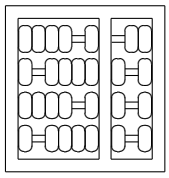

INSTITUTO DE COMPUTAÇÃO
UNIVERSIDADE ESTADUAL DE CAMPINAS

On the Quantum-Classical Separation of Marking and Multi-Head Automata

F. G. Jeronimo *A. V. Moura*

Technical Report - IC-14-07 - Relatório Técnico

May - 2014 - Maio

On the Quantum-Classical Separation of Marking and Multi-Head Automata

Fernando Granha Jeronimo * Arnaldo Vieira Moura †

Abstract

A 2QCFA is an automata model that combines constant size quantum and classical memories. This model is more powerful than any 2DFA since it recognizes some non-regular and even non-context free languages. In the classical paradigm, it is possible to increase the computational power of a 2DFA by using markers (pebbles) or augmenting the number of heads. Therefore, it is natural to investigate their quantum analogues. In this work, we propose a new general marking automaton based on the 2QCFA model. We prove that this model is more powerful than its classical analogues in terms of computability and complexity. Moreover, we answer affirmatively to an open question regarding the quantum-classical computability separation of multi-head 2QCFA and 2DFA.

1 Introduction

Feynman introduced the Quantum Computing paradigm after he realized that simulating quantum physics on classical computers was seemly hard [6]. Since then, many quantum computational models have been proposed. A crucial question to ask is whether these models are more powerful in terms of computability and computational complexity than their classical analogues. Quantum-classical separation theorems are important tools to investigate these questions. In this work, we introduce a new quantum automata model which uses markers and is based on the 2QCFA formalism of Ambainis and Watrous [1]. Moreover, we show that this new model is more powerful than the classical two-way marking deterministic and probabilistic finite automata in terms of computability and complexity, respectively.

Regarding computational complexity, there are two important results indicating that quantum computers could be more powerful than any classical one. One is Shor's algorithm [19] that factors integers in quantum polynomial time while there is no known polynomial time classical algorithm for this problem. The second result is an unordered database search known as Grover's algorithm that promotes a quadratic speedup [9]. In terms of computability, a quantum computer can simulate a classical one and vice versa. However, the quantum simulation by classical computers is only known to be of

*This work was supported in part by FAPESP 2013/20661-1. F. G. Jeronimo is with the Institute of Computing, University of Campinas, Campinas, SP Brazil (e-mail: fegranha@gmail.com).

†A. V. Moura is with the Institute of Computing, University of Campinas, Campinas, SP Brazil (e-mail: arnaldo@ic.unicamp.br).

exponential time complexity. Therefore, for quantum computers the only separation of interest is related to complexity issues.

In the case of restricted computational models, such as finite automata, the quantum-classical separation can also be in term of computability. Several models of quantum automata have been proposed [13] [14] [23] [20]. The 1QFA model recognizes only a proper subset of regular languages [13] [14]. Its extension, the 2QFA is strictly more powerful in the sense that it recognizes even non-regular languages [13] [8]. A less expressive model that uses only constant classical and quantum memory, namely 2QCFA, is also strictly more powerful than the 2DFA model, regarding computability and complexity [1] [18]. It recognizes all regular, some non-regular, and some non-context-free languages. Using this model, it is possible to recognize the palindrome language over a two letter alphabet, an impossible task, even for a two-way probabilistic finite automaton (2PFA) with bounded error [4]. Besides, 2QCFA also recognizes the non-regular language $L_{=} = \{a^n b^n | n \in N\}$ in polynomial time, whereas exponential time is needed in the 2PFA model.

There are several studies treating marking (pebble), and multi-head automata in the classical model [22] [17] [16] [3] [10] [12]. It is natural to study the impact of such extensions in the quantum paradigm and investigate quantum-classical separations in these more general models. Zheng et al. proposed a multi-head automata based on the 2QCFA model [23]. They presented the language $L_{pow(k)}$ as an indication that their new model was more powerful than its classical analogue in terms of computability for every number of k of heads with $k \geq 3$. Moreover, they let open the question of whether this was actually true. In this work, we show that, in fact, that language can not be used to establish such a separation. Nevertheless, we settle this question in the affirmative proving that their model is more powerful than its classical analogue.

Yakaryilmaz introduced a one counter automata based on the 2QCFA model and showed a quantum-classical computability separation of this new model and the classical 2DCA for sub-linear counter space [20] [21]. In principle the counter space in his model may be arbitrarily large. However, the languages he studied required only a logarithmic space counter in which case replacing the counter by a pebble would be equivalent. In fact, this is a special case of our marking 2QCFA model with just one marker.

This work is structured as follows. In section 2, we introduce the k-M2QCFA model, a marking extension of 2QCFA, and establish some general notation. In section 3, we show that the language $L_{pow(k)}$ [23] can not be used to obtain a computability quantum-classical separation for every number of heads $k \geq 3$. In section 4, we present our main results regarding a quantum-classical computability and complexity separations for marking and multi-head automata. In section 5, we revisit and prove some 2QCFA error amplification lemmas that are used in section 4, but that can also be used in broader contexts. Finally, in section 6, we conclude with some open problems. We assume the reader is familiar with quantum computing [15] and classical automata theory [11].

2 Definitions and Notations

In this section, we define the k-M2QCFA model and state some basic notations.

2.1 k Marker 2QCFA (k-M2QCFA)

A k-M2QCFA is formed by taking the 2QCFA model of Ambainis and Watrous [1] and extending it with the ability to place at most k labeled markers in the input. We define this model in the same way as was done for classical automata [12]. Formally, a k-M2QCFA is defined by a 10-tuple, $M = (Q, S, \Sigma, \Theta, \delta, K, q_0, s_0, S_{acc}, S_{rej})$, in which:

- Q and S are sets of quantum and classical states, respectively;
- Σ is the input alphabet;
- Θ and δ are the quantum and classical evolution functions, respectively;
- $K = \{1, ..., k\}$ is the set of labeled markers
- q_0 and s_0 are the quantum and classical initial states, respectively;
- S_{acc} and S_{rej} are sets of classical accepting and rejecting states, respectively ($S_{acc} \cap S_{rej} = \emptyset$).

The input tape has two tracks the input track and the marker track. Both tracks have the same number of cells. The tape head reads simultaneously the current symbol in the input and the marker track.

Now, we describe the input track. A generic input word x is placed in the input track with two special delimiters, † and \$, to mark the beginning and the end of input, respectively. These two symbols are not part of the input alphabet; and the track alphabet, Γ, is defined by $\Gamma = \{\dagger, \$\} \cup \Sigma$. Track position 0 has the symbol †. For $i \in [1, |x|]$, track position, i, has symbol x_i where $x = x_1 x_2 \dots x_{|x|}$. Finally, track position $|x| + 1$ has the symbol \$. The automaton is not allowed to move the tape head to the left of † nor to the right of \$.

For each cell in the input track, there is a corresponding cell in the marker track. The marker track has alphabet $K' = K \cup \{0\}$ and is initialized with all 0s. This symbol indicates the absence of markers. For any $m \in K$, there must be no more than one occurrence of m in the marker track. In spite of each cell holding only one marker, this is just a notation simplification since it is possible to store in the automaton states which markers share the same position of another already placed marker as he number of markers is constant. Let $S_{halt} = S_{acc} \cup S_{rej}$ be the set of halting states, and $S_{non} = S \setminus S_{halt}$ be the set of non halting states. We denote by s and q be the current classical and quantum state, respectively. Moreover, let σ and m be the current symbol in the input track and the current marker in the marker track.

The computation of a k-marker 2QCFA starts with the machine states at q_0 and s_0. At each iteration, the automaton uses s and σ to determine the quantum evolution $\Theta(s, \sigma)$, which can be a unitary transformation or a measurement. Formally, Θ is a mapping $\Theta : S_{non} \times \Gamma \to \mathcal{U}(\mathcal{H}(Q)) \cup \mathcal{M}(\mathcal{H}(Q))$. The set $\mathcal{U}(\mathcal{H}(Q))$ contains unitary transformations in the fixed dimension Hilbert space whose base states are in Q. Analogously, the set $\mathcal{M}(\mathcal{H}(Q))$ contains projective measurements in the same space. Next, the classical evolution, δ, is applied. Its form depends on the type of quantum evolution previously performed. If it was a measurement, δ is the mapping $\delta : S_{non} \times \Gamma \times K' \times R \to S \times K' \times D$, where, R is the set of possible measurement results, K is the set of markers, and $D = \{-1, 0, +1\}$ is the set of tape head directions of movement. The tape head directions left, none, and right are mapped to -1, 0, and $+1$, respectively. Otherwise, if the last quantum operation was a unitary transformation, δ is the usual classical transition function of the form $\delta : S_{non} \times \Gamma \times K' \to S \times K' \times D$.

The computation continues until a halting state is reached in which case the automaton accepts if $q \in S_{acc}$, and it rejects otherwise. For a given word x, there is a probability p_{acc} of x being accepted and a probability, p_{rej} of being rejected. The automaton may not halt, therefore, $p_{acc} + p_{rej}$ may be smaller than 1.

We observe that labeled and non-labeled markers are equivalent in terms of computability. This is because the number of markers is constant, and so a non-labeled M2QCFA can store in the classical states the current permutation of markers placed on the marker tape, together with the two markers on the left and on the right of the reading head. Since non-label markers are equivalent to pebbles, a k-M2QCFA is also a pebble automaton.

2.2 Notation

We present some definitions and notations used throughout this work. Besides, we follow our own pattern to designate automata models.

Definition 2.1. *($val[]$, $pos[]$) Let $x = \sigma_1\sigma_2\ldots\sigma_l$ ($l \geq 0$) be a word over the alphabet Σ. Let i be an integer value between 1 and l. We denote by $val[x, i]$ the symbol at the i^{th} position, that is $val[x, i] = \sigma_i$. Let h be the input head and suppose it is scanning symbol σ_i. We define $val[x, h] = \sigma_i$. Let m be a marker under the corresponding cell of σ_i. We denote by $pos[x, m]$ the position of m, that is $pos[x, m] = i$. If x is implicit from the context, we use the abbreviated notations $val[i] = val[x, i]$, $val[h] = val[x, h]$, and $pos[x, m] = pos[m]$.*

Definition 2.2. *($\Sigma[]$) Let L be a language. We denote by $\Sigma[L]$ the alphabet of L.*

Definition 2.3. *($markers[]$) Let M be a 2QCFA. We denote by $markers[M]$ the number of markers used by M. Note that we consider all M2QCFA to also be a 2QCFA, in the sense that the former is at least as powerful as the latter. If L is a language, $markers[M, L]$ is the minimum number of marker required to recognize L using model M. When M is clear from the context, we use $markers[L]$ to denote $markers[M, L]$.*

Definition 2.4. *(**0-sided**[], **1-sided**[], **2-sided**[], **and** L[]) Let M be a computational model, we denote by 0-sided[M], 1-sided[M], and 2-sided[M] the set of languages that can be recognized by M with zero error, one-sided error (words in the language always accepted [1]) and two sided error, respectively. Moreover, $L[M] =$ 0-sided[M] $\cup$ 1-sided[M] $\cup$ 2-sided[M].*

Definition 2.5. *The power language, $L_{pow(k)}$, is defined as $L_{pow(k)} = \{a^n b^{n^k} | n >$ 1 and k is a fixed constant in $\mathbb{N}\}$.*

We use the concatenation of extension, direction, type, and the suffix FA to designate a finite automata model. The extension can be none, T for Multi-tape, ST sensing Multi-tape, and M for marking. Direction can be 1 or 2 for one-way and two-way, respectively. Type can be D for deterministic, P for probabilistic, and QC for quantum classical. Table 1 summarizes the automata models used in this work.

Model	Description
2DFA	Two-way Deterministic Finite Automaton [11]
2PFA	Two-way Probabilistic Finite Automaton [2]
2QCFA	Two-way Quantum Classical Finite Automaton [1]
M2DFA	Marking Two-way Deterministic Finite Automaton [12]
T2DFA	Multi-head (Multi-tape) Two-way Deterministic Finite Automaton [10]
ST2DFA	Sensing Multi-head (Multi-tape) Two-way Deterministic Finite Automaton [10]
M2PFA	Marking Two-way Probabilistic Finite Automaton
M2QCFA	Marking Two-way Quantum Classical Finite Automaton
T2QCFA	Multi-head (Multi-tape) Two-way Quantum Classical Finite Automaton [23]
ST2QCFA	Sensing Multi-head (Multi-tape) Two-way Quantum Classical Finite Automaton

Table 1: Automata Summary

3 $L_{pow(k)}$ cannot separate k-T2DFA from k-T2QCFA for every k

Zheng et al. [23] used the language $L_{pow(k)}$ 2.5 as an indication of the computability separation for non-sensing k-head 2DFA (k-T2DFA) and non-sensing k-head 2QCFA (k-T2QCFA) for every $k \geq 3$. In this section, we show that this language cannot be used for this separation as a 4-marker 2DFA (4-M2DFA) can recognize it, and a k-marker 2DFA can be simulated by a non-sensing $(k+1)$-T2DFA. Therefore, there is a non-sensing 5-T2DFA for $L_{pow(k)}$ for every k. Firstly, we investigate the inclusion L[k-M2DFA] $\subseteq L$[(k+1)-T2DFA]. We close this section by presenting a description of 4-M2DFA that accepts $L_{pow(k)}$.

3.1 $(k+1)$-head 2DFA can simulate k-marker 2DFA

It is already known that a $(k+1)$-T2DFA can simulate k-M2DFA [16]. We just restate this result in terms of the equivalence between sensing k-ST2DFA and k-M2DFA [17].

Theorem 3.1. *L[k-M2DFA] $\subseteq L$[non-sensing (k+1)-T2DFA].*

Proof. Let A be a k-M2DFA. From [17], there is a sensing k-ST2DFA B that simulates A. Note that this result is not trivial, since as at least one head of B must simulate A's head resulting in at most $k-1$ heads to simulate markers. Now, it suffices to show how B can be simulated by a non-sensing $(k+1)$-T2DFA C. In order to provide the sensing feature, we use an extra head denoted h_e. Each time we want to know if two heads, h_i

and h_j, are in the same cell we run the simple procedure 1.

```
1  Place head h_e at † ;
2  same = yes; // h_i and h_j at same position?
3  while val[h_i] ≠ † and val[h_j] ≠ † do
4  |  Move h_e one position to the right ;
5  |  Move h_i and h_j one position to the left ;
6  end
7  if val[h_i] ≠ val[h_j] then
8  |  same = false ;
9  end
10 // Restore h_i and h_j original positions
11 while val[h_e] ≠ † do
12 |  Move h_e one position to the left ;
13 |  Move h_i and h_j one position to the right ;
14 end
```

Algorithm 1: Sesing feature with one extra head.

□

3.2 4-M2DFA to accept $L_{pow(k)}$, for any k

Zheng et al. created a non-sensing $(k+1)$-T2DFA for $L_{pow(k)}$. They used k heads to store digits of a number in base n. We follow a different approach. We show that given two delimited regions (contiguous sequence of symbols) in the input word of size C_1 and C_2, it is possible to obtain a delimited region comprising $C_1 * C_2$ symbols or detect no such region exists using only a fixed number of markers. This is formalized in the next lemma.

Lemma 3.2. *Let $x = a^+b^+$ be an input word with $l = |x|$. Let n denote the number of 'a's in x. If we have a delimited region from position p_{start} to p_{end} of size n^k, it is possible to find a region of size n^{k+1} starting from p'_{start} provided $(p'_{start} + n^{k+1}) \leq l$. We assume that it is possible to detect p_{start}, p_{end}, and p'_{start}. This procedure requires three additional temporary counters, and it runs in polynomial time. Moreover, the case $(p'_{start} + n^{k+1}) > l$ can be detected.*

Proof. The idea is to use the region of 'a's that contains n symbols in order to multiply the number of symbols between p_{start} and p_{end} by n.

Let m_{count} be a marker that will range from p_{start} to p_{end}. Let $m_{prospect}$ be a prospective marker that will start at p'_{start} and will finish at $(p'_{start} + n^{k+1})$ provided x is large enough. Let m_a be a marker that will range in the region of 'a's.

For each symbol within p_{start} to p_{end}, we do the following: starting from m_a in the first 'a', we advance m_a and $m_{prospect}$ one position to the left until $val[pos[m_a]] = b$.

The algorithm 2 makes precise this idea.

```
1  Let p_start and p_end be the positions delimiting a region of size n^k ;
2  Place a marker m_count at p_start ;
3  Place a marker m_prospect at p'_start ;
4  while pos[m_count] ≤ p_end do
5  |  Place a marker m_a at first 'a';
6  |  while val[pos[m_a]] ≠ b do
7  |  |  Move m_a one position to the right ;
8  |  |  Move m_prospect one position to the right ;
9  |  |  // |x| < p'_start + n^{k+1}
10 |  |  if val[pos[m_prospect]] = $ then
11 |  |  |  Error ;
12 |  |  end
13 |  end
14 |  Move m_count one position to the right ;
15 end
```

Algorithm 2: Multiply algorithm

Each step of the inner while loop adds n symbols to the region delimiteded by p'_{start} and $m_{prospect}$. Since this process is repeated n^k times, we end up with n^{k+1} symbols.

Placing markers m_{count}, $m_{prospect}$, and m_a requires time $O(l)$. Each step, of the inner while loop takes time $O(l)$ and is executed at most $O(l)$ times. The outer while loop is executed $O(l)$. Therefore, the total running time is $O(l^3)$.

If the input x is shorter than $(p'_{start}+n^{k+1})$, this situation is detected when $m_{prospect}$ is advanced. □

The following lemma shows how to find a region of size n^k. It is basically an induction using lemma 3.2.

Lemma 3.3. *Let $x = a^+b^+$ be an input word with $l = |x|$. Let n and num_b denote the number of 'a's and 'b's in x, respectively. If $num_b \geq n^k$, then it is possible to find a region of size n^k starting at the first 'b'. Otherwise, it is possible to detect that $num_b < n^k$. This procedure, runs in polynomial time and uses at most 4 markers.*

Proof. We proceed by induction on $k \geq 1$. Let $k = 1$. In this case, we use two markers m_a and $m_{prospect}$. The counter m_a starts from the first 'a' and $m_{prospect}$ starts from the first 'b'. Counter m_a and $m_{prospect}$ are advanced one position until $val[m_a + 1] = b$. The region from the first 'b' to $m_{prospect}$ contain n symbols. In this step, we use only two markers. This step runs in time $O(l^2)$.

Inductive step. We assume the result holds for some k, with $k > 1$. Let p_{start} be the position of the first 'b' from left to right. Let p_{end} the position of a marker satisfying $(p_{end} - p_{start} + 1) = n^k$. From lemma 3.2, making $p'_{start} = p_{start}$, the region can be increased to n^{k+1} using three additional markers. Since, we were already using a marker for p_{end} (p_{start} is implicit), at most 4 markers were used. For the running time, each of the k steps runs in $O(l^3)$. Therefore, the total running time is polynomial, even if $k \in O(p(l))$ for some polynomial $p(l)$. □

We compile these results in the following theorem.

Theorem 3.4. *Let $x = a^+b^+$ be an input string with $l = |x|$. There is a 4-marker 2-way deterministic finite automaton that decides $L_{power(k)}$ in polynomial time.*

Proof. If the number of 'b's satisfies $num_b \geq n^k$, then Lemma 3.3 states that a region of n^k 'b's can be delimited. It suffices to ensure that $val[m_{prospect}+1] = \$$ to guarantee that $x \in L_{power(k)}$. If $num_b < n^k$, by Lemma 3.3 this situation can be detected. Regarding the polynomial total running time, this same Lemma guarantees the procedure takes only polynomial time, and detecting if $pos[m_{prospect}] = l$ requires $O(l)$ time. Therefore, the result follows. □

4 Computability Separation

In this section, we prove a separation in the computability power of k-M2DFA and of k-M2QCFA, in the sense that there are languages recognizable by the latter but not by the former. We also prove a similar result for the multi-head case by showing a computability separation of k-T2DFA and k-T2QCFA. This last separation result was left as an open question in [23]. Furthermore, we present a complexity separation of the k-M2QCFA and a k-M2PFA. To accomplish these results, we use a hierarchy of languages, which was already used to prove that $(k+1)$ markers are better than k markers, in the classical case [12]. This hierarchy is defined next:

Definition 4.1. *$L_k(L)$ is a language defined inductively in terms of L as follows:*

- *If $k = 0$, then $L_0(L) = L$*
- *Otherwise, $L_k(L) = \{x_1\#x_2\#...\#x_{2n} |$ exactly half $x_i \in L_{k-1}(L), n \geq 1$ and $\# \notin \Sigma[L_{k-1}(L)]\}$*

Since half of the sub-strings x_i belong to $L_{k-1}(L)$, an automaton for deciding $L_k(L)$ will need to count. But, as n can be arbitrary, this counting is impossible using only finite classical memory, and so, a new marker is required to keep the counting.

One important property of this language hierarchy is stated in the next theorem, adapted from [12].

Theorem 4.2. *If L requires m markers to be recognized, then $L_k(L)$ requires $m + k$ markers to be decided using the M2DFA model.*

The next definition names the set of languages recognized by a k-M2DFA.

Definition 4.3. *The Classical Hierarchy level k, denoted by $CH(k)$, is defined as the set of languages $CH(k) = \{L |$ there is a k-M2DFA that recognizes $L\}$.*

As a corollary to Theorem 4.2, we have a computability power separation for 2DFA with $k+1$ and with k markers.

Corollary 4.4. $CH(k+1) \not\subseteq CH(k)$.

Proof. From Theorem 4.2 $L_{k+1}(L) \in CH(k+m+1)$, but $L_{k+1}(L) \notin CH(k+m)$ where $m = markers[L_k(L)]$. □

Analogously to the classical case, we define the set of languages recognized by a k-M2QCFA.

Definition 4.5. *The Quantum Hierarchy level k, denoted by $QH(k)$, is defined as the set of languages $QH(k) = \{L |$ there is a k-M2QCFA that recognizes $L\}$.*

4.1 Quantum-Classical Separations

In order to prove quantum-classical separations, we first prove several auxiliary lemmas. Let L be a language, and consider languages $L_k(L)$ and $L_{k-1}(L)$ for some $k \geq 1$. Let M be a 2QCFA that accepts language $L_{k-1}(L)$ with two-sided error ϵ. For an input word $y = x_1\#x_2\#\ldots\#x_{2n}$, the value of n can be arbitrarily large. Since the error ϵ is constant, the probability of correctly recognizing all x_i decreases exponentially if $\epsilon > 0$. The challenge is to amplify the error ϵ to $\frac{\epsilon}{f(n)}$ when simulating M on each x_i. We observe that the amplification must consider the global parameter of the input n; it is not possible to analyze each x_i in isolation if $\epsilon > 0$.

The first lemma assumes that this amplification is possible. We denote by $amplify[M]$ the new 2QCFA recognizing the same language as M, but, with error $\frac{\epsilon}{f(n)}$. With this assumption, we show how to create a 2QCFA, M' that decides $L_k(L)$. The next two lemmas show how to amplify the error in the special cases of a one-sided and two-sided error 2QCFA, respectively. They use results from some general amplification lemmas established in section 5. These three lemmas will allow us to show that languages at levels 0, 1, and 2 can be recognized using the same number of markers in the quantum M2QCFA model. As a M2DFA is trivially a k-M2QCFA, we will be able to conclude that M2QCFA is computationally more powerful than its classical counterpart.

Lemma 4.6. *Let $L_k(L)$ and $L_{k-1}(L)$ be languages at two consecutive levels in the language hierarchy based on a language L. If there is a 2QCFA M for language $L_{k-1}(L)$, such that $amplify[M]$ has two-sided error $\frac{\epsilon_0}{f(n)}$ with $f(n) = cn^3$ for some constant c, then there is a 2QCFA M' that accepts $L_k(L)$.*

Proof. An automaton M' is described algorithmically as follows:

```
// Let R_α denote a rotation by angle α
Check classically if the input is y = x_1#x_2#...#x_2n ;
while True do
    // Initialize one-qubit used for counting
    Let q_count = q_0 ;
    for i ∈ {1,...,2n} do
        Simulate amplify[M] on y with sub-string x_i ;
        if M accepts then
            q_count = R_{√2π} q_count ;
        end
        else
            q_count = R_{-√2π} q_count ;
        end
    end
    Measure q_count ;
    if q_count = q_0 then
        if Rand[p'_suc / (2^k n^2)] then
            Accept ;
        end
    end
    else
        Reject ;
    end
end
```

Algorithm 3: Automaton M' accepting the language $L_k(L)$.

Let the input word be $y = x_1\#x_2\#\ldots\#x_{2n}$. The probability of correctly recognizing all $2n$ x_is is $(1 - \frac{\epsilon_0}{f(n)})^{2n}$. Using Bernoulli's inequality, this probability, denoted p_{suc}, can be bounded:

$$p_{suc} \geq 1 - 2n\frac{\epsilon_0}{f(n)}.$$

Let $p'_{suc} = 1 - 2n\frac{\epsilon_0}{f(n)}$ be the minimum value of p_{suc}. Furthermore, let $n_{L_{k-1}(L)}$ and $n_{\not L_{k-1}(L)}$ be the number of x_i terms in $L_{k-1}(L)$ and not in $L_{k-1}(L)$, respectively. In [1], Ambainis and Watrous determined the probability of detecting when the number of clockwise and counter-clockwise $\sqrt{2}\pi$ rotations differs. In other words, when $n_{L_{k-1}(L)} \neq n_{\not L_{k-1}(L)}$ we get:

$$p_{detect} \geq \frac{1}{2(n_{L_{k-1}(L)} - n_{\not L_{k-1}(L)})^2}.$$

Since $n_{L_{k-1}(L)} + n_{\not L_{k-1}(L)} = 2n$, p_{detect} can be bounded below by:

$$p_{detect} \geq \frac{1}{2(2n)^2} = \frac{1}{8n^2}.$$

If $y \notin L_k(L)$, the probability of rejecting at each iteration is:

$$p_{rej} \geq p'_{suc}\frac{1}{8n^2}.$$

Otherwise, if $y \in L_k(L)$, the probability of rejecting it at each iteration is $p_{rej} \leq (1 - p_{suc})p_{detect}$. Moreover, since $p_{detect} \leq 1$, we get:

$$p_{rej} \leq 2n\frac{\epsilon_0}{f(n)}1.$$

We fix the acceptance probability at each iteration to $p_{acc} = \frac{p'_{suc}}{2^k n^2}$ and show that the error of M', denoted by ϵ', can be made arbitrarily small.

If $y \in L_k(L)$, the total acceptance probability, denoted by P_{acc}, is:

$$P_{acc} = \sum_{i \geq 0}(1 - p_{acc} - p_{rej})^i p_{acc} = \frac{p_{acc}}{p_{acc} + p_{rej}} = \frac{1}{1 + \frac{p_{rej}}{p_{acc}}}.$$

Now, we show that the ratio $\frac{p_{rej}}{p_{acc}}$ can be made arbitrarily small for $y \in L_k(L)$. Let $\gamma = \frac{2n\epsilon_0}{f(n)}$. By adjusting $f(n)$, $1 - \gamma$ can be made greater than $\frac{1}{2}$. The ratio $\frac{p_{rej}}{p_{acc}}$ can be expressed as:

$$\frac{p_{rej}}{p_{acc}} \leq \frac{\gamma}{1 - \gamma} 2^k n^2 \leq \gamma 2^{k+1} n^2.$$

For $f(n) = cn^3$, with c constant and $n > 1$, the fraction $\frac{p_{rej}}{p_{acc}}$ can be made arbitrarily small.

For $y \notin L_k(L)$, the total probability of rejecting denoted by P_{rej} is:

$$P_{rej} = \sum_{i \geq 0}(1 - p_{acc} - p_{rej})^i p_{rej} = \frac{p_{rej}}{p_{rej} + p_{acc}} = \frac{1}{1 + \frac{p_{acc}}{p_{rej}}}.$$

In this case, the fraction $\frac{p_{acc}}{p_{rej}}$ can also be made arbitrarily small by adjusting k:

$$\frac{p_{acc}}{p_{rej}} \leq \frac{8n^2}{2^k n^2}.$$

The expected running time of M' is polynomial provided that M runs in expected polynomial time.

□

The following lemma shows how a one-sided error 2QCFA for $L_{k-1}(L)$ can be amplified.

Lemma 4.7. *Let M be a 2QCFA accepting $L_{k-1}(L)$ with one-sided error ϵ. We build a M2QCFA M' with one-sided error at most $\frac{\epsilon}{f(n)}$ that accepts $L_{k-1}(L)$. The expected number of iterations of M' is polynomial in n provided that: $f(n)$ is polynomial and M runs in expected polynomial time. The number of marker of M' is $markers[M'] = \max\{markers[M], 1\}$.*

Proof. (We will also denote M' by $amplify[M]$). Note that from the definition of $L_k(L)$ the parameter n is half the number of strings x_i in the input. The description of M' is

given in 4.

```
Let y = x_1#x_2#...#x_2n ;
Let M be a 2QCFA with one-sided error ε;
Let p_stop = 1/(2f(n)n) be the probability of stopping the procedure ;
Let the head be at pos_{x_i}: the position preceding the first symbol of x_i († or #);
// To simulate probability p_stop we need to remember position x_i
Place marker m_{x_i} at pos_{x_i} ;
// Rand[(1 − p_stop)] returns true with probability (1 − p_stop)
while Rand[(1 − p_stop)] do
    // M is never simulated with marker m_{x_i} present
    Remove marker m_{x_i} ;
    Simulate M on x_i ;
    if M rejects x then
        Reject ;
    end
    Place marker m_{x_i} at pos_{x_i} ;
end
Remove marker m_{x_i} ;
Accept ;
```

Algorithm 4: $amplify[M]$ where M is a one-sided error 2QCFA

From Lemma 5.1 and given that $p_{stop} \leq \frac{1}{2f(n)\log f(n)}$, we have that $amplify[M]$ has one-sided error $\frac{\epsilon}{f(n)}$ and simulates M a polynomial number of times in n. Moreover, the only marker $amplify[M]$ uses is never used when M is simulated. Therefore $markers[amplify[M]] = \max\{markers[M], 1\}$. □

The following lemma shows how a two-sided error 2QCFA for $L_{k-1}(L)$ can be amplified.

Lemma 4.8. *Let M be a 2QCFA accepting $L_{k-1}(L)$ with two-sided error ϵ. We build a M2QCFA M' with two-sided error at most $\frac{\epsilon}{f(n)}$ that accepts $L_{k-1}(L)$. The expected number of iterations of M' is polynomial in n provided that: $f(n)$ is polynomial and M runs in expected polynomial time. The number of markers of M' is $markers[M'] = \max\{markers[M] + 1, 2\}$.*

Proof. (We will also denote M' by $amplify[M]$). From Lemma 5.4, we know that if we can store and update a gambler's ruin game state starting with n coins for each player, then we can reduce the error ϵ to $\frac{\epsilon}{2f(n)}$ where $f(n)$ is $\cup_{k>1}O(k^n)$. Now, we show how to do this using a marker to store the game state. Initially, we use a one-sided 1-M2QCFA to find the n^{th} symbol # with probability $(1 - \frac{\epsilon}{2f(n)})$ and we place a marker m_{gamble} at this position. Note that we could have done this classically as we have two markers. However, we intend to keep the number of markers as low as possible. Afterwards, we simulate M on x_i. Each time it accepts x_i, we move m_{gamble} to the next # or \$ on the right. Each time, it rejects x_i, we move m_{gamble} to the next # or † on the left. If $val[m_{gamble}]$ is \$, there is a high probability that $x_i \in L_{k-1}(L)$. Otherwise, there is a high probability that $x_i \notin L_{k-1}(L)$.

The description of M' is detailed in 5.

```
Let y = x_1#x_2#...#x_2n ;
Let M be a 2QCFA with two-sided error ε;
Let the head be at pos_x_i: the position preceding the first symbol of x_i († or #);
Place a marker m_x_i at pos_x_i ;
// Marker m_gamble will store the state of the gambler's ruin game
Place a marker m_gamble in the n symbol # ;
while val[m_gamble] = # do
    Remove m_x_i ;
    Run M on x_i ;
    Put m_x_i back at pos_x_i ;
    if M accepts x_i then
        Move m_gamble to the next # or $ ;
    end
    else
        Move m_gamble to the previous # or † ;
    end
end
if val[m_gamble] = $ then
    Accept ;
end
else
    Reject ;
end
```

Algorithm 5: $amplify[M]$ where M is a two-sided error 2QCFA

Note that $amplify[M]$ keeps the marker m_{gamble} while simulating M. Therefore, $amplify[M]$ use, at least, one more marker than M. Additionally, $amplify[M]$ uses an anchor marker m_{x_i} not to loose the position of x_i. This marker m_{x_i} is not kept when simulating M, but it is used at the same time m_{gamble} is used. For this reason, the total number of markers used by M' is $markers[M'] = \max\{markers[M] + 1, 2\}$. □

Now, we are ready to show the computability separations lemmas. Initially, we show how to go from a language L recognized by a M2DFA to a language $L_1(L)$ without using extra markers and obtaining a one-sided error 2QCFA. Secondly, we show how to go from a one-sided error 2QCFA for $L_1(L)$ to a two-sided 2QCFA for $L_2(L)$, again using the same number of markers. Finally, we show how to go from a two-sided error 2QCFA for $L_{k-1}(L)$ to a two-sided error M2QCFA for $L_k(L)$, but now we require one extra marker. These results imply that the language hierarchy moves down two levels in terms of required markers.

Lemma 4.9. *If $L \in CH(markers[L])$, then $L_1(L) \in QH(markers[L])$. Moreover, $L_1(L)$ is accepted by a one-sided 2QCFA.*

Proof. Let M_0 be the m-M2DFA that recognizes L with $m = markers[L]$. We build a m-M2QCFA, M, that simulates M_0 and has one-sided error. From our assumption M_0 recognizes L with zero error as it is a M2DFA. Consequently, $amplify[M_0]$ satisfies the hypothesis of Lemma 4.6. For a $y = x_1\#x_2\#\ldots\#x_{2n} \in L_1(L)$, as every x_i is recognized exactly the net rotation will be zero and M' will be a one-sided error automaton. □

Lemma 4.10. *If $L \in CH(markers[L])$, then $L_2(L) \in QH(markers[L])$, for $markers[L] \geq 1$. Moreover, $L_2(L)$ is accepted by a two-sided 2QCFA.*

Proof. Lemma 4.9 states that there is a one-sided error 2QCFA M accepting L_1. Besides, Lemma 4.6 states that from $amplify[M]$ with error $\frac{\epsilon}{f(n)}$, we can get a two-sided error 2QCFA accepting $L_k(L)$. From Lemma 4.7, we get this one-sided error amplification with a number of markers of $amplify[M]$ as $\min\{markers[M], 1\}$. □

Lemma 4.11. *The language hierarchy shifts down two levels in the M2QCFA model, that is, $L_k(L) \in QH(markers[L] + k - 2)$ for $markers[L] \geq 2$.*

Proof. The proof is an induction on $k \geq 1$.

Base cases: $k = 1$ and $k = 2$ follow from Lemmas 4.9 and 4.10, respectively.

Inductive Step. Suppose the result holds for some $k \geq 2$, then there is a M2QCFA, M, with $markers[M] = (m + k - 2)$, $m \geq 2$, and $k > 2$. Combining Lemmas 4.6 and 4.8, we have a 2QCFA M' with $markers[M] = (m+(k+1)-2)$ accepting $L_{k+1}(L)$. □

It is important to note that as we move from L to $L_1(L)$, if a probabilistic automaton were used, instead of a quantum one, the expected time would have been exponential. Replacing each x_i in an input $y = x_1\#x_2\#\ldots\#x_{2n}$ by a if $x_i \in L$ and by b otherwise, the resulting language is still non-regular. Therefore, any 2PFA would require exponential time to accept this language [7]. This same reasoning holds for every level $k \geq 1$ in the M2PFA model. This result represents a quantum-classical complexity separation.

If 0-sided[$2QCFA$] = 1-sided[$2QCFA$] or 1-sided[$2QCFA$] = 2-sided[$2QCFA$], a simple induction using Lemma 4.9 or Lemma 4.10 would show that the language hierarchy $L_k(L)$ would collapse, in the sense that a fixed number of markers would be used to recognize $L_k(L)$ for every k, in the quantum model.

Theorem 4.12. *If M_0 with two-sided error ϵ can be amplified for an arbitrary error $\frac{\epsilon}{p(n)}$ for some polynomial $p(n)$, then $L_k(L) \in QH(markers[L])$ for every k.*

It is more likely that 0-sided[$2QCFA$] $\subset$ 1-sided[$2QCFA$] $\subset$ 2-sided[$2QCFA$] with strict inclusions. In this case, there would be a language L requiring a two-sided error 2QCFA M_0 to be recognized. We could use L to derive the language $L_1(L)$. For a word $y = x_1\#x_2\#\ldots\#x_{2n}$, n could be made arbitrarily large in which case M_0 error would need to be amplified. If $markers[M_0] = 0$, we cannot even use a marker to store x_i's position. In all cases, the amplification needs to be done using only constant classical and quantum states, otherwise it will require extra markers. As these obstacles seem hard to overcome for a 2QCFA, we conjecture that this amplification is impossible. An important implication of this conjecture is a hierarchy of languages not recognized by 2QCFA. Currently, there is no known language which can not be recognized by this model.

Conjecture 4.13. *Suppose 0-sided*[$2QCFA$] $\subset$ *1-sided*[$2QCFA$] $\subset$ *2-sided*[$2QCFA$] *with strict inclusions. Let $L \in$ 2-sided*[$2QCFA$] $\setminus$ *1-sided*[$2QCFA$], *and let M_0 be a 2QCFA recognizing L. Let the input word be $y = x_1\#x_2\#\ldots\#x_{2n}$. If M_0's error cannot be amplified, then $L_{k+1}(L) \notin QH(k + markers[L])$.*

In the classical paradigm, one or two markers are equivalent that is L[1-M2DFA] = L[2-M2DFA], trivially from [17]. Nonetheless, in the quantum case a 1-M2QCFA can recognize languages not known to be recognizable by 2QCFA. An example is the language $L_{middle} = \{xay | x, y \in \{a, b\}^\star | x| = |y|\}$.

For brevity, we give a sketch of the proof that L_{middle} is in L[1-M2QCFA]. It is known that we can get an one-sided error 2QCFA for $L_{=} = \{a^n b^n | n \in \mathbb{N}\}$. For an input $x = \sigma_1 \sigma_2 \dots \sigma_l$, it is possible to sequentially place a marker in each σ_i an test if $|\sigma_1 \dots \sigma_{i-1}| = |\sigma_{i+1} \dots \sigma_l|$ using a simple variation of this automaton. From Lemma 5.1, this can be accomplished with high probability. In case the test succeeds, if $\sigma_i = a$ the input is accepted, otherwise it is rejected. A similar procedure is used to find the n^{th} symbol # in Lemma 4.8.

4.2 Computability Seperation for Multi-head Automata

In this section, we answer an open question raised in [23]. That is, we show that an m-T2QCFA can recognize languages that m-T2DFA can not. Firstly, we prove that for positive integer k_1 and k_2 with $k_1 > k_2$, a T2DFA for $L_{k_1}(L)$ requires more heads than for a T2DFA that accepts $L_{k_2}(L)$. Then, the proof separating quantum-classical marking automaton follows by a similar reasoning as was done for multi-head case.

Theorem 4.14. *Let L be a language recognized by a non-sensing multi-head 2DFA. Then $L_{k+1}(L)$ requires more heads than $L_k(L)$.*

Proof. For a fixed m, suppose that there is a non-sensing m-T2DFA, M, for $L_k(L)$ and $L_{k+1}(L)$. It is trivial to see that M can be simulated by a m-M2DFA. It suffices to use one marker for each head of M. We have two levels of the language hierarchy being recognized by the same number of markers which is a contradiction to Theorem 4.14. □

Theorem 4.15. *L[k-T2DFA] $\subset$ L[k-T2QCFA] with strict containment for $k \geq 3$.*

Proof. The proof is analogous to the marker case as was done to prove Theorem [**?**]. Assume L requires m heads to be accepted in the T2DFA model. From L to $L_1(L)$, the proof is the same. From $L_1(L)$ to $L_2(L)$, three heads will be needed. One to mark the x_i being amplified, other to run through the whole input to simulate a probability depending on n, and a last one to provide the sensing feature. If the number of heads required by L is greater than or equal to 3, no extra heads are necessary. From $L_k(L)$ to $L_{k+1}(L)$, we need the three heads of the previous cases, as well as one extra head to keep the gambler's ruin game status. □

5 General Amplification Lemmas

In this section, we develop amplification lemmas that can be used for any 2QCFA. Lemmas 5.1 and 5.4 regard the cases of one-sided and two-sided 2QCFA amplification, respectively.

5.1 One-sided amplification

The following lemma states that it is possible to amplify a one-sided error 2QCFA to a polynomially small error.

Lemma 5.1. *(One-sided amplification) Let M be a 2QCFA with one-sided error ϵ that recognizes a language L, and let n denote some parameter. If it is possible to simulate a probability $p \leq \frac{\epsilon}{2f(n)\log f(n)}$, then it is possible to build a one-sided error 2QCFA, M'*

accepting L with error at most $\frac{\epsilon}{f(n)}$. The automaton M' simulates M an expected $O(\frac{1}{p})$ times.

Proof. We exploit the fact that amplifying a one-sided error 2QCFA can be simply done by a probabilistic procedure. To achieve the desired error, it suffices to probabilistically simulate M enough times with high probability. If M rejects, we known with certainty that the input x is not in L. The automaton M' is specified in 6.

```
1 Let M be a 2QCFA with one-sided error ϵ;
2 Let p_stop = p be the probability of stopping the procedure ;
3 while Rand[(1 − p_stop)] do
4 |   Simulate M on x ;
5 |   if M rejects x then
6 |   |   Reject ;
7 |   end
8 end
9 Accept ;
```

Algorithm 6: 2QCFA M' for one-sided error amplification

Let n_{it} be the minimum number of iterations to achieve the desired error. Note that once M is a one-sided error 2QCFA, the error decreases exponentially with n_{it}. We want the error after n_{it} to be:

$$\epsilon^{n_{it}} \leq \frac{\epsilon}{2f(n)}.$$

Then, n_{it} can be bounded below by:

$$n_{it} \geq \frac{1}{\log \epsilon}(\log \epsilon - \log 2 - \log f(n)).$$

Hence, there is suitable constant c' such that $n_{it} = c' \log f(n)$ satisfying the previous equation.

Let $(1 - p_{stop})$ be the probability of running M one more time. To run M n_{it} times with high probability, we have:

$$(1 - p_{stop})^{n_{it}} \geq 1 - \frac{\epsilon}{2f(n)}.$$

Using Bernoulli's inequality, that is $(1 + x)^n \geq 1 + nx$ for $x > -1$, we can obtain a lower bound for $(1 - p_{stop})^{n_{it}}$. We just make the lower bound greater than $1 - \frac{\epsilon}{2f(n)}$:

$$1 - n_{it}p_{stop} \geq 1 - \frac{\epsilon}{2f(n)}.$$

Then,

$$p_{stop} \leq \frac{\epsilon}{2f(n)n_{it}}.$$

The probability of running the desired number of iteration is at least $(1 - \frac{\epsilon}{2f(n)})$. In this case, the success probability is at least $(1 - \frac{\epsilon}{2f(n)})$. Therefore, the overall success probability is at least:

$$(1 - \frac{\epsilon}{2f(n)})(1 - \frac{\epsilon}{2f(n)}) \geq 1 - \frac{\epsilon}{f(n)}.$$

Denote the expected number times M will be simulated by $E[n_{it}]$. Then,

$$E[n_{it}] = p_{stop} \sum_{i \geq 1} i(1 - p_{stop})^i = p_{stop} \frac{(1 - p_{stop})}{(1 - (1 - p_{stop}))^2} = \frac{(1 - p_{stop})}{p_{stop}}.$$

Since $1 - p_{stop} > \frac{1}{2}$ for n sufficiently large, $E[n_{it}]$ is $O(\frac{1}{p_{stop}})$. □

Corollary 5.2. *If M runs in expected polynomial time, then M' also runs in expected polynomial time.*

Corollary 5.3. *Let M be a 2QCFA with one-sided error ϵ and let n denote the input size. It is possible to build a 2QCFA, M', with one-sided error $\frac{\epsilon}{f(n)}$ where $f(n)$ is a polynomial or $f(n) = 2^n$.*

Proof. We know how to generate probabilities of the form $\frac{1}{n}$ using the input word [1]. Besides, we also know how to generate probabilities of the form $\frac{1}{2}$ using Hadamard gate of one qubit. In order to generate a more sophisticated probability such as $\frac{1}{2^i n^j}$, we run the first procedure j times and the second i times and accept only if all have succeeded. □

5.2 Two-sided amplification

Lemma 5.4 states that is possible to amplify a one-sided error 2QCFA to a polynomially small error provided the required space to store the gambler's ruin game state.

Lemma 5.4. *(Two-sided amplification) Let M be a 2QCFA with two-sided error ϵ that recognizes a language L, and let n denote some parameter. If it is possible to store and update the state of a gambler's ruin game in which each player starts with n coins, then it is possible to build a two-sided error 2QCFA M' accepting L with error $\frac{\epsilon}{f(n)}$ where $f(n)$ is $\cup_{k>1} O(k^n)$. The automaton M' simulates M an expected $O(n)$ times.*

Proof. The amplification proof relies on simple results of the gambler's ruin problem. A gambler with $coins_g$ coins makes bets in a casino with $coins_c$ coins. The gambler wins each bet with probability p. The game only ends when the gambler or the casino is ruined loosing all coins. This problem is described in more depth in [5].

In the automata case, each simulation of M on input x can be viewed as a bet with probability $p = 1 - \epsilon$. Define q as $q = 1 - p$. The algorithm simulates a simple gambler's ruin problem and is described in 7. The probability of the gambler wining the game is:

$$P_{gambler} = \frac{1 - (\frac{q}{p})^n}{1 - (\frac{q}{p})^{2n}}.$$

Since $q < p$, we have:

$$P_{gambler} \geq 1 - (\frac{q}{p})^n.$$

The fraction $\frac{q}{p}$ is equal to $\frac{\epsilon}{1-\epsilon}$. From the definition of a M2QCFA, we have that $\frac{q}{p} < \frac{1}{k}$, for every constant $k > 1$. Therefore, the error becomes exponentially small in n.

The total success probability, p_{suc}, is:

$$p_{suc} \geq 1 - (\frac{1}{k})^n.$$

For every k and with $f(n)$ in $O(k^n)$, this probability is bounded below by:

$$p_{suc} \geq 1 - \frac{\epsilon}{f(n)}.$$

```
1  Let x be the input word ;
2  Let M be to 2QCFA to be amplified ;
3  Let coins_g be the number of gambler's coins ;
4  Let coins_c be the number of casino's coins ;
5  coins_g = coins_c = n ;
6  while coins_g > 0 and coins_c > 0 do
7  |  Run M on x ;
8  |  if M accepts x then
9  |  |  Increment coins_g ;
10 |  |  Decrement coins_c ;
11 |  end
12 |  else
13 |  |  Decrement coins_g ;
14 |  |  Increment coins_c ;
15 |  end
16 end
17 if coins_g = 2n then
18 |  Accept ;
19 end
20 else
21 |  Reject ;
22 end
```

Algorithm 7: 2QCFA M' for two-sided error amplification

The expected number of times M will be simulated for a given x_i is given by the equation from gambler's ruin problem:

$$n_{it} = \frac{2n}{p-q}\frac{1-(\frac{q}{p})^n}{1-(\frac{q}{p})^{2n}} - \frac{2n}{2(p-q)}.$$

Since $p > q$, n_{it} can be bounded by:

$$n_{it} \leq \frac{n}{p-q}.$$

From the definition of a 2QCFA, p and q are constants. Hence n_{it} is $O(n)$. Therefore, if M runs in polynomial time, M' will also run in polynomial time. □

6 Conclusion

In this work, we introduced the M2QCFA model by extending the 2QCFA model with markers. We showed that this new model is more powerful than its classical analogue M2DFA and M2PFA model in terms of computability and complexity, respectively. Regarding the open question about computability power of the quantum T2QCFA model versus the classical T2DFA model, with a number $k \geq 3$ of heads, we showed that the

language $L_{pow(k)}$, which was proposed as an indication of their separation is, in fact, not a good witness. There is a 5-T2DFA recognizing this language for every k. Nevertheless, we answered affirmatively the question: the T2QCFA model is indeed computationally more powerful than the T2DFA model.

For future work, there are some important questions left open. Among them, we would like to highlight:

- A language that can not be recognized by the 2QCFA model.
- Increasing the number of markers or heads also increases the computability power of the M2QCFA or T2QCFA model?
- Given access to a biased coin whose probability of turning heads is p, can a quantum algorithm identify if $p > \frac{1}{2}$ using a fixed amount of quantum memory?
- Obtaining more tight relationships among the languages accepted by the k-M2QCFA, k-T2QCFA, and k-ST2QCFA models, other than the trivial containments L[k-M2QCFA] $\subseteq$ L[(k+1)-ST2QCFA] $\subseteq$ L[(k+2)-ST2QCFA].

References

[1] A. Ambainis and J. Watrous. Two-way finite automata with quantum and classical states. *Theor. Comput. Sci.*, 287(1):299–311, September 2002.

[2] A. Condon. *Bounded error probabilistic finite state automata, Handbook on Randomized Computing*. Kluwer, 2001.

[3] Pavol Duris and Zvi Galil. Sensing versus nonsensing automata. In *Automata, Languages and Programming*, volume 944 of *Lecture Notes in Computer Science*, pages 455–463. Springer Berlin Heidelberg, 1995.

[4] Cynthia Dwork and Larry Stockmeyer. Finite state verifiers i: The power of interaction. *J. ACM*, 39(4):800–828, October 1992.

[5] W. Feller. *An Introduction to Probability Theory, Vol. 1.* Wiley, New York, NY, third edition, 1968.

[6] Richard P Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6-7):467–488, 1982.

[7] Albert G Greenberg and Alan Weiss. A lower bound for probabilistic algorithms for finite state machines. *J. Comput. Syst. Sci.*, 33(1):88–105, August 1986.

[8] A. B. Grilo and A. V. Moura. Language classes and quantum finite automata. In *Proceedings of the the IV Workshop-School in Quantum Computation and Information*, WECIC '12, pages 90 – 95, 2012.

[9] L. K. Grover. A fast quantum mechanical algorithm for database search. In *28th Annual ACM Symposium on the Theory of Computing*, page 212, 1996. New York.

[10] Markus Holzer, Martin Kutrib, and Andreas Malcher. Multi-head finite automata: Characterizations, concepts and open problems. In Turlough Neary, Damien Woods, Anthony Karel Seda, and Niall Murphy, editors, *CSP*, volume 1 of *EPTCS*, pages 93–107, 2008.

[11] J.E. Hopcroft and J.D. Ullman. *Introduction to automata theory, languages, and computation*. Addison-Wesley series in computer science. Addison-Wesley, 1999.

[12] Pei Hsia and Raymond T. Yeh. Finite automata with markers. In *ICALP*, pages 443–451, 1972.

[13] A. Kondacs and J. Watrous. On the power of quantum finite state automata. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, pages 66–, Washington, DC, USA, 1997. IEEE Computer Society.

[14] C. Moore and J. P. Crutchfield. Quantum automata and quantum grammars. *Theor. Comput. Sci.*, 237:275–306, April 2000.

[15] M.A. Nielsen and I.L. Chuang. *Quantum computation and quantum information*. Cambridge Series on Information and the Natural Sciences. Cambridge University Press, 2000.

[16] H. Petersen. Automata with sensing heads. In *Theory of Computing and Systems, 1995. Proceedings., Third Israel Symposium on the*, pages 150–157, Jan 1995.

[17] Holger Petersen. The equivalence of pebbles and sensing heads for finite automata. In BogdanS. Chlebus and Ludwik Czaja, editors, *Fundamentals of Computation Theory*, volume 1279 of *Lecture Notes in Computer Science*, pages 400–410. Springer Berlin Heidelberg, 1997.

[18] Daowen Qiu. Some observations on two-way finite automata with quantum and classical states. In De-Shuang Huang, II Wunsch, DonaldC., DanielS. Levine, and Kang-Hyun Jo, editors, *Advanced Intelligent Computing Theories and Applications. With Aspects of Theoretical and Methodological Issues*, volume 5226 of *Lecture Notes in Computer Science*, pages 1–8. Springer Berlin Heidelberg, 2008.

[19] P. W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. *Proceedings 35th Annual Symposium on Foundations of Computer Science*, 35:124–134, 1994.

[20] Abuzer Yakaryilmaz. One-counter verifiers for decidable languages. *CoRR*, abs/1207.3880, 2012.

[21] Abuzer Yakaryilmaz. Log-space counter is useful for unary languages by help of a constant-size quantum register. *CoRR*, abs/1309.4767, 2013.

[22] Andrew C. Yao and Ronald L. Rivest. K + 1 heads are better than k. *J. ACM*, 25(2):337–340, April 1978.

[23] Shenggen Zheng, Lvzhou Li, and Daowen Qiu. Two-tape finite automata with quantum and classical states. *International Journal of Theoretical Physics*, 50(4):1262–1281, 2011.