{M} : \Sigma^* \to \mathbb{C} \) is defined in the way: \( f_\mathcal{M}(w) = \pi M(w_1) \ldots M(w_n) \eta \), where \( w = w_1 \ldots w_n \in \Sigma^* \). In particular, when \( f_\mathcal{M}(w) \in \mathbb{R} \) for every \( w \in \Sigma^* \), \( \mathcal{M} \) is called a real-valued bilinear machine (RBLM).

Turakainen [33] defined a model called generalized automata (GAs) and characterized the languages recognized by them. In fact, a GA \( \mathcal{M} \) is just a BLM with the restriction that \( \pi, \eta \), and \( M(\sigma) (\sigma \in \Sigma) \) have components in \( \mathbb{R} \). The word function \( f_\mathcal{M} \) associated to GA \( \mathcal{M} \) is defined as in the case of BLMs.

Another important computing model is the so-called probabilistic automata (PAs) [30,28]. A PA is a GA with the restriction that \( \pi \) is a stochastic vector, \( \eta \) consists of the entries with 0’s and 1’s only, and the matrices \( M(\sigma) (\sigma \in \Sigma) \) are stochastic. Then, the word function \( f_\mathcal{M} \) associated to PA \( \mathcal{M} \) has range \([0, 1]\). Given a PA \( \mathcal{M} \), if \( \pi \) is a degenerate stochastic vector, and stochastic matrices \( M(\sigma) (\sigma \in \Sigma) \) consist of the entries with 0’s and 1’s only, then \( \mathcal{M} \) is called a deterministic finite automaton (DFA) [19]. Then, the word function \( f_\mathcal{M} \) associated to DFA \( \mathcal{M} \) has range \([0, 1]\).

The language \( \mathcal{L} \) recognized by DFA \( \mathcal{M} \) is defined by the following set:

\[ \mathcal{L} = \{ w : w \in \Sigma^* \text{ and } f_\mathcal{M}(w) = 1 \} \]

In this case, we also call the function \( f_\mathcal{M} \) as the characterization function of \( \mathcal{L} \), denoted by \( \chi_{\mathcal{L}} \), where for any \( x \in \Sigma^* \),

\[ \chi_{\mathcal{L}}(x) = \begin{cases} 1 & x \in \mathcal{L} \\ 0 & x \notin \mathcal{L} \end{cases} \]

It is well known that DFAs can recognize only regular languages, and for every regular language \( \mathcal{L} \), there is a minimal DFA recognizing it.

From the above definitions, it is readily seen that:

\( \text{DFAs} \subset \text{PAs} \subset \text{GAs} \subset \text{RBLMs} \subset \text{BLMs} \).

2.3 Quantum computing models

In this paper, only one-way quantum computing models are considered. So in the following, when introducing quantum models, we always leave out the word “one-way”.

The simplest quantum computing model may be the MO-1QFAs firstly defined by [26]. In this model, the transformation on any symbol in the input alphabet is realized by a unitary operator, and a unique measurement is performed at the end of a computation. In the interest of compactness, we do not detail the definition for MO-1QFAs, whereas in the following, we will give the detailed definitions for MM-1QFAs and CL-1QFAs which are the focus of this paper.
Measure-many quantum finite automata (MM-1QFAs). Unlike MO-1QFAs that allow only one measurement at the end of a computation, MM-1QFAs allow measurement at each step. Due to this difference, MM-1QFAs are more powerful than MO-1QFAs.

Formally, given an input alphabet $\Sigma$ and an end-maker $\$ \notin \Sigma$, an MM-1QFA with $n$ states over the working alphabet $\Gamma = \Sigma \cup \{\$\}$ is a four-tuple $\mathcal{M} = (Q, \pi, \{U(\sigma)\}_{\sigma \in \Gamma}, \Theta)$, where

- $Q = \{q_1, \ldots, q_n\}$ is the basic state set; at any time, the state of $\mathcal{M}$ is a superposition of these basic states;
- $\pi \in \mathbb{C}^{1 \times n}$ with $\|\pi\| = 1$ is the initial vector;
- for any $\sigma \in \Gamma$, $U(\sigma) \in \mathbb{C}^{n \times n}$ is a unitary matrix;
- $\Theta$ is an observable described by the projectors $P(a), P(r)$ and $P(g)$, with the results in $\{a, r, g\}$ of which 'a', 'r' and 'g' denote "accept", "reject" and "go on", respectively.

Any input word $w$ to MM-1QFAs is in the form: $w \in \Sigma^*$, with symbol $\$ denoting the end of a word. Given an input word $x_1 \ldots x_n\$ where $x_1 \ldots x_n \in \Sigma^n$, MM-1QFA $\mathcal{M}$ performs the following computation:

1. Starting from $\pi$, $U(x_1)$ is applied, and then we get a new state $|\phi_1\rangle = \pi U(x_1)$. In succession, a measurement of $\Theta$ is performed on $|\phi_1\rangle$, and then the measurement result $i(i \in \{a, g, r\})$ is yielded as well as a new state $|\phi'_1\rangle = \frac{|\phi_1\rangle\langle\phi_1|_{P(i)}}{\|P(i)\|}$ is obtained, with corresponding probability $p'_1 = \|P(i)\|^2$.
2. In the above step, if $|\phi'_1\rangle$ is obtained, then starting from $|\phi'_1\rangle$, $U(x_2)$ is applied and a measurement of $\Theta$ is performed. The evolution rule is the same as the above step.
3. The process continues as far as the measurement result 'g' is yielded. As soon as the measurement result is 'a'('r'), the computation halts and the input word is accepted (rejected).

Thus, MM-1QFA $\mathcal{M}$ defines a word function $f_{\mathcal{M}} : \Sigma^* \rightarrow [0, 1]$ in the following:

$$f_{\mathcal{M}}(x_1 \ldots x_n\$) = \sum_{k=1}^{\|\pi\|^2} \prod_{i=1}^{k-1} (U(x_i)p(g))U(x_k)p(a),$$

or equivalently,

$$f_{\mathcal{M}}(x_1 \ldots x_n\$) = \sum_{k=1}^{\|\pi\|^2} \prod_{i=1}^{k} (U(x_i)p(g))U(x_k+1)p(a),$$

where, for simplicity, we denote $\$ by $x_{n+1}$ and we will always use this denotation in the following. $f_{\mathcal{M}}(x_1 \ldots x_n\$) is the probability of $\mathcal{M}$ accepting the word $x_1 \ldots x_n$, and usually, we would like to use another function denoted by $P_{\mathcal{M}} : \Sigma^* \rightarrow [0, 1]$ to denote this probability such that $P_{\mathcal{M}}(x_1 \ldots x_n) = f_{\mathcal{M}}(x_1 \ldots x_n\$).

Quantum automata with control language(CL-1QFAs). Bertoni et al. [10] introduced a new one-way quantum computing model that allows a more general measurement than the previous models. Similarly to the case in MM-1QFAs, the state of this model can be observed at each step, but an observable $\Theta$ is considered with a fixed, but arbitrary, set of possible results $C = \{c_1, \ldots, c_s\}$, without limit to $\{a, r, g\}$ as in MM-1QFAs. The accepting behavior in this model is also different from that of the previous models. On any given input word $x$, the computation displays a sequence $y \in \mathbb{C}^*$ of results of $\Theta$ with a certain probability $p(y|x)$, and the computation is accepted if and only if $y$ belongs to a fixed regular language $L \subseteq \mathbb{C}^*$. Visually, Bertoni et al. [10] called such a language $L$ control language. One should not confuse the term "control language" in this context with that in [13].

More formally, given an input alphabet $\Sigma$ and the end-marker symbol $\$ \notin \Sigma$, a CL-1QFA over the working alphabet $\Gamma = \Sigma \cup \{\$\}$ is a five-tuple $\mathcal{M} = (Q, \pi, \{U(\sigma)\}_{\sigma \in \Gamma}, \Theta, L)$, where

- $Q$ is the set of states; $\pi$ is the initial state; $U(\sigma)$ is the transition function for any symbol $\sigma$ in $\Gamma$;
- $\Theta$ is an observable described by the projectors $P(c_i) : c_i \in C$ and the projector set $\{P(c_i) : i = 1, \ldots, s\}$ of which $P(c_i)$ denotes the projector onto the eigenspace corresponding to $c_i$;
- $L \subseteq \mathbb{C}^*$ is a regular language (control language).

The input word $w$ to CL-1QFA $\mathcal{M}$ is in the form: $w \in \Sigma^*$, with symbol $\$ denoting the end of a word. Now, we define the behavior of $\mathcal{M}$ on word $x_1 \ldots x_n\$. The computation starts in the state $\pi$, and then the transformations associated with symbols in the word $x_1 \ldots x_n\$ are applied in succession. The transformation associated with any symbol $\sigma \in \Gamma$ consists of two steps:

1. Firstly, $U(\sigma)$ is applied to the current state $|\phi\rangle$ of $\mathcal{M}$, yielding the new state $|\phi\rangle = |\phi\rangle U(\sigma)$.
2. Secondly, the observable $\Theta$ is measured on $|\phi\rangle$. According to quantum mechanics principle, this measurement yields result $c_k$ with probability $p_k = \|P(c_k)|\|^2$, and the state of $\mathcal{M}$ collapses to $|\phi\rangle P(c_k)/\sqrt{p_k}$.
Thus, the computation on word $x_1 \ldots x_n$ leads to a sequence $y_1 \ldots y_{n+1} \in E^*$ with probability $p(y_1 \ldots y_{n+1} | x_1 \ldots x_n)$ given by

$$p(y_1 \ldots y_{n+1} | x_1 \ldots x_n) = \left| \prod_{i=1}^{n+1} U(x_i) p(y_i) \right|^2.$$  

(4)

where we let $x_{n+1} = \$$. The computation leading to the word $y \in E^*$ is said to be accepted if $y \in L$. Otherwise, it is rejected. Hence, CL-1QFA $M$ defines a word function $f_M : \Sigma^* \rightarrow [0,1]$ in the form:

$$f_M(x_1 \ldots x_n) = \sum_{y_1 \ldots y_{n+1} \in L} p(y_1 \ldots y_{n+1} | x_1 \ldots x_n).$$  

(5)

which denotes the probability of $M$ accepting the word $x_1 \ldots x_n$. Usually, we also denote this accepting probability by the function $P_M : \Sigma^* \rightarrow [0,1]$ where

$$P_M(x_1 \ldots x_n) = f_M(x_1 \ldots x_n).$$  

(6)

3. Determining the equivalence for computing models

Determining the equivalence for computing models is an important issue in the theory of computation. However, this problem has not been well investigated for quantum computing models. In this section, we will deal with the equivalence problem for CL-1QFAs and MM-1QFAs. Our idea is to first transform these quantum models to BLMs, and then we deal with the equivalence problem for BLMs. So, below we first give some results on BLMs.

3.1. Some definitions and results on BLMs to be used

Firstly, by $x_1 \ldots x_n \$ where $\$ denotes the end-marker, we mean $|x_1 \ldots x_n| = |x_1 \ldots x_n| = n$. Now we give two definitions concerning the equivalence for models.

Definition 2. Two BLMs (including RBLMs, GAs, PAs, and DFAs) $M_1$ and $M_2$ over the same alphabet $\Sigma$ are said to be equivalent (resp. $k$-equivalent) if $f_{M_1}(w) = f_{M_2}(w)$ for any $w \in \Sigma^*$ (resp. for any input string $w$ with $|w| \leq k$).

Definition 3. Two QFAs (including MM-1QFAs and CL-1QFAs) $M_1$ and $M_2$ over the same input alphabet $\Sigma$ are said to be equivalent (resp. $k$-equivalent) if $P_{M_1}(w) = P_{M_2}(w)$ for any $w \in \Sigma^*$ (resp. for any input string $w$ with $|w| \leq k$).

Next we give two propositions concerning BLMs that will be used later. The first one is Proposition 1 in the following that allows us to remove the end-maker $\$ in the input word, as we will see when we deal with the equivalence for QFAs.

Proposition 1. Let BLM $M$ have $n$ states and the alphabet $\Sigma \cup \{\tau\}$ where $\tau \notin \Sigma$. Then we can give another BLM $\hat{M}$ over the alphabet $\Sigma$ with the same states, such that $f_{M}(w\tau) = f_{\hat{M}}(w)$, for any $w \in \Sigma^*$.

Proof. We let $\hat{M}$ be the same as $M$ except that $\hat{\eta} = U(\tau) \eta$, where $\hat{\eta}$ belongs to $\hat{M}$ and $U(\tau)$ and $\eta$ belong to $M$. It is clear that $f_{M}(w\tau) = f_{\hat{M}}(w)$, for any $w \in \Sigma^*$. □

The second one is Proposition 2 that allows us to convert the problems in the field of complex numbers to the ones in the field of real numbers. The idea behind this proposition was first pointed out by Moore and Crutchfield in [26].

Proposition 2. For any RBLM $M$ with $n$ states and the alphabet $\Sigma$, we can construct effectively an equivalent GA $\hat{M}$ with $2n$ states and the same alphabet $\Sigma$.

Proof. It is well known that any complex number $c = a + bi$ has a real matrix representation in the form $c = \begin{bmatrix} a & b \\ -b & a \end{bmatrix}$. Then in the same way any $n \times n$ complex matrix has a representation by a $2n \times 2n$ real matrix. We can also check that given two matrices $A$ and $B$ (assuming that $A$ and $B$ can multiply), if $A$ and $B$ are the real matrix representations of $A$ and $B$, respectively, then $AB$ will be the real matrix representation of $AB$.

Now suppose that we have an $n$-state RBLM $M = (S, \pi, [M(\sigma)]_{\sigma \in \Sigma}, \eta)$. Then for $x = x_1 \ldots x_n \in \Sigma^*$, there is $\pi M(x_1) \ldots M(x_n) \eta = f_M(x) \in R$. Using the above representation, we transform $\pi, M(\sigma)$ and $\eta$ into $2 \times 2n, 2n \times 2n$ and $2n \times 2$ real matrices $\hat{\pi}, \hat{M}(\sigma)$ and $\hat{\eta}$, respectively. Then we have

$$\hat{\pi} \hat{M}(x_1) \ldots \hat{M}(x_n) \hat{\eta} = \begin{bmatrix} f_M(x) & 0 \\ -0 & f_M(x) \end{bmatrix}. \tag{7}$$

Letting $\hat{\pi}'$ be the top row of $\hat{\pi}$ and $\hat{\eta}'$ the left column of $\hat{\eta}$, and letting $M(\sigma) = \hat{M}(\sigma)$ for $\sigma \in \Sigma$, we get the expected GA $M = (S', \pi', [M'(\sigma)]_{\sigma \in \Sigma}, \eta)$ such that $f_M(w) = f_M'(w)$ for any $w \in \Sigma^*$. Therefore, we have completed the proof. □
There is a polynomial-time algorithm running in time \( O((n_1 + n_2)^4) \) that as input two PAs \( A_1 \) and \( A_2 \) and determines whether \( A_1 \) and \( A_2 \) are equivalent, where \( n_1 \) and \( n_2 \) are the numbers of states in \( A_1 \) and \( A_2 \), respectively.

Although Theorem 3 provides a necessary and sufficient condition for the equivalence between PAs, directly testing it needs exponential time. Then Tzeng [34] further provided a polynomial-time algorithm to determine whether two PAs are equivalent. Hence, the equivalence problem of PAs has been solved completely.

**Theorem 4** ([34]). There is a polynomial-time algorithm running in time \( O((n_1 + n_2)^4) \) that as input two PAs \( A_1 \) and \( A_2 \) and determines whether \( A_1 \) and \( A_2 \) are equivalent, where \( n_1 \) and \( n_2 \) are the numbers of states in \( A_1 \) and \( A_2 \), respectively.

In fact, if we refer to [28,34] and read carefully the proofs of Theorems 3 and 4, then we can find that their proofs did not use any essential property of PAs, just based on some ordinary knowledge on matrices and linear spaces, and as a result, the proofs can also be extended to BLMs. Thus, we get a more general result as follows.

**Proposition 5.** Two BLMs (including RBLMs, GAs, PAs, and DFAs) \( A_1 \) and \( A_2 \) with \( n_1 \) and \( n_2 \) states, respectively, are equivalent if and only if they are \( (n_1 + n_2 - 1) \)-equivalent. Furthermore, there exists a polynomial-time algorithm running in time \( O((n_1 + n_2)^4) \) that as input two BLMs \( A_1 \) and \( A_2 \) and determines whether \( A_1 \) and \( A_2 \) are equivalent.

**Remark 1.** (i) The algorithm for BLMs performs the same process as that for PAs. The consuming time of the algorithms for theorems may differ by a constant factor, but with the same magnitude \( O((n_1 + n_2)^4) \), because BLMs are considered in the field of complex numbers while PAs are restricted to the field of real numbers. (ii) When designing the algorithm for RBLMs, in order to avoid the operation on complex numbers, one may first transform RBLMs to GAs by Proposition 2, and then determine the equivalence for GAs, using the algorithm stated in [34]. However, the transforming process is not necessary.

Now we turn to the problem of determining the equivalence for 1QFAs. For the equivalence between MO-1QFAs, some solutions have been obtained by [11,21]. Their idea is to firstly transform MO-1QFAs to RBLMs by the bilinearization technique stated in [26] and in succession transform RBLMs to GAs, and then determine the equivalence for GAs using the results obtained on PAs. As indicated before, transforming RBLMs to GAs is not necessary when dealing with the equivalence between RBLMs.

Due to their complex behaviors, CL-1QFAs and MM-1QFAs may not be bilinearized as Moore and Crutchfield [26] did for MO-1QFAs. Hence, we need new ways to deal with them. Indeed, we find that Bertoni et al. [10] provided a useful technique to our problem. In the following two subsections, we will focus on determining the equivalence for CL-1QFAs and MM-1QFAs.

### 3.2. Determining the equivalence for CL-1QFAs

Determining whether two CL-1QFAs \( A_1 \) and \( A_2 \) are equivalent is to verify whether \( f_{A_1}(xS^*) = f_{A_2}(xS^*) \) holds for any \( x \in \Sigma^* \). We may learn something from how we deal with the equivalence problems for MO-1QFAs [11,21] and QSMs [24], where the ways have a common point, that is, firstly transforming quantum machines to be in a bilinear form and then using some knowledge on matrices and linear spaces to deal with that. However, we can see that the behavior of CL-1QFAs is more complex than those of MO-1QFAs and QSMs. Then we may need some more elaborate work on them.

Below, we will give a key lemma that allows us to transform CL-1QFAs to be in the bilinear form—RBLMs. Then, the equivalence problem of CL-1QFAs is transformed to that of RBLMs which can be solved by using Proposition 5. The idea behind the following lemma mainly originates from Ref. [10].

**Lemma 6.** Any \( m \)-state CL-1QFA \( M \) over the working alphabet \( \Gamma = \Sigma \cup \{S\} \) with control language \( L \) can be simulated by a \((km^2)\)-state RBLM \( \tilde{M} \) over \( \Gamma \), where the factor \( k \) is the number of states in the minimal DFA that recognizes the control language \( L \).

**Proof.** Suppose that we have a CL-1QFA \( M = (Q, \pi, \{U(\sigma)\}_{\sigma \in \Gamma}, \sigma, L) \) with \( m \) states, where the observable \( \sigma \) has eigenvalue set \( C \) and projector set \( \{P(c) : c \in C\} \). Since the control language \( L \subseteq C^* \) is regular, there exists a minimal DFA recognizing \( L \). Then we suppose that DFA \( A = (S, \rho, \{M(c)\}_{c \in C}, \xi) \) recognizes \( L \) with \( |S| = k \). Now we construct a RBLM \( \tilde{M} = (\hat{S}, \tilde{\pi}, \{\hat{M}(\sigma)\}_{\sigma \in \Gamma}, \hat{\xi}) \) as follows:

- \( \hat{\pi} = (\pi \otimes \pi^* \otimes \rho) \), where the symbol \( \ast \) denotes conjugate operation;
- \( \hat{M}(\sigma) = \left(U(\sigma) \otimes U^*(\sigma) \otimes I\right) \left(\sum_{c \in C} P(c) \otimes P(c) \otimes M(c)\right) \);
- \( \hat{\xi} = \sum_{k=0}^{m} e_k \otimes e_k \otimes \xi \), where \( e_k \) is the column vector having 1 only at the \( k \)th component and 0's elsewhere.
Then we have (denoting $\mathcal{S}$ by $x_n+$):

$$f_{\tilde{M}}(x_1 \ldots x_n,) = \tilde{M}(x_1) \ldots \tilde{M}(x_n) \tilde{M}(\mathcal{S}) \tilde{M}(\mathcal{S})$$

$$= (\pi \otimes \pi' \otimes \rho) \prod_{i=1}^{n+1} \left( U(x_i) \cup U'(x_i) \cup I \right) \left( \sum_{c \in C} P(c) \otimes P(c) \otimes M(c) \right) \cdot \left( \sum_{k=1}^{m} e_k \otimes e_k \otimes \xi \right)$$

$$= (\pi \otimes \pi' \otimes \rho) \sum_{y=y_1 \ldots y_n \in \mathcal{C}^{n+1}} \left( \prod_{i=1}^{n+1} U(x_i)(y_i) \otimes \prod_{i=1}^{n+1} U'(x_i)(y_i) \otimes \prod_{i=1}^{n+1} M(y_i) \right) \cdot \left( \sum_{k=1}^{m} e_k \otimes e_k \otimes \xi \right)$$

$$= \sum_{y=y_1 \ldots y_n \in \mathcal{L}} \left( \prod_{k=1}^{m} U(x_i)(y_i) \right)^2 \rho M(y) \xi$$

$$= f_{\tilde{M}}(x_1 \ldots x_n,).$$

We have shown that $\mathcal{M}$ and $\tilde{M}$ have the same behavior for any word $w \in \Sigma^*$, and $\tilde{M}$ has $km^2$ states. □

**Remark 2.** One can find that in the above process, the DFA recognizing the control language $\mathcal{L}$ has no need to be minimal. In practice, when some DFA recognizing the control language is given, we can construct the RBLM by using it. However, as we can see, the minimal DFA can keep the resulting RBLM as small as possible, and then leads to a tight bound in Theorem 7 as follows.

**Theorem 7.** Two CL-1QFAs $A_1$ and $A_2$ with control languages $\mathcal{L}_1$ and $\mathcal{L}_2$, respectively, are equivalent if and only if they are $(c_1n_1^2 + c_2n_2^2 - 1)$-equivalent, where $n_1$ and $n_2$ are the numbers of states in $A_1$ and $A_2$, respectively, and $c_1$ and $c_2$ are the numbers of states in the minimal DFAs that recognize $\mathcal{L}_1$ and $\mathcal{L}_2$, respectively. Furthermore, if $\mathcal{L}_1$ and $\mathcal{L}_2$ are given in the form of DFAs, with $m_1$ and $m_2$ states, respectively, then there exists a polynomial-time algorithm running in time $O((m_1n_1^2 + m_2n_2^2)^4)$ that takes as input $A_1$ and $A_2$ and determines whether they are equivalent.

**Proof.** Suppose that CL-1QFAs $A_1$ and $A_2$ with control languages $\mathcal{L}_1$ and $\mathcal{L}_2$, respectively, have the same input alphabet $\Sigma$ and the end-marker $\mathcal{S}$, and $\mathcal{L}_1$ and $\mathcal{L}_2$ can be recognized by the minimal DFAs with $c_1$ and $c_2$ states, respectively. Now we have to determine whether $f_{A_1}(w) = f_{A_2}(w)$ holds for any $w \in \Sigma^*$. We do that in the following steps, where we firstly transform CL-1QFAs to RBLMs, then remove the end-maker $\mathcal{S}$, and lastly determine the equivalence for RBLMs.

(1) By Lemma 6, $A_1$ and $A_2$ can be simulated by two RBLMs $A_1^{(1)}$ and $A_2^{(1)}$ over the alphabet $\Sigma \cup \{\mathcal{S}\}$ with $c_1n_1^2$ and $c_2n_2^2$ states, respectively, such that $f_{A_1}(w) = f_{A_1^{(1)}}(w\mathcal{S})$ and $f_{A_2}(w) = f_{A_2^{(1)}}(w\mathcal{S})$ for any $w \in \Sigma^*$.

(2) By Proposition 1, there are two RBLMs $A_1^{(2)}$ and $A_2^{(2)}$ over the alphabet $\Sigma$, with $c_1n_1^2$ and $c_2n_2^2$ states, respectively, such that $f_{A_1^{(1)}}(w\mathcal{S}) = f_{A_1^{(2)}}(w)$ and $f_{A_2^{(1)}}(w\mathcal{S}) = f_{A_2^{(2)}}(w)$.

(3) By Definition 2 and Proposition 5, $f_{A_1^{(2)}}(w) = f_{A_2^{(2)}}(w)$ holds for any $w \in \Sigma^*$ if and only if it holds for any $w \in \Sigma^*$ with $|w| \leq c_1n_1^2 + c_2n_2^2 - 1$.

Therefore, $f_{A_1}(w) = f_{A_2}(w)$ holds for any $w \in \Sigma^*$ if and only if it holds for any $w \in \Sigma^*$ with $|w| \leq c_1n_1^2 + c_2n_2^2 - 1$.

Furthermore, if we want to design an algorithm that simulates the above steps to determine whether $A_1$ and $A_2$ are equivalent, then the consuming time will vary with the given forms of $\mathcal{L}_1$ and $\mathcal{L}_2$.

(i) $\mathcal{L}_1$ and $\mathcal{L}_2$ are given in the form of regular expressions. Then, according to the results in [19], exponential time (in the lengths of $\mathcal{L}_1$ and $\mathcal{L}_2$) will be needed to construct DFAs from $\mathcal{L}_1$ and $\mathcal{L}_2$ in step (1), and as a result, the total time will have an exponential additive factor.

(ii) $\mathcal{L}_1$ and $\mathcal{L}_2$ are given in the form of DFAs (not necessarily in minimal form), say $M_1$ and $M_2$ with $m_1$ and $m_2$ states, respectively. Recall that we have assumed that the operations of addition and multiplication on two complex numbers can all be done in constant time. Then, from the proof of Lemma 6, we can find that step (1) consumes time $O((m_1n_1^2)^2 + (m_2n_2^2)^2)$ that is mainly used on the multiplication and Kronecker product of matrices, producing two RBLMs $A_1^{(1)}$ and $A_2^{(1)}$ with $m_1n_1^2$ and $m_2n_2^2$ states, respectively. Step (2) taking as input $A_1^{(1)}$ and $A_2^{(1)}$ can be done in time $O((m_1n_1^2)^2 + (m_2n_2^2)^2)$. From Proposition 5, step (3) taking as input two RBLMs with $m_1n_1^2$ and $m_2n_2^2$ states, respectively, can be done in time $O((m_1n_1^2 + m_2n_2^2)^4)$. Therefore, the total time is $O((m_1n_1^2 + m_2n_2^2)^4)$.

Now we have proved the theorem. □

**Remark 3.** There may be a better solution to the problem of determining the equivalence between CL-1QFAs. Nevertheless, the current good news is that Theorem 7 indeed provides a bound on the length of strings that need to be verified when we want to determine the equivalence between two CL-1QFAs.
3.3. Determining the equivalence for MM-1QFAs

Gruska [17] proposed as an open problem that, is it decidable whether two MM-1QFAs are equivalent. Then Koshiba [21] tried to solve the problem. His method consists of two steps: (i) for any MM-1QFA, construct an equivalent MO-g1QFA (like MO-1QFA but with transformation matrices not necessarily unitary); (ii) determine the equivalence for MO-g1QFAs using the known way on MO-1QFAs. Now, we have known that the construction technique stated in [21] for step (i) is not valid, i.e., it produces an MO-g1QFA that is not equivalent to the original MM-1QFA. Also, this flaw has been confirmed by [22], and one may refer to [25] for a detailed explanation of this flaw. Thus, the equivalence problem for MM-1QFAs is in fact not solved there. In addition, to our knowledge, so far there seems to have been no existing valid solution to this problem. Therefore, we would like to find that solution in the following.

Now we try to determine the equivalence between MM-1QFAs, starting with a proposition introduced as follows.

Proposition 8 ([10]). Let \( U(\sigma) \) be a unitary matrix, for \( \sigma \in \Sigma \), and \( \Theta \) an observable with results in \( \mathcal{C} \), described by projectors \( P(c) \), for \( c \in \mathcal{C} \). For any complex vector \( \alpha \) and any word \( x = x_1 \ldots x_r \in \Sigma^r \), we get

\[
\sum_{y_1 \ldots y_r \in \mathcal{C}} \| \alpha \prod_{i=1}^r U(x_i)P(y_i) \|^2 = \| \alpha \|^2.
\]

Proof. Using the properties of unitary matrices and projective measurement, it is easy to prove this proposition by induction on the length of \( x \). \( \square \)

Based on [10], we get another key lemma. With this lemma, we can transform MM-1QFAs to CL-1QFAs for which the equivalence problem has been solved.

Lemma 9. Given an MM-1QFA \( M = (Q, \pi, \{ U(\sigma) \}_{\sigma \in \Sigma \cup \{$}, \Theta) \), there is a CL-1QFA \( M' = (Q, \pi, \{ U(\sigma) \}_{\sigma \in \Sigma \cup \{$}, \Theta, g^* a(a, r, g)^* \) such that for any \( w \in \Sigma^*, f_M(w) = f_M'(w) \).

Proof. Suppose that there are MM-1QFA \( M \) and CL-1QFA \( M' \) as stated above. For any \( x_1 \ldots x_n \in \Sigma^* \), there is (denoting \$ by \( x_{n+1} \)):

\[
f_M'(x_1 \ldots x_n$) = \sum_{y_1 \ldots y_{n+1} \in \mathcal{C}^* \cup \{$} \prod_{i=1}^{n+1} U(x_i)P(y_i) \|^2 = \sum_{k=0}^{n} \prod_{i=1}^{k} U(x_i)P(g)U(x_{k+1})P(a) \prod_{j=k+2}^{n+1} U(x_j)P(y_j) \|^2 = \sum_{k=0}^{n} \prod_{i=1}^{k} U(x_i)P(a) \|^2 \quad \text{(by Proposition 8)}
\]

Note that the two automata have the same states. We end the proof here. \( \square \)

Combining Theorem 7 and Lemma 9, we now obtain the following theorem determining the equivalence between two MM-1QFAs, which is actually a corollary of Theorem 7.

Theorem 10. Two MM-1QFAs \( A_1 \) and \( A_2 \) with \( n_1 \) and \( n_2 \) states, respectively, are equivalent if and only if they are \((3n_1^2 + 3n_2^2 - 1)\)-equivalent. Furthermore, there is a polynomial-time algorithm running in time \( O((3n_1^2 + 3n_2^2)^6) \) that takes as input \( A_1 \) and \( A_2 \) and determines whether \( A_1 \) and \( A_2 \) are equivalent.

Proof. Suppose that MM-1QFAs \( A_1 \) and \( A_2 \) with \( n_1 \) and \( n_2 \) states, respectively, have the same input alphabet \( \Sigma \) and the end-marker $. Now we determine whether \( f_{A_1}(w$) = \( f_{A_2}(w$) holds for any \( w \in \Sigma^* \). We can do that by the following steps.

(1) By Lemma 9, \( A_1 \) and \( A_2 \) can be transformed into two CL-1QFAs \( A^{(1)}_1 \) and \( A^{(1)}_2 \) over the working alphabet \( \Gamma = \Sigma \cup \{$ \) with \( n_1 \) and \( n_2 \) states, respectively, both of which have the same constant control language \( g^* a(a, r, g)^* \).

(2) By Lemma 6, \( A^{(1)}_1 \) and \( A^{(1)}_2 \) can be transformed into two RBLMs \( A^{(2)}_1 \) and \( A^{(2)}_2 \) over \( \Gamma' \), with \( n_1^2 \) and \( n_2^2 \) states, respectively, where the factor 3 is the number of states in the DFA (described in Fig. 1) recognizing the control language \( g^* a(a, r, g)^* \).

(3) By Proposition 1, we can construct \( A^{(3)}_1 \) and \( A^{(3)}_2 \) over the alphabet \( \Sigma \) from \( A^{(2)}_1 \) and \( A^{(2)}_2 \), such that \( f_{A_1}(w$) = \( f_{A_2}(w$) \) and \( f_{A_1}(w$) = \( f_{A_2}(w$) \) for any \( w \in \Sigma^* \). Therefore, determining whether \( f_{A_1}(w$) = \( f_{A_2}(w$) holds for any \( w \in \Sigma^* \) is equivalent to determining whether \( A^{(3)}_1 \) and \( A^{(3)}_2 \) are equivalent.

(4) By Proposition 5, \( A^{(3)}_1 \) and \( A^{(3)}_2 \) are equivalent if and only if they are \((3n_1^2 + 3n_2^2 - 1)\)-equivalent.

Therefore, \( f_{A_1}(w$) = \( f_{A_2}(w$) holds for any \( w \in \Sigma^* \) if and only if it holds for any \( w \in \Sigma^* \) with \( |w| \leq 3n_1^2 + 3n_2^2 - 1 \). Furthermore, it is readily seen that step (1) can be done in constant time, and the other steps can be done in time \( O((3n_1^2 + 3n_2^2)^6) \) from the proof of Theorem 7. Therefore, there exists a polynomial-time algorithm simulating the above steps to determine whether two MM-1QFAs are equivalent. Hence, we have completed the proof. \( \square \)
4. Conclusions

QFAs are simple but basic models of quantum computation, but the decidability problem for equivalence between QFAs has not been solved completely. In this paper, we considered the decision of equivalence for CL-1QFAs and MM-1QFAs. Specifically, we have shown that two CL-1QFAs $A_1$ and $A_2$ with control languages (regular languages) $L_1$ and $L_2$, respectively, are equivalent if and only if they are $(c_1 n_1^2 + c_2 n_2^2 - 1)$-equivalent, where $n_1$ and $n_2$ are the numbers of states in $A_1$ and $A_2$, respectively, and $c_1$ and $c_2$ are the numbers of states in the minimal DFAs that recognize $L_1$ and $L_2$, respectively. Furthermore, given $L_1$ and $L_2$ in the form of DFAs, with $n_1$ and $n_2$ states, respectively, a polynomial-time algorithm was given, that determines whether $A_1$ and $A_2$ are equivalent in time $O((m n_1^2 + m n_2^2)^4)$.

On the other hand, we corrected Koshiba’s argument [21] for the equivalence problem of MM-1QFAs. In particular, we showed that two MM-1QFAs $A_1$ and $A_2$ with $n_1$ and $n_2$ states, respectively, are equivalent if, and only if they are $(3n_1^2 + 3n_2^2 - 1)$-equivalent. Also, a polynomial-time algorithm was presented, that determines whether $A_1$ and $A_2$ are equivalent in time $O(3n_1^2 + 3n_2^2)^4)$. Thus, the problem proposed by Gruska [17] has been addressed.

So far, the equivalence issues for MO-1QFAs, MM-1QFAs, and CL-1QFAs have been addressed. However, the equivalence concerning another important model—2QFAs [23] is still open and worthy of further consideration.

Acknowledgements

The authors are grateful to the referees and the editor for their invaluable comments and suggestions that greatly helped in improving the quality of this paper. Also, they would like to thank Dr. Koshiba for valuable communication.

It is worth mentioning that one of the anonymous referees pointed out a different idea to deal with the equivalence problems stated in this paper, that is, to write the configuration of quantum states by the mixed state notation, instead in improving the quality of this paper. Also, they would like to thank Dr. Koshiba for valuable communication.

References

[22] T. Koshiba, Personal communication.