Classically Time-Controlled Quantum Automata*

Alejandro Díaz-Caro[1,2] and Marcos Villagra[3]

[1] Departamento de Ciencia y Tecnología
Universidad Nacional de Quilmes
Roque Sáenz Peña 352, B1876BXD, Bernal, Buenos Aires, Argentina
[2] Instituto de Ciencias de la Computación
CONICET-Universidad de Buenos Aires
Pabellón 1, Ciudad Universitaria, C1428EGA Buenos Aires, Argentina
adiazcaro@icc.fcen.uba.ar
[3] Núcleo de Investigación y Desarrollo Tecnológico
Universidad Nacional de Asunción
San Lorenzo C.P. 2169, Asunción, Paraguay
mvillagra@pol.una.py

Abstract. In this paper we introduce classically time-controlled quantum automata or CTQA, which is a slight but reasonable modification of Moore-Crutchfield quantum finite automata that uses time-dependent evolution operators and a scheduler defining how long each operator will run. Surprisingly enough, time-dependent evolutions provide a significant change in the computational power of quantum automata with respect to a discrete quantum model. Furthermore, CTQA presents itself as a new model of computation that provides a different approach to a formal study of "classical control, quantum data" schemes in quantum computing.

Keywords: quantum computing, quantum finite automata, time-dependent unitary evolution, bounded error, cutpoint language

1 Introduction

A well-known hardware model for a future design of quantum computers is the QRAM model proposed by Knill [6]. The idea is that a quantum device will be attached to a classical computer controlling all operations. Several programming languages have been designed and studied using this model (e.g. [4, 5, 9, 12, 13]) where the classical part constructs the circuit and the quantum part manipulates the quantum state. This scheme is the so-called "classical control, quantum-data."

To understand the capabilities and limitations of quantum computers with classical control it is interesting to conceptualize a formal model of quantum computations that incorporates in some way the idea of a classical control. The most

* A. Díaz-Caro is supported by projects PICT 2015-1208, PUNQ 1370/17, and ECOS-Sud A17C03 QuCa. M. Villagra is supported by Conacyt research grant PINV15-208. Part of this work was done while A. Díaz-Caro was visiting Universidad Nacional de Asunción in July 2018 under Conacyt research grant PVCT18-102.

simple model of computation currently known is the finite-state automaton, and it is, arguably, the best model to initiate a study of new methods of computation.

The first model of a quantum automaton with classical control was studied by Ambainis and Watrous [2] and consisted in a two-way quantum automaton with quantum and classical inner states, with the addition that the input tape head is also classical. Ambainis and Watrous showed that for this model of quantum automata there exists a non-regular language that can be recognized in expected polynomial time, whereas for the same language any two-way probabilistic automaton requires expected exponential time. Another way to introduce classical components in quantum computations is in the context of quantum interactive proof systems (QIP) with quantum automata as verifiers [8, 11, 14, 17]. These works showed that having a quantum automaton interacting with a prover that can be quantum or classical does indeed help the automaton to recognize more languages.

In all cited works of the previous paragraph, the classical control is implemented via discrete circuits, that is, a "program" decides what gates to apply to which qubits. However, a quantum computer could do more than just apply discrete matrices. Indeed, the Schrödinger equation, which is the equation governing the time-evolution of all quantum systems, is defined over a continuous time, and whose solutions are time-dependent unitary operators.

In this work we present a new type of classical control where all unitary operators of a quantum automaton depend on time, and their time-evolutions can be adjusted or tuned in order to assist the automaton in its computations. In order to control the time of each unitary operator we introduce an idea of a *scheduler* that feeds the automaton with a *time schedule* specifying for how long each unitary operation must be executed. We call this model *classically time-controlled quantum automata* or CTQA.

The automaton model used for CTQAs is the so-called "measure-once" or "Moore-Crutchfield" quantum automaton [7], where only one measurement is allowed at the end of any computation. Brodsky and Pippenger [3] proved that Moore-Crutchfield quantum automata are equivalent in computational power to classical permutation automata, which is a much weaker and restricted model of a deterministic finite-state automaton. The class of languages recognized by Moore-Crutchfield automata includes only regular languages and there are many natural regular languages that do not belong to this class. For example, the languages $\mathtt{L}_{ab} = \{a^n b^m \mid n, m \geq 0\}$ and $\mathtt{L}_1 = \{x1 \mid x \in \{0,1\}^*\}$ are not recognized by any permutation automaton or Moore-Crutchfield quantum automaton. In this work we show that even though a CTQA uses a quite restricted model of quantum automata, when time evolutions of unitary operators can be controlled by an external classical scheduler, more languages can be recognized. In fact, we show that non-recursive languages are recognized by CTQAs if we allow unrestricted time schedules (Theorem 2). Since arbitrary time schedules give extreme computational power to a quantum automaton, we study the language recognition power of CTQAs when assisted by computationally restricted schedulers. When the scheduler is implemented via a deterministic finite-state

automaton we show that CTQAs can recognize all regular languages (Theorem 3) and even non-regular languages (Theorem 4). We also show the existence of two languages recognized by CTQAs that can be concatenated as long each CTQA uses "similar" schedulers and different alphabets (Theorem 10).

The rest of this paper is organized as follows. In Section 2 we introduce the notation used throughout this paper and briefly review some relevant results from quantum automata theory. In Section 3 we present a formal definition of CTQAs together with some basic properties. In Section 4 we present our results about restricted time schedules. Finally, in Section 5 we conclude this paper and present some open problems.

2 Preliminaries

In this section we briefly explain the notation used in the rest of this work and review some well-known definitions and results on quantum automata.

We use $\mathbb{R}$ to denote the set of real numbers and $\mathbb{C}$ the set of complex numbers. The set of all nonnegative real numbers is denoted $\mathbb{R}_0^+$. The set of natural numbers including 0 is denoted $\mathbb{N}$.

Given any finite set A, we let $\mathbb{C}^A$ be the Hilbert space generated by the finite basis A. Vectors from $\mathbb{C}^A$ are denoted using the *ket* notation $|v\rangle$. An element of the dual space of $\mathbb{C}^A$ is denoted using the *bra* notation $\langle v|$. The inner product between two vectors $|v\rangle$ and $|u\rangle$ is denoted $\langle v|u\rangle$.

Let Σ be a finite alphabet and let Σ^* denote the set of all strings of finite length over Σ. A string $x \in \Sigma^*$ of length n can be written as $x = x_1 \dots x_n$ where each $x_i \in \Sigma$. The length of x is denoted $|x|$. A language $\mathtt{L}$ is a subset of Σ^*. The concatenation of two languages $\mathtt{L}_1$ and $\mathtt{L}_2$ is denoted $\mathtt{L}_1 \cdot \mathtt{L}_2$. We also let $\mathtt{L}^* = \cup_{k\in\mathbb{N}} \mathtt{L}^k$ where $\mathtt{L}^k$ is the language $\mathtt{L}$ concatenated with itself k times.

A *quantum finite automaton* (or QFA) is a 5-tuple $M = (Q, \Sigma, \{\xi_\sigma \mid \sigma \in \Sigma\}, s, A)$ where Q is a finite set of inner states, ξ_σ is a transition superoperator[1] for a symbol $\sigma \in \Sigma$, the initial inner state is $s \in Q$, and $A \subseteq Q$ is a set of accepting states. On input $x \in \Sigma^*$, a computation of M on $x = x_1 \dots x_n$ is given by $\rho_j = \xi_{x_j}(\rho_{j-1})$, where $\rho_0 = |s\rangle\langle s|$ and $1 \leq j \leq |x|$. The most restricted model of QFA currently known is the so-called *Moore-Crutchfield QFA* or MCQFA [7]. A MCQFA is a 5-tuple $M = (Q, \Sigma, \delta, s, A)$, where all components are defined exactly in the same way as for QFAs except that the transition function $\delta : Q \times \Sigma \times Q \to \mathbb{C}$ defines a collection of unitary matrices $\{U_\sigma \mid \sigma \in \Sigma\}$ where U_σ has $\delta(q, \sigma, p)$ in the (p, q)-entry and each U_σ acts on $\mathbb{C}^Q$. Physically M corresponds to a closed-system based on pure states.[2] For any given input w, the machine M is initialized in the quantum state $|\psi_0\rangle = |s\rangle$ and each step of a computation is given by $|\psi_j\rangle = U_{w_j}|\psi_{j-1}\rangle$, where $1 \leq j \leq |w|$. The probability that M accepts w is $p_{A,M}(w) = \sum_{q_j \in A} |\langle q_j|\psi_{|w|}\rangle|^2$. This is

[1] A superoperator or quantum operator is a positive-semidefinite operation that maps density matrices to density matrices [10].

[2] Pure states are vectors in a complex Hilbert space normalized with respect to the ℓ_2-norm.

equivalent to M performing a single measurement of its quantum state at the end of a computation. The class of languages recognized by MCQFAs with bounded-error is denoted **MCQFA**. Brodsky and Pippenger [3] showed using a non-constructive argument that **MCQFA** coincides with the class of languages recognized by permutation automata; see Villagra and Yamakami [15] for a constructive argument of the same result. Ambainis and Freivalds [1] studied quantum automata with pure states where measurements are allowed at each step of a computation. We denote by **1QFA** the class of languages recognized by quantum automata with pure states and with many measurements allowed. Ambainis and Freivalds [1] showed that **MCQFA** $\subsetneq$ **1QFA** by proving that the language $\mathtt{L}_{ab} = \{a\}^* \cdot \{b\}^* \notin$ **MCQFA**. The class of regular languages is denoted **REG** and it is known that **1QFA** $\subsetneq$ **REG** [1].

3 Definition and Basic Properties

A *classically time-controlled quantum automaton* (CTQA in short) is defined as $(Q, \Sigma, \delta, \tau, s, A, R)$, where Q is a finite set of inner states, Σ is an alphabet, $\delta : Q \times \Sigma \times Q \times \mathbb{R}^+ \to \mathbb{C}$ is a transition function, $\tau : \Sigma^* \to (\mathbb{R}^+)^*$ is a function called *time schedule* function, s is an initial inner state, $A \subseteq Q$ is the set of accepting inner states, and $R \subseteq Q$ is the set of rejecting inner states.

A CTQA has a single tape split into two tracks, where an upper track contains the original input x and a lower track contains a time schedule string $\tau(x) = (\tau_1, \ldots, \tau_{|x|})$ where each $\tau_i \in \mathbb{R}_0^+$.

Given an input x and a time schedule τ, the operation of the automaton is as follows. Given any positive real number t, for each $\sigma \in \Sigma$ we have

$$U_\sigma(t)|q\rangle = \sum_{p \in Q} \delta(q, \sigma, p, t)|p\rangle,$$

where $U_\sigma(t)$ is a unitary time-dependent evolution operator. Given an input x of length n, the time schedule string maps x to a sequence of $|x|$ positive real numbers $\tau(x) = (\tau_1 \ldots \tau_n)$ where each τ_i indicates for how much time the unitary operator U_{x_i} must be executed.

A CTQA M starts in the quantum state $|s\rangle$ corresponding to the inner state s and its tape is of the form $\left[\begin{smallmatrix} x \\ \tau(x) \end{smallmatrix}\right]$, where $\left[\begin{smallmatrix} x \\ \tau(x) \end{smallmatrix}\right] = \left[\begin{smallmatrix} x_1 \ldots x_n \\ \tau_1 \ldots \tau_n \end{smallmatrix}\right]$ is a track notation that denotes the contents of the two tracks of the tape, the input x and the time schedule $\tau(x)$. At step i if the machine M is in the quantum state $|\psi_{i-1}\rangle$ and scanning $\left[\begin{smallmatrix} x_i \\ \tau_i \end{smallmatrix}\right]$, then the next quantum state $|\psi_i\rangle$ is given by

$$|\psi_i\rangle = U_{x_i}(\tau_i)|\psi_{i-1}\rangle.$$

After scanning an entire input the machine M observes the quantum state $|\psi_n\rangle$ with respect to the subspaces $span(A) = \mathbb{C}^A$, $span(R) = \mathbb{C}^R$ and $span(Q \setminus (A \cup R)) = \mathbb{C}^{Q \setminus (A \cup R)}$. If we observe a quantum state in $span(A)$, we say that x is accepted by M. Similarly, if we observe a quantum state in $span(R)$, x is rejected by M; otherwise, M answers " I do not know."

Let Π_A be a projection onto the subspace $span(A)$ and let

$$|\psi_n\rangle = U_{x_n}(\tau_n)\cdots U_{x_1}(\tau_1)|s\rangle.$$

The probability that M accepts x is defined as

$$p_{M,A}(x) = \langle\psi_n|\Pi_A|\psi_n\rangle.$$

Similarly, if Π_R is a projection onto the subspace $span(R)$, the probability that M rejects x is

$$p_{M,R}(x) = \langle\psi_n|\Pi_R|\psi_n\rangle.$$

Let $\lambda \in (0,1]$. A language $\mathtt{L}$ is said to be *recognized* or *accepted* by M with cutpoint λ if

$$\mathtt{L} = \{x \in \Sigma^* \mid p_{M,A}(x) \geq \lambda\}.$$

A CTQA A is *time-independent* if and only if for any given $(q,\sigma,p) \in Q\times\Sigma\times Q$ it holds that $\delta(q,\sigma,p,t) = \delta(q,\sigma,p,t')$ for all t,t'. Thus, if A is time-independent, then $U_\sigma(t) = U_\sigma(t')$ for all $\sigma \in \Sigma$ and $t,t' \in \mathbb{R}_0^+$.

The class of languages recognized by CTQA with cutpoint λ is denoted $\mathbf{CTQ}_\lambda$. The class of languages recognized by time-independent CTQA with cutpoint λ is denoted $\mathbf{t\text{-}CTQ}_\lambda$.

A language $\mathtt{L}$ is said to be recognized by M with isolated cutpoint λ if there exists a positive real number α such that $p_{M,A}(x) \geq \lambda + \alpha$ for all $x \in \mathtt{L}$ and $P_{M,R}(x) \leq \lambda - \alpha$ for all $x \notin \mathtt{L}$. Language recognition with isolated cutpoint is easily described as recognition with bounded-error. Let $\epsilon \in [0, \frac{1}{2})$. We say that $\mathtt{L}$ is recognized with bounded-error by M with error bound ϵ if $p_{M,A}(x) \geq 1-\epsilon$ for all $x \in \mathtt{L}$ and $p_{M,R}(x) \leq \epsilon$ for all $x \notin \mathtt{L}$. The class of languages recognized by CTQA with bounded-error in the time-dependent and time-independent cases are denoted **BCTQ** and **t-BCTQ**, respectively.

Theorem 1. $\mathbf{t\text{-}BCTQ} = \mathbf{MCQFA}$.

Proof. Let $A = (Q,\Sigma,\delta,q_0,A,R)$ be a time-independent CTQA. Take $B = (Q,\Sigma,\delta',q_0,A,R)$ where $\delta'(q,\sigma,p) = \delta(q,\sigma,p,1)$. To see the other side of the implication it suffices to see that δ' is time-independent and thus any time schedule works. □

This first naïve definition allowing any arbitrary time schedule, allows arbitrary power to CTQA, as exemplified by the following theorem.

Theorem 2. *If the time schedule is not restricted, there exists a bounded-error CTQA deciding the Halting problem with* $\epsilon = 0$.

Proof. Let $\mathtt{HALT}$ be the language denoting the halting problem, that is, a string $x \in \mathtt{HALT}$ if and only x is a reasonable encoding using an alphabet Σ of a Turing machine N and a string w such that M halts on input w. We construct a CTQA $M = (Q,\Sigma,\delta,\tau,s,A,R)$ recognizing $\mathtt{HALT}$.

Let τ be a time schedule such that if an input x of M is the encoding of a Turing machine N and an input w for N, then $\tau(x) = (1,0,0,\ldots,0)$ if N

does not halt on input w; otherwise, $\tau(x) = (4, 0, 0, \ldots, 0)$ if M halts on input w. Then, define $Q = \{q_0, q_1\}$, $s = q_0$, $A = \{q_0\}$, and $R = \{q_1\}$. The transition function δ is defined as

$$\delta(0, \sigma, 0, t) = \delta(1, \sigma, 1, t) = \cos\left(t \cdot \frac{\pi}{2}\right),$$
$$\delta(0, \sigma, 1, t) = \delta(1, \sigma, 0, t) = -i \sin\left(t \cdot \frac{\pi}{2}\right).$$

That is, the time-dependent unitary operator U_σ acts on $span(Q)$ as given by

$$U_\sigma(t)|q\rangle = \delta(q, \sigma, 0, t)|q_0\rangle + \delta(q, \sigma, 1, t)|q_1\rangle,$$

and hence,

$$U(t) = R_{t\pi} = \begin{pmatrix} \cos(t\frac{\pi}{2}) & -i\sin(t\frac{\pi}{2}) \\ -i\sin(t\frac{\pi}{2}) & \cos(t\frac{\pi}{2}) \end{pmatrix}.$$

where $R_{t\pi}$ denotes a $t\pi$ degrees rotation about the x-axis of the Bloch sphere (cf. Figure 1). Note that $U(0) = I_2$, $U(1) = i\mathsf{Not}$ and $U(4) = I_2$, where Not is the quantum negation operator.

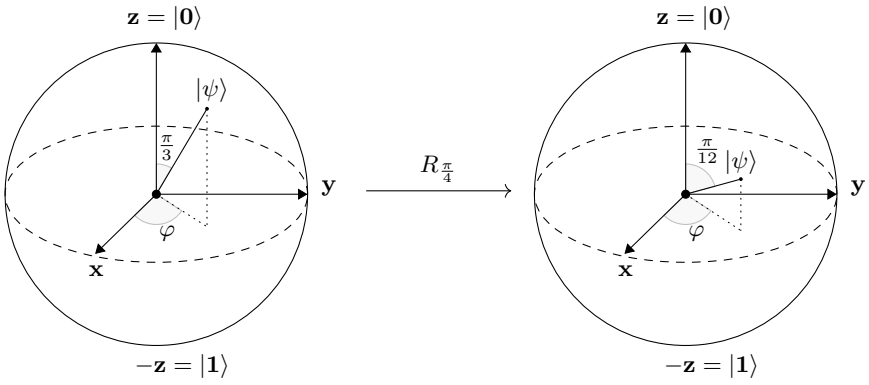

Fig. 1. $U(0.25) = R_{\frac{\pi}{4}}$ rotation

Therefore, if the input represents a halting Turing machine, the computation will be $I_2|q_0\rangle = |q_0\rangle$ and the accepting state $|q_0\rangle$ is observed with probability 1. If the input is a non-halting Turing machine, then a computation is $i\mathsf{Not}|q_0\rangle = i|q_1\rangle$ and the rejecting state $|q_1\rangle$ is observed with probability 1. □

The previous theorem shows that the expressive power of a time schedule can be easily passed to a CTQA. Hence, in order uncover the capabilities of CTQAs we will introduce a machine called *scheduler* that takes care of computing a time schedule.

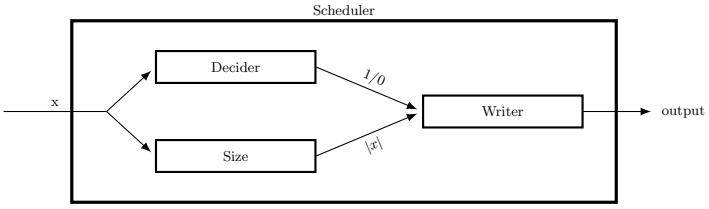

Fig. 2. Scheduler diagram

4 Language Recognition with Restricted Time Schedules

A *scheduler* S is defined as a pair (D, W) where D is a multitape Turing machine that halts on all inputs called a *decider* and W is a multi-valued function called a *writer*. Besides the decider and writer, the scheduler S includes the capability of counting the size on an input. On input x an scheduler S works as depicted in Figure 2: First S runs D on input x and outputs a bit b where $b = 1$ if x is accepted by D or $b = 0$ if x is rejected by D. Then S runs the writer W on input b and $n = |x|$. For some constant positive integer k, the writer W is defined using two sets of functions $\mathcal{F} = \{f_1, \ldots, f_k\}$ and $\mathcal{G}_1 = \{g_1, \ldots, g_k\}$ where for each i, $f_i : \mathbb{N} \to \mathbb{R}_0^+$ and $g_i : \mathbb{N} \to \mathbb{R}_0^+$. The writer W on input b and $n = |x|$ generates as an output a time schedule $(f_{i_1}(n), \ldots, f_{i_n}(n))$ if $b = 1$ or $(g_{i_1}(n), \ldots, g_{i_n}(n))$ if $b = 0$, where each $i_j \in \{1, 2, \ldots, k\}$.

Let $\mathbf{C}$ be a complexity class. We denote by $\mathbf{C\text{-}CTQ}_\lambda$ the class of languages recognized by CTQA with cutpoint λ where the computational power of the decider in the scheduler is restricted to $\mathbf{C}$. In particular, $\mathbf{REG\text{-}CTQ}_\lambda$ is the class of languages recognized by CTQAs with cutpoint λ where the decider D is a finite-state automaton. When a CTQA is bounded-error we write $\mathbf{C\text{-}BCTQ}$.

It is clear that a CTQA has, at least, as much computational power as the decider in its scheduler, as stated in Theorem 3 below. Later we will show that even if a scheduler is computationally restricted, a CTQA can recognize more languages than what is allowed by its scheduler.

Theorem 3. $\mathbf{C} \subseteq \mathbf{C\text{-}BCTQA}$.

Proof. We can consider the same CTQA from the proof of Theorem 2. Take a decider D recognizing a language $\mathtt{L} \in \mathbf{C}$. Then, we consider the scheduler $S = (D, W)$ where $W(0, n) = (1, 0, \ldots, 0)$ and $W(1, n) = (4, 0, \ldots, 0)$. □

Let $\mathtt{L}_{ab}^\lambda = \{a^n b^m \mid \cos^2(\frac{\pi(n-m)}{2(n+m)}) \geq \lambda\}$. Using a pumping argument, it is easy to prove that $\mathtt{L}_{ab}^\lambda$ is not a regular language. The following theorem shows that even in the presence of a finite-state scheduler, that is the decider of the scheduler is a finite-state automaton, there exists a CTQA recognizing $\mathtt{L}_{ab}^\lambda$ with cutpoint λ.

Theorem 4. $L_{ab}^\lambda \in \mathbf{REG\text{-}CTQA}_\lambda$.

Proof. Let $M = (Q, \Sigma, \delta, \tau, s, A, R)$ where $Q = \{q_0, q_1\}$, $\Sigma = \{a, b\}$, $s = q_0$, $A = \{q_0\}$, and $R = \{q_1\}$. The transition function δ is defined as

$$\begin{aligned}
\delta(q_0, a, q_0, t) = \delta(q_1, a, q_1, t) &= \cos\left(t\frac{\pi}{2}\right), \\
\delta(q_0, a, q_1, t) = \delta(q_1, a, q_0, t) &= -i\sin\left(t\frac{\pi}{2}\right), \\
\delta(q_0, b, q_0, t) = \delta(q_1, b, q_1, t) &= \cos\left(-t\frac{\pi}{2}\right), \\
\delta(q_0, b, q_1, t) = \delta(q_1, b, q_0, t) &= -i\sin\left(-t\frac{\pi}{2}\right).
\end{aligned}$$

The transition function δ thus defines unitary operators U_a and U_b acting on $span(Q)$ as

$$\begin{aligned}
U_a(t)|q\rangle &= \delta(q, a, q_0, t)|q_0\rangle + \delta(q, a, q_1, t)|q_1\rangle, \\
U_b(t)|q\rangle &= \delta(q, b, q_0, t)|q_0\rangle + \delta(q, b, q_1, t)|q_1\rangle.
\end{aligned}$$

where $U_a(t) = U(t)$ and $U_b(t) = U^{-1}(t) = U(-t)$ with

$$U(t) = R_{t\pi} = \begin{pmatrix} \cos(t\frac{\pi}{2}) & -i\sin(t\frac{\pi}{2}) \\ -i\sin(t\frac{\pi}{2}) & \cos(t\frac{\pi}{2}) \end{pmatrix}.$$

The intuition is that $U_a(0)$ is the identity, $U_a(1)$ is a Not operator whereas $U_a(t)$ is a unitary operation between the identity and the Not operator for $t \in (0, 1)$. On the other hand, $U_b(t)$ is a rotation in the opposite direction.

We define the scheduler S computing τ as $S = (D, W)$ where

- D is a finite state decider recognizing the regular language $\mathtt{L}_{ab} = \{a\}^* \cdot \{b\}^*$ such that D outputs $b = 1$ for all strings in $\mathtt{L}_{ab}$ and $b = 0$ otherwise, and
- W is a writer given by

$$W(n+m, b) = \begin{cases} (\frac{1}{n+m}, \frac{1}{n+m}, \ldots, \frac{1}{n+m}) & \text{if } b = 1 \\ (1, 0, \ldots, 0) & \text{if } b = 0. \end{cases}$$

Suppose $x \notin \mathtt{L}_{ab}$. The scheduler runs D on x which rejects and the writer outputs $(1, 0, \ldots, 0)$ as a time schedule for M. The first unitary operator that is applied is either $U_a(1)$ or $U_b(1)$ which is a Not operator, and for each remaining 0 in the time schedule all unitary operators behave as the identity. The machine M will apply Not on $|0\rangle$, obtaining $|1\rangle$ and then it stays in $|1\rangle$. After scanning the entire input, M measures its quantum state and observes $|1\rangle$, thus, rejecting x.

Now suppose $x \in \mathtt{L}_{ab}$ and let $x = a^n b^m$. The scheduler runs D on x which this time accepts, and the writer outputs $(\frac{1}{n+m}, \frac{1}{n+m}, \ldots, \frac{1}{n+m})$. The unitary operators that M uses are $U_a^n(\frac{1}{n+m}) = U(\frac{n}{n+m})$ and $U_b^m(\frac{1}{n+m}) = U(-\frac{m}{n+m})$. After scanning the entire input, the quantum state of M is

$$U\left(-\frac{m}{n+m}\right) U\left(\frac{n}{n+m}\right) |q_0\rangle = \cos\left(\frac{\pi(n-m)}{2(n+m)}\right) |q_0\rangle + i\sin\left(\frac{\pi(n-m)}{2(n+m)}\right) |q_1\rangle.$$

Hence, the probability of accepting $a^n b^m$ is $\cos^2(\frac{\pi(n-m)}{2(n+m)})$, which is greater or equal than λ. Notice that the accepting probability is 1 if $n = m$ and 0 if $n = 0$ or $m = 0$. □

Corollary 5. $\mathbf{REG\text{-}CTQA}_\lambda \not\subseteq \mathbf{REG}$. □

Let $\mathtt{L}_1 = \{w \cdot 1 \mid w \in \{0,1\}^*\}$. The language $\mathtt{L}_1$ is a regular language that is not recognized by any 1QFA [1]. This language can be recognized by a $\mathbf{CTQ}_\lambda$ with a decider restricted to a constant function. Let $\mathbf{\Sigma^*\text{-}CTQ}_\lambda$ be the class of languages recognized by CTQAs with cutpoint λ where the decider accepts any string over the alphabet Σ. Note that when a decider computes a constant function, the output of the scheduler only depends of the length of the input string. This situation is similar to quantum automata assisted by advice as studied in [15, 16].

Theorem 6. $L_1 \in \mathbf{\Sigma^*\text{-}CTQ}_1$.

Proof. Let $M = (Q, \Sigma, \delta, \tau, s, A, R)$, where $Q = \{q_0, q_1\}$, $\Sigma = \{0,1\}$, $s = q_0$, $A = \{q_0\}$, and $R = \{q_1\}$. The transition function δ is defined such that $U_0(t) = R_{(1-t)\pi}$ and $U_1(t) = R_{t\pi}$, so $U_0(1) = I_2$ and $U_1(1) = \mathsf{Not}$.

The decider of the scheduler is defined by $D(x) = 1$ for any $x \in \{0,1\}^*$, and the writer is defined by

$$W(n,b) = \begin{cases} (0,0,\ldots,0,1) & \text{if } b = 1 \\ \\ (0,0,\ldots,0,0) & \text{if } b = 0 \end{cases}$$

Notice that since the decider is the constant function 1, the scheduler will always output a time schedule with $n-1$ zeroes and a single one in the last position. Therefore, the automaton M will do nothing with the $n-1$ first symbols, and it will apply $U_0(1)|0\rangle = |0\rangle$ if the last symbol is 0 rejecting the input, or $U_1(1)|0\rangle = |1\rangle$ if the last symbol is 1 accepting the input. □

Corollary 7. $\mathbf{\Sigma^*\text{-}CTQ}_1 \not\subseteq \mathbf{1QFA}$. □

Restricting the decider to a constant function accepting any input, we can still recognize a non-regular language, as stated by the following theorem. Let $\mathtt{L}^\lambda_{a\sim b} = \{x \mid |x|_a = n, |x|_b = m, \cos^2(\frac{\pi(n-m)}{2(n+m)}) \geq \lambda\}$. Using a pumping argument it can be proved that $\mathtt{L}^\lambda_{a\sim b}$ is not regular.

Theorem 8. $L^\lambda_{a\sim b} \in \mathbf{\Sigma^*\text{-}CTQ}_\lambda$.

Proof. It suffices to construct an automaton M' similar to the automaton M from the proof of Theorem 4 with a decider D' defined by $D'(x) = 1$, for any $x \in \{a,b\}^*$. Indeed, on input $x \in \mathtt{L}^\lambda_{a\sim b}$ the machine M' will execute n times $U(\frac{1}{n+m})$ and m times $U(-\frac{1}{n+m})$, in any order, producing the quantum sate

$$\cos\left(\frac{\pi(n-m)}{2(n+m)}\right)|0\rangle + i\sin\left(\frac{\pi(n-m)}{2(n+m)}\right)|1\rangle.$$

The probability of accepting a string in $\mathtt{L}^{\lambda}_{a\sim b}$ is $\cos^2(\frac{\pi(n-m)}{2(n+m)})$ which is at least λ. If $x \notin \mathtt{L}^{\lambda}_{a\sim b}$, then the probability of accepting x is less than λ. Notice that such probability is 1 if $n = m$ and 0 if $n = 0$ or $m = 0$. □

Corollary 9. $\mathbf{\Sigma^*\text{-CTQ}_1} \not\subseteq \mathbf{REG}$. □

If two automata recognizing languages on different alphabets have the same writer, then we can easily construct a new automaton recognizing the concatenation of both languages. As an example, consider the following language. Let $\mathtt{L}^{\lambda_1,\lambda_2}_{ab\cdot c\sim d} = L^{\lambda_1}_{ab} \cdot L^{\lambda_2}_{c\sim d}$ where $L^{\lambda_1}_{ab}$ and $L^{\lambda_2}_{c\sim d}$ are defined as before but over alphabets $\{a, b\}$ and $\{c, d\}$, respectively.

Theorem 10. $L^{\lambda_1,\lambda_2}_{ab\cdot c\sim d} \in \mathbf{REG\text{-}CTQA}_{\lambda_1\cdot\lambda_2}$.

Proof. Let $M = (Q, \Sigma, \delta, \tau, s, A, R)$ where $Q = \{00, 01, 10, 11\}$, $\Sigma = \{a, b, c, d\}$, $s = 00$, $A = \{00\}$, and $R = \{01, 10, 11\}$. The transition function δ is defined by

$$U_a(t) = U_b(-t) = I_2 \otimes R_{t\pi} = \begin{pmatrix} \cos(t\frac{\pi}{2}) & -i\sin(t\frac{\pi}{2}) & 0 & 0 \\ -i\sin(t\frac{\pi}{2}) & \cos(t\frac{\pi}{2}) & 0 & 0 \\ 0 & 0 & \cos(t\frac{\pi}{2}) & -i\sin(t\frac{\pi}{2}) \\ 0 & 0 & -i\sin(t\frac{\pi}{2}) & \cos(t\frac{\pi}{2}) \end{pmatrix},$$

$$U_c(t) = U_d(-t) = R_{t\pi} \otimes I_2 = \begin{pmatrix} \cos(t\frac{\pi}{2}) & 0 & -i\sin(t\frac{\pi}{2}) & 0 \\ 0 & \cos(t\frac{\pi}{2}) & 0 & -i\sin(t\frac{\pi}{2}) \\ -i\sin(t\frac{\pi}{2}) & 0 & \cos(t\frac{\pi}{2}) & 0 \\ 0 & -i\sin(t\frac{\pi}{2}) & 0 & \cos(t\frac{\pi}{2})) \end{pmatrix}.$$

The intuition is that $U_a(0)$ is the identity, $U_a(1)$ is $I_2 \otimes \mathsf{Not}$ whereas $U_a(t)$, with $t \in (0, 1)$ is a unitary operator between the identity and $I_2 \otimes \mathsf{Not}$. Furthermore, $U_b(t)$ is a rotation in the opposite direction of U_a. Similarly, $U_c(0)$ is the identity, $U_c(1)$ is $\mathsf{Not} \otimes I_2$ and $U_d(t)$ is a rotation in the opposite direction of U_c.

A scheduler S for τ is given by (D, W) where

- D is a finite state decider recognizing $\mathtt{L}_{abcd} = \{a\}^*\{b\}^*\{c, d\}^*$ such that $D(x) = 1$ if $x \in \mathtt{L}_{abcd}$ and $D(x) = 0$ for $x \notin \mathtt{L}_{abcd}$, and
- W is a writer defined by

$$W(n, b) = \begin{cases} (\frac{1}{n+m+k+h}, \frac{1}{n+m+k+h}, \ldots, \frac{1}{n+m+k+h}) & \text{if } b = 1 \\ (1, 0, \ldots, 0) & \text{if } b = 0 \end{cases}$$

Therefore, the decider is a concatenation of the decider of the automaton defined in Theorems 4 and 6.

Suppose $x \notin \mathtt{L}_{abcd}$. The scheduler outputs $(1, 0, \ldots, 0)$ as a time schedule and M changes the state $|00\rangle$ to $|01\rangle$ using $U_a(1)$ or $U_b(1)$, or M changes the state $|00\rangle$ to $|10\rangle$ using $U_c(1)$ or $U_d(-1)$. In either case, after M reads the first symbol, it stays in the same quantum state, and after scanning the entire input a rejecting state is observed with probability 1.

Now suppose $x \in \mathtt{L}_{abcd}$. The scheduler outputs $(\frac{1}{n+m+k+h}, \ldots, \frac{1}{n+m+k+h})$ as the time schedule and the unitary operators used by M are $U_a(\frac{n}{n+m+k+h})$, $U_b(\frac{m}{n+m+k+h})$, $U_c(\frac{k}{n+m+k+h})$ and $U_d(\frac{h}{n+m+k+h})$. Note that $U_b(\frac{m}{n+m+k+h}) = U_a(-\frac{m}{n+m+k+h})$ and $U_d(\frac{h}{n+m+k+h}) = U_c(-\frac{h}{n+m+k+h})$. The resulting quantum state after M scans x starting a computation at $|00\rangle$ is then

$$\begin{aligned}
&\cos\left(\frac{\pi(h-k)}{2(n+m+k+h)}\right)\cos\left(\frac{\pi(m-n)}{2(n+m+k+h)}\right)|00\rangle \\
+i&\cos\left(\frac{\pi(h-k)}{2(n+m+k+h)}\right)\sin\left(\frac{\pi(m-n)}{2(n+m+k+h)}\right)|01\rangle \\
+i&\sin\left(\frac{\pi(h-k)}{2(n+m+k+h)}\right)\cos\left(\frac{\pi(m-n)}{2(n+m+k+h)}\right)|10\rangle \\
-&\sin\left(\frac{\pi(h-k)}{2(n+m+k+h)}\right)\sin\left(\frac{\pi(m-n)}{2(n+m+k+h)}\right)|11\rangle.
\end{aligned}$$

The probability of accepting $a^n b^m x$ with $|x|_{a,b} = 0, |x|_c = k$ and $|x|_d = h$ is $\cos^2(\frac{\pi(h-k)}{2(n+m+k+h)})\cos^2(\frac{\pi(m-n)}{2(n+m+k+h)})$ which is at least $\lambda_1 \cdot \lambda_2$. Notice that such probability is 1 if $n = m$ and $k = p$ and 0 if $m = 0$, $p = 0$, $k = 0$ or $n = 0$. □

It can be argued that the time schedule demands too much precision to be implemented. Indeed, running an unitary operator for time $\frac{1}{n}$ with large n may be a challenge. Fortunately, the time can be rescaled as stated by the following theorem.

For any input x, time schedule $\tau(x) = (\tau_1, \ldots, \tau_n)$ and a positive real number k, we say that $k\tau(x) = (k\tau_1, \ldots, k\tau_n)$ is the time schedule τ *scaled* by k.

Theorem 11. *Given any positive real constant k, for any CTQA M with time schedule τ, there exists a CTQA M' with time schedule τ' where τ' is τ scaled by k and M' recognizes the same language as M.*

Proof. Let $M = (Q, \Sigma, \delta, \tau, s, A, R)$ be a CTQA such that δ defines unitary operations $U_\sigma(t)$ for each $\sigma \in \Sigma$ and a scheduler S computes a time schedule $\tau(x) = (\tau_1, \ldots, \tau_{|x|})$. Then, we can define $M' = (Q, \Sigma, \delta', \tau', s, A, R)$ where for each $\sigma \in \Sigma$, the transition function δ' computes $U'_\sigma(t) = U_\sigma(\frac{t}{k})$ and $\tau'(x) = (k\tau_1, \ldots, k\tau_{|x|})$.

On input $x = x_1 \ldots x_n$, the machine M computes

$$U_{x_1}(\tau_1)\ldots U_{x_n}(\tau_n)|s\rangle = U'_{k_1}(k\tau_1)\ldots U_{x_n}(k\tau_n)|s\rangle$$

which is also the computation done by M'. □

5 Concluding Remarks and Open Problems

In this work we introduce a new model of quantum computation with classical control called CTQA (for classically time-controlled quantum automata) where

all unitary operators are time-dependent and their time executions are externally controlled by a scheduler. We show in Theorem 2 that if Moore-Crutchfield quantum automata use time-dependent unitary operators with unrestricted time schedules, then they can recognize non-recursive languages. If the scheduler is defined via a finite-state automaton, however, a CTQA can recognize non-regular languages as shown in theorems 4 and 8. The CTQA model is an interesting model to study quantum computations assisted by a classical control that can tune or adjust execution times. Below we present some interesting open problems that remain from this work.

1. *Upper bound for classes of languages recognized by CTQAs.* To prove an upper bound on the simulation of CTQAs we require a simulation of the behavior of schedulers. Since a scheduler output real numbers, it is necessary to consider computable real numbers and study how much error in the time-dependent computation is introduced.
2. *Closure of well-known operations.* It is unknown under which operations the classes of languages recognized by CTQAs are closed, like union, intersection, homomorphism, inverse homomorphism, etc.
3. *Impossibility results.* We have not shown any impossibility result in this work. It will be interesting to see a lower bound technique for CTQAs analogous to a pumping lemma.

Acknowledgements

The authors thank Abuzer Yakaryılmaz for comments and discussions on a preliminary version of this paper.

References

1. Ambainis, A., Freivalds, R.: 1-way quantum finite automata: Strengths, weaknesses and generalizations. In: Proceedings of the 39th Annual Symposium on Foundations of Computer Science (FOCS). pp. 332–341 (1998)
2. Ambainis, A., Watrous, J.: Two-way finite automata with quantum and classical states. Theoretical Computer Science **287**(1), 299–311 (2002)
3. Brodsky, A., Pippenger, N.: Characterizations of 1-way quantum finite automata. SIAM Journal on Computing **31**(5), 1456—1478 (2002)
4. Díaz-Caro, A.: A lambda calculus for density matrices with classical and probabilistic controls. In: Chang, B.Y.E. (ed.) Programming Languages and Systems (APLAS 2017). Lecture Notes in Computer Science, vol. 10695, pp. 448–467. Springer, Cham (2017)
5. Green, A.S., Lumsdaine, P.L., Ross, N.J., Selinger, P., Valiron, B.: Quipper: a scalable quantum programming language. ACM SIGPLAN Notices (PLDI'13) **48**(6), 333–342 (2013)
6. Knill, E.H.: Conventions for quantum pseudocode. Tech. Rep. LA-UR-96-2724, Los Alamos National Laboratory (1996)
7. Moore, C., Crutchfield, J.P.: Quantum automata and quantum grammars. Theoretical Computer Science **237**(1–2), 275–306 (2000)

8. Nishimura, H., Yamakami, T.: An application of quantum finite automata to interactive proof systems. Journal of Computer and System Sciences **75**(4), 255–269 (2009)
9. Paykin, J., Rand, R., Zdancewic, S.: Qwire: A core language for quantum circuits. In: Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages. pp. 846–858. POPL 2017, ACM, New York, NY, USA (2017)
10. Say, A., Yakaryilmaz, A.: Quantum finite automata: A modern introduction. In: Computing with New Resources, LNCS, vol. 8808, pp. 208–222. Springer (2014)
11. Say, A., Yakaryilmaz, A.: Magic coins are useful for small-space quantum machines. Quantum Information & Computation **17**(11–12), 1027–1043 (2017)
12. Selinger, P.: Towards a quantum programming language. Mathematical Structures in Computer Science **14**(4), 527–586 (2004)
13. Selinger, P., Valiron, B.: A lambda calculus for quantum computation with classical control. Mathematical Structures in Computer Science **16**(3), 527–552 (2006)
14. Villagra, M., Yamakami, T.: Quantum and reversible verification of proofs using constant memory space. In: Proceedings of the 3rd International Conference on the Theory and Practice of Natural Computing (TPNC). LNCS, vol. 8890, pp. 144–156 (2014)
15. Villagra, M., Yamakami, T.: Quantum state complexity of formal languages. In: Shallit, J., Okhotin, A. (eds.) Proceedings of the 17th International Workshop on Descriptional Complexity of Formal Systems (DCFS). LNCS, vol. 9118, pp. 280–291 (2015)
16. Yamakami, T.: One-way reversible and quantum finite automata with advice. Information and Computation **239**, 122–148 (2014)
17. Zheng, S., Qiu, D., Gruska, J.: Power of the interactive proof systems with verifiers modeled by semi-quantum two-way finite automata. Information and Computation **241**, 197–214 (2015)