EFFICIENT TWO-WAY QUANTUM FINITE STATE AUTOMATA

by

Abuzer Yakaryılmaz

B.S., Computer Education and Educational Technology, Boğaziçi University, 2004

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computer Engineering
Boğaziçi University

2007

ACKNOWLEDGEMENTS

First of all, my wholehearted thanks go to my thesis supervisor Prof. A. C. Cem Say for his lasting guidance and support throughout this research. Without Prof. A. C. Cem Say's contributions, this thesis would not have been completed to the full.

I also would like to express my gratitude to Assoc. Prof. Can Özturan and Assist. Prof. Müfit Sezer, my thesis comittee members, who devoted their time and energy to this research.

I wish to thank TÜBİTAK for their financial support during my research. Thanks to their financial help, I found the chance to devote more of my time to this study.

I want to thank Hüseyin Birkan Yılmaz who shared his mathematical knowledge and provided me with technical support while writing this thesis. My heartfelt thanks also go to my dear friends Nuri Taşdemir, Dilek Çankaya, Kadir Kozan, and Fatih Mehmet Atak. I can never forget the fruitful contributions of the Department of Computer Engineering to whom I owe special thanks.

Last but not least, I would like to dedicate this thesis to my family for their continuous encouragement and motivation not only with this work but also with much else.

ABSTRACT

EFFICIENT TWO-WAY QUANTUM FINITE STATE AUTOMATA

The discovery of quantum algorithms which are exponentially more efficient than the best known classical algorithms for similar tasks has spurred researchers to compare the relative powers of the classical and quantum versions of several computational models to better understand the causes and limitations of the apparent power of quantum computing. One model for which such comparative analyses have led to interesting results is that of finite automata.

Among the various types of quantum finite automata, we concentrate on the strongest family, namely, two–way quantum finite automata (2qfa's). Kondacs and Watrous proved that 2qfa's are more powerful than their classical counterparts by describing a method for constructing 2qfa's that recognize the non–regular language $L_{eq} = \{a^n b^n |\ n > 0\}$ for any given error bound $\epsilon > 0$. Machines built according to this method have $O((\frac{1}{\epsilon})^2)$ states, and they halt after $O((\frac{1}{\epsilon})|w|)$ steps, where w is the input string.

In this thesis, we examine ways of reducing the dependence of these cost functions on the desired error bound. We present more efficient constructions to recognize the same language. One of our methods produces machines which halt in $O(|w|)$ time (i.e. the running time does not depend on the error bound) and which have $O((\frac{1}{\epsilon})^{\frac{2}{c}})$ states for any given constant $c > 1$. Other methods, yielding machines whose state complexities are polylogarithmic in $\frac{1}{\epsilon}$, and which halt in $O(\log(\frac{1}{\epsilon})|w|)$ time, are also presented.

ÖZET

VERİMLİ ÇİFT YÖNLÜ KUANTUM SONLU DURUMLU MAKİNELER

Benzer işler için bilinen en iyi klasik algoritmalardan üstel olarak daha verimli kuantum algoritmalarının keşfi, araştırmacıları kuantum hesaplamanın görünürdeki üstünlüğünün sebeplerini ve sınırlarını daha iyi anlamak için bir çok hesaplama modelinin klasik ve kuantum sürümlerinin göreceli güçlerini karşılaştırmaya itmiştir. Bu şekildeki karşılaştırmalı analiz sonuçları ilginç görünen modellerden biri sonlu durumlu makinelerdir.

Bu çalışmada farklı kuantum sonlu makine türleri arasında en güçlü aile olan çift yönlü sonlu durumlu makinelere (2ksm) odaklanılmıştır. Kondacs ve Watrous herhangi bir verili pozitif hata sınırı ϵ için $L_{eq} = \{a^n b^n | \ n > 0\}$ dilini tanıyan 2ksm'nin inşası için buldukları yöntemi kullanarak 2ksm'lerin klasik benzerlerinden daha güçlü olduğunu ispatlamışlardır. w girdi dizisi olmak üzere, bu yönteme göre inşa edilen makineler $O((\frac{1}{\epsilon})^2)$ sayısında duruma sahiptir ve $O((\frac{1}{\epsilon})|w|)$ adım çalışırlar.

Araştırmamızda, bu masraf fonksiyonlarının istenen hata sınırına bağlılığını azaltmanın yolları incelenmektedir. Aynı dili tanıyan daha etkili yöntemler sunulmaktadır. Yöntemlerimizden birinin ürettiği sonlu makineler $O(|w|)$ adımda dururlar (çalışma zamanı hata sınırına bağlı değildir) ve verili herhangi bir c sabiti için $O((\frac{1}{\epsilon})^{\frac{2}{c}})$ sayıda duruma sahiptirler. Durum sayısı karmaşıklığı $\frac{1}{\epsilon}$'a göre poli–logaritmik olan ve $O(\log(\frac{1}{\epsilon})|w|)$ adımda duran sonlu makineler üreten başka yöntemler de sunulmaktadır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

A^*	Conjugate of the matrix A
A^T	Transpose of the matrix A
$A^\dagger$	Hermitian conjugate or adjoint of the matrix A
$\overline{c}$ or c^*	Conjugate of c
$\mathbb{C}$	Set of complex numbers
$L(A)$	The language recognized by automaton A
$L(A, \lambda)$	The language recognized by probabilistic automaton A with cut point λ
Λ	Empty string
Σ	Alphabet
Σ^*	The set of all strings written by symbols in the alphabet including empty string
1dfa	One–way deterministic finite state automaton
1gfa	One–way group finite state automaton
1moqfa	One–way measure-once quantum finite state automaton
1mmqfa	One–way measure-many quantum finite state automaton
1pfa	One–way probabilistic finite state automaton
1qfa	One–way quantum finite state automaton
1rfa	One–way reversible finite state automaton
1.5qfa	1.5–way quantum finite state automaton
2dfa	Two–way deterministic finite state automaton
2pfa	Two–way probabilistic finite state automaton
2qcfa	Two–way finite automaton with quantum and classical states
2qfa	Two–way quantum finite state automaton

1. INTRODUCTION

Researchers considering restricted theoretical models of quantum computers have focused on the quantum counterparts of classical finite state automata [1–11]. It is well known [12] that two–way deterministic finite state automata (2dfa's) recognize all and only the regular languages. There exist two–way probabilistic finite state automata (2pfa's) that can recognize some non-regular languages (like $L_{eq} = \{a^n b^n | \ n > 0\}$) [13], however, these machines necessarily have exponential running times [14]. Kondacs and Watrous [3] proved that two–way quantum finite automata (2qfa's) are more powerful than their classical counterparts by describing a method for constructing 2qfa's that recognize L_{eq} in time linear in terms of the input length for any given (one–sided) error bound $\epsilon > 0$. Machines built according to this method have $O((\frac{1}{\epsilon})^2)$ states, and they halt after $O((\frac{1}{\epsilon})|w|)$ steps, where w is the input string.

In this thesis, we examine ways of reducing the dependence of these cost functions on the desired error bound. We present more efficient constructions of 2qfa's to recognize L_{eq}.

One of our methods (Section 4.2) is inspired by the technique used by Freivalds [13] to construct 2pfa's that recognize the same language. Machines built according to the method of Kondacs and Watrous compare the numbers of a's and b's by utilizing the positions of the tape heads in different computational paths. By first checking whether the numbers of a's and b's are equivalent modulo k, we modify the Kondacs–Watrous method to then make a correspondingly faster comparison between the quotients obtained when the numbers of a's and b's are divided by k. Choosing k and the number of computational paths carefully, we are able to produce machines that halt in $O(|w|)$ time. This is the best runtime possible, because one cannot even read the complete string w in time less than that. The fact that the runtime does not depend on the error bound is interesting, since the conventional way of recognizing a language with desired error bound ϵ when given a machine for that language with one–sided error, say, $\frac{1}{2}$, involves a repetition procedure whose overall runtime is proportional to $\log \frac{1}{\epsilon}$.

In our method, only the size of the constructed program is dependent on ϵ.

In Section 4.3, we examine ways of producing smaller machines employing generalizations of the quotient comparison method. We present a method which can be used to construct machines with runtime $O(|w|)$, and state set size $O((\frac{1}{\epsilon})^{\frac{2}{c}})$ for any given constant $c > 1$. We also show that two other constructions with even smaller state complexities exist, if one allows the runtimes of the resulting machines to have small dependences on ϵ. One of these methods yields machines with $O(\frac{\log^3(\frac{1}{\epsilon})}{\log^3(\log(\frac{1}{\epsilon}))} \log^2(\frac{\log(\frac{1}{\epsilon})}{\log(\log(\frac{1}{\epsilon}))}))$ states and runtime $O(\frac{\log(\frac{1}{\epsilon})}{\log(\log(\frac{1}{\epsilon}))}|w|)$, which is still better than that of the conventional probability amplification technique mentioned above. The other method produces machines with the lowest state complexity $(O(\log^2(\frac{1}{\epsilon}) \log(\log \frac{1}{\epsilon})))$ known so far for 2qfa's that recognize L_{eq} in time linear in terms of the input length, and with runtime $O(\log(\frac{1}{\epsilon})|w|)$.

All the techniques described in Section 4.3 involve the repeated application of the quotient comparison procedure of Section 4.2 in multiple passes over the input string, with the aim of reducing the error probability to the desired value. As noted by Watrous, [2] a naive approach which cascades k copies of a QFT–based 2qfa with error $\frac{1}{N}$ ends up with an error probability which is considerably worse than N^{-k}. The reason is that the basic algorithm depends on different computational paths reaching the end of the string at the same time just if $w \in L_{eq}$, and unwanted interference occurs when, for instance, a path which takes i steps in the first pass and j steps in the second pass reaches the end of the second pass at exactly the same time as another path which takes j steps in the first pass and i steps in the second pass, even when $w \notin L_{eq}$. In our framework, the quotient comparison passes are performed with different divisors in each round, and these numbers are chosen in a way which guarantees that such undesired collisions never occur.

The remainder of this thesis is organized as follows: Chapter 2 and 3 are reviews of classical automata and quantum automata, respectively. In Chapter 4, we present our methods. We conclude with Chapter 5.

2. CLASSICAL FINITE AUTOMATA

In this chapter, we review the definitions of one– and two–way classical deterministic and probabilistic automata and their language recognition power. Moreover, we take a look at about reversible finite automata.

2.1. One–Way Deterministic Finite State Automata

Finite automata can be seen as Turing machines with read–only tapes. One–way deterministic finite state automata (1dfa's) are specifically one–tape, one–head, and one–way finite state automata. "One–way" means that their heads only move in one direction. We restrict our discussion to accepters, which after reading the whole input, just output "yes" or "no".

Rabin and Scott [15] give a detailed analysis of 1dfa's. The formal definition of 1dfa is below:

Definition 1. *A 1dfa is a 5–tuple* $\mathcal{U} = (S, \Sigma, M, s_0, F)$*, where* S *is a finite non–empty set (the internal state of* $\mathcal{U}$*),* Σ *is the input alphabet,* M *is a function defined on the Cartesian product* $S \times \Sigma$ *of all pairs of states and symbols with values in* S *(the table of transitions or moves of* $\mathcal{U}$*),* s_0 *is an element of* S *(the initial state of* $\mathcal{U}$*), and* F *is a subset of* S *(the designated final states of* $\mathcal{U}$*).*

The function M can be extended from $S \times \Sigma$ to $S \times \Sigma^*$ in a natural way by a definition by recursion as follows:

$$M(s, \Lambda) = s, \text{ for } s \text{ in } S;$$
$$M(s, x\sigma) = M(M(s, x), \sigma), \text{ for } s \text{ in } S, x \text{ in } \Sigma^*, \text{ and } \sigma \text{ in } \Sigma.$$

We can now define the languages recognized by 1dfa.

Definition 2. *The languages recognized by the 1dfa* $\mathcal{U}$*, i.e., L(U), is the collection of all strings* x *in* Σ^* *such that* $M(s_0, x)$ *is in* F*.*

Definition 3. *A language is called a regular language if some 1dfa recognizes it [16].*

The Myhill–Nerode theorem provides the necessary and sufficient conditions for a language to be regular [17]. In order to understand the theorem, we need some additional definitions.

Definition 4. *An equivalence relation R over the set Σ^* of strings is right invariant if whenever xRy, then $xzRyz$ for all z in Σ^*.*

Definition 5. *An equivalence relation over Σ^* is of finite index if there are only finitely many equivalence classes under the relation.*

Theorem 1. *(Myhill–Nerode) Let L be a language. The following three conditions are equivalent:*

(i) *L is regular;*

(ii) *L is the union of some the equivalence classes of a right–invariant equivalence relation over Σ^* of finite index;*

(iii) *the explicit right–invariant equivalence relation E defined by the condition that for all x, y in Σ^*, xEy if and only if for all z in Σ^*, whenever xz is in L, then yz is in L, and conversely, is an equivalence relation of finite index.*

Proof. See [15] or page 97 on [16]. □

2.2. Two–Way Deterministic Finite State Automata

In two–way deterministic finite state automata (2dfa's), the tape head can move in both directions and can also stay in place. The definition of 2dfa is as follows:

Definition 6. *A 2dfa is a 5–tuple $\mathcal{U} = (S, \Sigma, M, s_0, F)$ as in Definition 1 with the difference that now M is a function from $\Sigma \times S$ into $D \times S$ where $D = \{-1, 0, 1\}$. $\mathcal{U}$ operates as follows: it starts on the leftmost square of the tape in state s_0. When its internal state is s and it scans the symbol σ, then if $M(\sigma, s) = (d, s')$ it goes into the*

new state s' and moves one square to the right, stays where it is, or moves one square to the left, respectively, for the values of $d = +1, 0, -1$. The language recognized by $\mathcal{U}$ is the class of strings t such that $\mathcal{U}$ eventually moves off the right hand edge of t in a state belonging to F.

The languages recognized by 2dfa's are the same as those recognized by 1dfa's. Shepherdson [12] gives the following proof by using the Myhill–Nerode theorem.

Theorem 2. *For every two–way deterministic finite state automaton $\mathcal{U}$, there exists a one–way deterministic finite state automaton $\bar{\mathcal{U}}$ such that $L(\mathcal{U}) = L(\bar{\mathcal{U}})$. Furthermore $\bar{\mathcal{U}}$ can be obtained effectively from $\mathcal{U}$.*

Proof. For each $t \in \Sigma^*$ define a function $\tau_t : \bar{s_0} \cup S \to 0 \cup S$ (where $\bar{s_0}$ and 0 are not elements of S) as follows:
If $\mathcal{U}$ begins a walk on the rightmost symbol of t with state s ($s \in S$), there are three possibilities:

1. $\mathcal{U}$ never leaves t from the left or right, then $\tau_t(s) = 0$;
2. $\mathcal{U}$ leaves t from the left in any state, then $\tau_t(s) = 0$;
3. $\mathcal{U}$ leaves t from the right in state s', then $\tau_t(s) = s'$.

Similarly, $\tau_t(\bar{s_0})$ describes the result of the motion when $\mathcal{U}$ is started in the initial state s_0 on the leftmost symbol of t.

$\tau_{t_1} = \tau_{t_2}$ means that $\tau_{t_1}(s) = \tau_{t_2}(s)$ for all s in S and $\tau_{t_1}(\bar{s_0}) = \tau_{t_2}(\bar{s_0})$. In this case, it is easily seen that $t_1 \equiv t_2 \mod L(\mathcal{U})_R$ (i.e., $\forall t \in \Sigma^*,\ t_1 t \in L(\mathcal{U}) \leftrightarrow t_2 t \in L(\mathcal{U})$). If n is the number of internal states of $\mathcal{U}$, there are at most $(n+1)^{n+1}$ distinct τ_t. It follows that the set $L(\mathcal{U})$ is definable by a one–way finite state automaton $\bar{U}$. In other words, we have a bound , $(n+1)^{n+1}$, on the number of elements of $\Sigma^*/L(\mathcal{U})_R$ (i.e., all equivalence classes partitioned by automaton $\mathcal{U}$).

The remaining part of the proof is to show the effective method of obtaining $\bar{\mathcal{U}}$

from $\mathcal{U}$. $\bar{\mathcal{U}} = (S', \Sigma, M', s_0', F')$ can be defined as follows:

1. S' is the state set of $\bar{\mathcal{U}}$ and any element of S' corresponds to a function from $\{\bar{s_0}\} \cup S$ to $\{0\} \cup S$. In fact, for each $t \in \Sigma^*$, τ_t corresponds to a state of $\bar{\mathcal{U}}$. Since there may be at most $(n+1)^{n+1}$ distinct τ_t, the number of elements of S' is at most $(n+1)^{n+1}$.
2. s_0' is the initial state and it corresponds to the function τ_Λ where Λ is the empty string.
3. The elements of S' and the rules of M' can be found iteratively. Find all τ_t's for each $t \in \Sigma^*$ with length less than $(n+1)^{n+1}$ in the lexicographical order by simulating $\mathcal{U}$ at most $n|\,t|$ (maximum number configurations of $\mathcal{U}$) steps as follows:
   (a) start $\mathcal{U}$ in state s from the right–end for each $s \in S$ $(\tau_t(s))$.
   (b) start $\mathcal{U}$ in state s_0 from left–end $(\tau_t(\bar{s_0}))$.

   Note that, if the machine does not halt in $n|\,t|$ steps then it must enter a loop. Since the machine we are building has at most $(n+1)^{n+1}$ states, any state must be reachable by a string of length at most $(n+1)^{n+1}$,
   (a) we can find all elements of S' and
   (b) we can find all transitions between states (i.e., if $\bar{\mathcal{U}}$ is in the state corresponding to τ_t and reads symbol σ, then it goes to the state corresponding to $\tau_{t\sigma}$).
4. F' is a subset of S' and elements of it are found by simulating $\mathcal{U}$ for each $t \in \Sigma^*$ with length less than $(n+1)^{n+1}$ in lexicographical order for at most $n|\,t|$ steps. If the machine accepts t in $n|\,t|$ steps, then the state corresponding to τ_t is a final state. Otherwise, the machine rejects or does not move off t (the machine must enter a loop) and so the corresponding state is not a final state.

□

2.3. One–Way Probabilistic Finite State Automata

One–way probabilistic finite state automata (1pfa's) are similar to 1dfa's except that the state transitions are probabilistic. The definition of 1pfa is as follows [18]:

Definition 7. *A 1pfa is a 5–tuple* $\mathcal{U} = (S, \Sigma, M, s_0, F)$ *where* $S = \{s_0, \cdots, s_n\}$ *is a finite set (the set of states),* Σ *is the input alphabet,* M *is a function from* $S \times \Sigma$ *into* $[0,1]^{n+1}$ [1] *(the transition probabilities table) such that for* $(s, \sigma) \in S \times \Sigma$ *(*$p_i(s, \sigma)$ *is the probability of transition from* s *to* s_i*),*

$$M(s,\sigma) = (p_0(s,\sigma), \cdots, p_n(s,\sigma)),\ 0 \le p_i(s_i,\sigma) \le 1,\ \sum_i p_i(s,\sigma) = 1,$$

$s_0 \in S$ *(the initial state), and* $F \subseteq S$ *(the set of designated final states).*

For each symbol ($\sigma \in \Sigma$), we can show state transitions as a stochastic matrix where i^{th} row and j^{th} column shows the transition value going from s_{i-1} to s_{j-1}.

Definition 8. *For* $\sigma \in \Sigma$ *and* $x = \sigma_1\sigma_2\cdots\sigma_m$*, define the* $n+1$ *by* $n+1$ *matrices* $A(\sigma)$ *and* $A(x)$ *by*

$$A(\sigma) = [p_j(s_i,\sigma)]_{\substack{0\le i\le n\\ 0\le j\le n}},$$
$$A(x) = A(\sigma_1)A(\sigma_2)\cdots A(\sigma_m) = [p_j(s_i,x)]_{\substack{0\le i\le n\\ 0\le j\le n}}.$$

Note that $p_j(s_i, x)$ is the probability of $\mathcal{U}$ for moving from state s_i to state s_j by the input $x \in \Sigma^*$.

Definition 9. *If* $\mathcal{U} = (S, \Sigma, M, s_0, F)$ *and* $F = \{s_{i_0}, \cdots, s_{i_r}\}$, $I = \{i_0, \cdots, i_r\}$, *define*

$$p(x) = \sum_{i\in I} p_i(s_0, x).$$

[1] $[0,1]$ is the closed unit interval $0 \le x \le 1$. $[0,1]^{n+1}$ is the set of all $n+1$–tuples $(x_0, \cdots, x_n)$, where $0 \le x_i \le 1$.

$p(x)$ is the probability of entering final states when $\mathcal{U}$ starts with s_0 and reads the input x.

Definition 10. *Let $\mathcal{U}$ be a 1pfa and λ be a real number, $0 \leq \lambda < 1$. The language $L(\mathcal{U}, \lambda)$ is defined by*

$$L(\mathcal{U}, \lambda) = \{x | x \in \Sigma^*,\ \lambda < p(x)\}.$$

If $x \in L(\mathcal{U}, \lambda)$, we say that x is accepted by $\mathcal{U}$ with cut–point λ. $L(\mathcal{U}, \lambda)$ will also be called the set defined by $\mathcal{U}$ with cut–point λ.

It is easily seen that, the definition of 1dfa is a special case of 1pfa, and so 1pfa's can recognize all regular languages.

It is an interesting fact that if λ is a rational number, then the class defined by any 1pfa is regular; however, if λ is allowed to be irrational, 1pfa's can recognize non–regular languages.

Theorem 3. *There exists a λ, $0 \leq \lambda < 1$ such that $L(\mathcal{U}, \lambda)$ is not regular.*

Proof. Let $\mathcal{U} = (S, \Sigma, M, s_0, F)$ be an 1pfa such that $S = \{s_0, s_1\}$, $\Sigma = \{0, 1\}$, $F = \{s_1\}$, $A(0) = P_0$, $A(1) = P_1$ where

$$P_0 = \begin{pmatrix} 1 & 0 \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}, \quad P_1 = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ 0 & 1 \end{pmatrix}.$$

If $P_{\delta_1}.P_{\delta_2}.\cdots.P_{\delta_n} = \begin{pmatrix} m & p \\ q & r \end{pmatrix}$, $\delta_i \in \{0, 1\}$ then $p = 0.\delta_n\delta_{n-1}\cdots\delta_1$, where p is written in binary expansion.

If $x = \delta_1\delta_2\cdots\delta_n \in \Sigma^*$, then $p(x) = 0.\delta_n\delta_{n-1}\cdots\delta_1$. The values $p(x)$ are dense in the whole interval $[0, 1]$. Thus, $\exists\lambda_1$ such that $0 \leq \lambda < \lambda_1 < 1$ and $L(\mathcal{U}, \lambda_1) \subset L(\mathcal{U}, \lambda)$ where the inclusion is proper. The sets $L(\mathcal{U}, \lambda),\ 0 \leq \lambda < 1$ form a nondenumerable

pairwise different collection of sets. However, regular languages form a denumerable set. This completes the proof. □

Although the proof of Theorem 3 is purely an existence proof, we can give a specific λ such that $L(\mathcal{U}, \lambda)$ is not regular.

Example 1. *Let $w_1, w_2, \cdots$, be any enumeration of Σ^*; then for $\lambda = 0.w_1w_2\cdots$, $L(\mathcal{U}, \lambda)$ is not regular.*

Definition 11. *A cut–point λ is called isolated with respect to $\mathcal{U}$ if there exists a $0 < \delta$ such that*

$$\delta \leq |p(x) - \lambda| \text{ for all } x \in \Sigma^*. \tag{2.1}$$

Theorem 4. *Let $\mathcal{U}$ be an 1pfa and λ be an isolated cut–point with respect to $\mathcal{U}$ satisfying (2.1). There there exists a 1dfa $\mathcal{B}$ such that $L(\mathcal{U}, \lambda) = L(\mathcal{B})$. If $\mathcal{U}$ has n states and r states, then $\mathcal{B}$ can be chosen to have e states where*

$$e \leq (1 + \frac{r}{\delta})^{n-1}. \tag{2.2}$$

Proof. The proof is a bit complicated. We refer the reader to [18]. □

2.3.1. State Set Size Efficiency of 1pfa

If we consider the equivalent 1dfa's and 1pfa's, the state set sizes of 1pfa's are same as or less than those of 1dfa's. In this part, we will give an example 1pfa with n states that is equivalent to a 1dfa with $\Omega(2^n)$ states, which recognizes the language L_n on the alphabet $\{a, b\}$ consisting of all strings that have n or more symbols and the n^{th} symbol from the end of string is a [19, 20].

Lemma 1. *Any one–way deterministic finite automaton recognizing L_n has $\Omega(2^n)$ states.*

Proof. Any 1dfa recognizing L_n must remember the last n symbols during reading the string. Since keeping track of n symbols requires 2^n states, the 1dfa must have at least 2^n states. □

Let $\mathcal{U} = (S, \Sigma, M, s_0, F)$ be a 1pfa with $n+3$ states as follows:

1. $S = \{s_0, s_1\} \cup \{q_i |\ 0 \leq i \leq n\}$.
2. s_0 is the start state.
3. $F = \{s_1, q_n\}$.
4. M is defined with two constants δ $(0 < \delta < \frac{1}{4e(n+1)})$ and x $(= \frac{n}{n+1})$:
   (a) Without reading any symbol, $\mathcal{U}$ passes from state s_0 to state s_1 with probability $\frac{1}{2} - \delta$ and to state q_0 with probability $\frac{1}{2} + \delta$.
   (b) On any symbol, $\mathcal{U}$ passes from state s_1 to state s_1.
   (c) On symbol a, $\mathcal{U}$ passes from state q_0 to state q_0 with probability x and to state q_1 with probability $1 - x$; On symbol b, it passes from state q_0 to state q_0.
   (d) On any symbol, $\mathcal{U}$ passes from state q_i to state q_{i+1} $(1 \leq i \leq n-1)$.
   (e) On any symbol, $\mathcal{U}$ passes from state q_n to state q_0.

Lemma 2. *If the input string $w \notin L_n$ then $\mathcal{U}$ rejects (does not accept) it with probability $\frac{1}{2} + \delta$.*

Proof. There are two final (accepting) states (s_1, q_n) and the probability of being at state s_1 is always $\frac{1}{2} - \delta$. If the input string length is less than n then the probability of being at state q_n is 0. Otherwise, the probability of being at state q_n equals the probability of being at state q_1 $n-1$ symbols ago. Since the n^{th} symbol from the end is "b", the probability of being at state q_1 ($n-1$ symbols ago) is 0. Therefore, the string $w \notin L_n$ is accepted by $\mathcal{U}$ with probability $\frac{1}{2} - \delta$ and rejected with probability $\frac{1}{2} + \delta$. □

Lemma 3. *The probability of being at state q_0 is never less than $(\frac{1}{2} + \delta)x^n$.*

Proof. If the input string length is k ($< n$), the probability of being at state q_0 is $\frac{1}{2}+\delta$ at the beginning and after each symbol, $\mathcal{U}$ passes from state q_0 to state q_0 with probability at least x so the probability of being at state q_0 is at least $(\frac{1}{2}+\delta)x^k$, which is bigger than $(\frac{1}{2}+\delta)x^n$. If the input string length is equal to or more than n, then the probability of being at state q_0 (p) is $p \geq p_0x^n + p_nx^{n-1} + p_{n-1}x^{n-2} + \cdots + p_2x + p_1$, where p_i denotes the probability of being at state q_i n symbols ago for $0 \leq i \leq n$. The equality holds only if the n^{th} symbol from the end is "a", in any other cases p is greater since $\mathcal{U}$ passes from state q_0 to state q_0 with probability 1 instead of x. So,

$$p_0x^n + p_nx^{n-1} + p_{n-1}x^{n-2} + \cdots + p_2x + p_1 \geq (p_0 + p_1 + \cdots + p_n)x^n = (\frac{1}{2}+\delta)x^n$$

□

Lemma 4. *If the input string $w \in L_n$, then $\mathcal{U}$ accepts it with probability at least $\frac{1}{2}+\delta$.*

Proof. Suppose that the probability of being at state q_0 n symbols ago is p. If $w \in L_n$, then the probability of q_1 $n-1$ symbols ago is $p(1-x)$, the same as the probability of being at state q_n at the end. Therefore, overall accepting probability of $\mathcal{U}$ is

$$\begin{aligned}
\Pr[\mathcal{U} \text{ accepts } w] &\geq \tfrac{1}{2} - \delta + p(1-x) \\
&\geq \tfrac{1}{2} - \delta + (\tfrac{1}{2}+\delta)x^n(1-x) \\
&\geq \tfrac{1}{2} - \delta + (\tfrac{1}{2}+\delta)(\tfrac{n}{n+1})^n\tfrac{1}{n+1} \\
&> \tfrac{1}{2} - \delta + (\tfrac{1}{2}+\delta)\tfrac{1}{e}\tfrac{1}{(n+1)} \\
&> \tfrac{1}{2} - \delta + (\tfrac{1}{2}+\delta)4\delta \\
&> \tfrac{1}{2} + \delta + 4\delta^2
\end{aligned}$$

$$\Pr[\mathcal{U} \text{ accepts } w] > \frac{1}{2}+\delta \tag{2.3}$$

□

Theorem 5. *There exists a 1pfa with n states such that the smallest equivalent 1dfa contains $\Omega(2^n)$ states.*

Proof. 1pfa $\mathcal{U}$ (described above) with n states (recognizing the language L_{n-3}) is such an automaton and any equivalent 1dfa must have at least $2^{n-3}(\in \Omega(2^n))$ states. □

2.4. Two–Way Probabilistic Finite State Automata

Similar to the relation between 1dfa's and 1pfa's, two–way probabilistic finite state automata (2pfa's) are the probabilistic version of 2dfa's. The first definition of 2pfa was given by Kuklin [21] without mentioning language recognition. The definition of 2pfa is as follows [22]:

Definition 12. *A particular 2pfa M is specified by a finite set Q of states, a finite input alphabet Σ, and a transition function δ. The set Q contains designated states q_0 (the initial state), q_a (the accepting state), and q_r (the rejecting state). Let ¢ be a symbol not in Σ. The transition function has the form*

$$\delta : (Q - \{q_a, q_r\} \times (\Sigma \cup \{¢\}) \times Q \times \{\textit{left, right, stationary}\} \rightarrow [0, 1]$$

where, for each fixed q and σ, the sum of $\delta(q, \sigma, q', d)$ over all q' and d equals 1. The meaning of δ is that, if M is in state q with the head scanning the symbol σ, then with probability $\delta(q, \sigma, q', d)$ the machine enters state q' and either moves the head one symbol in direction d if $d \in \{\textit{left, right}\}$ or does not move the head if $d = \{\textit{stationary}\}$.

The computation of M on input $x \in \Sigma^*$ begins with the word ¢x¢ written on the input tape; the head is positioned on the left endmarker ¢, and the state is q_0. The computation is then governed (probabilistically) by the transition function δ until M either accepts by entering state q_a or rejects by entering state q_r. It is assumed that δ is defined so that the head never moves outside the word ¢x¢. M halts when it enters state q_a or q_r. The language recognition of a 2pfa within a given error probability (ϵ) is defined as follows:

Definition 13. *Let $L \subseteq \Sigma^*$, let M be a 2pfa with input alphabet Σ, and let $0 \leq \epsilon < \frac{1}{2}$. Then M recognizes L within error probability ϵ if*

1. *for all* $x \in L$, *Pr[*M *accepts* x*]* $\geq 1 - \epsilon$, *and*
2. *for all* $x \notin L$, *Pr[*M *rejects* x*]* $\geq 1 - \epsilon$.

Generally speaking, if M recognizes L within error probability ϵ for some $\epsilon < \frac{1}{2}$, we say that M recognizes L. It is easily seen that 1dfa's, 2dfa's, and 1pfa's are special cases of 2pfa's by making small modifications. The language recognition power of 1dfa's, 2dfa's and 1pfa's with rational bounded error are equal, i.e., the set of regular languages. On the contrary, Freivalds [13] showed that a 2pfa can recognize the non–regular language L_{eq} but requires exponential running time.

$$L_{eq} = \{a^n b^n | \ n > 0\} \tag{2.4}$$

The algorithm is as follows [23]:

1. Check (deterministically) that the input is of the form $a^n b^m$ for some $n > 0, m > 0$ and that $n \equiv m \mod (k+1)$ for some constant k and if not, reject. (k controls the error probability of the 2pfa.)
2. While scanning the input repeatedly, flip a fair coin for each a and b; call these a–flips and b–flips, respectively.
   (a) A scan is said to be an a–success if all a–flips are heads but at least one b–flip is a tail.
   (b) A scan is said to be a b–success if all b–flips are heads but at least one a–flip is a tail.
3. If there are L a–successes before any b–success or vice versa, then reject, else accept. (L is also a constant that controls the error probability.)

If the input is not of the form $a^n b^m$ or $n \not\equiv m \mod (k+1)$, the input is rejected with probability 1. If the input is not rejected, we have two cases.

If $n = m$ then the probability of a–success and b–success is the same. The chance of L a–successes occuring before any b–success (and vice versa) is $(\frac{1}{2})^L$. Therefore, the

2pfa accepts a string in L_{eq} with probability at least $1 - (\frac{1}{2})^{L-1}$.

If $n \neq m$ then $n > m + k$ (suppose $n > m$). In a single trial,

$$\begin{aligned}
\frac{\Pr[a\text{–success}]}{\Pr[a\text{–success or } b\text{–success}]} &= \frac{\frac{1}{2^n}(1-\frac{1}{2^m})}{\frac{1}{2^n}(1-\frac{1}{2^m})+\frac{1}{2^m}(1-\frac{1}{2^n})} \\
&= \frac{\frac{2^m-1}{2^{m+n}}}{\frac{2^m-1}{2^{m+n}}+\frac{2^n-1}{2^{m+n}}} \\
&= \frac{2^m-1}{2^m-1(1+\frac{2^n-1}{2^m-1})} \\
&= \frac{1}{1+\frac{2^n-1}{2^m-1}} \\
&< \frac{1}{1+\frac{2^n}{2^m}}
\end{aligned}$$

$$\frac{\Pr[a\text{–success}]}{\Pr[a\text{–success or } b\text{–success}]} < \frac{1}{2^k+1} \tag{2.5}$$

Therefore, the probability of L b–successes without any a–success (causing the 2pfa to reject) is at least $(1 - \frac{1}{2^k+1})^L$.

For given ϵ, the calculation of k and L as follows:

For all $x \in L_{eq}$, $\Pr[M \text{ accepts } x] \geq 1 - \epsilon$

$$\begin{aligned}
&1 - \tfrac{1}{2}^{L-1} \geq 1 - \epsilon \\
&2^{1-L} \leq \epsilon \\
&1 - L \leq \log_2(\epsilon)
\end{aligned}$$

$$L \geq 1 - \log_2(\epsilon) \tag{2.6}$$

For all $x \notin L$, $\Pr[M \text{ rejects } x] \geq 1 - \epsilon$

$$
\begin{aligned}
&(1 - \tfrac{1}{2^k+1})^L \geq 1 - \epsilon \\
&1 - \tfrac{1}{2^k+1} \geq (1-\epsilon)^{\frac{1}{L}} \\
&1 - (1-\epsilon)^{\frac{1}{L}} \geq \tfrac{1}{2^k+1} \\
&2^k + 1 \geq \tfrac{1}{1-(1-\epsilon)^{\frac{1}{L}}} \\
&2^k \geq \tfrac{(1-\epsilon)^{\frac{1}{L}}}{1-(1-\epsilon)^{\frac{1}{L}}}
\end{aligned}
$$

$$k \geq \log_2\left(\frac{(1-\epsilon)^{\frac{1}{L}}}{1-(1-\epsilon)^{\frac{1}{L}}}\right) \tag{2.7}$$

If w is the input and $|w|$ is the length of w, the expected running time of the 2pfa must exceed $2^{|w|^b}$, where positive constant b depends on k and L. Note that there is no upper bound on the worst–case runtime. It is an interesting result that 2pfa's that run in subexponential time only recognize the regular languages [14, 24].

2.5. Reversible Finite State Automata

Since one of the characterictics of quantum computation is reversibility, quantum automata are closely related with reversible automata. Reversible automata were introduced by Pin [25]. Here, we give basic definitions of one–way group finite state automaton (1gfa) and one–way reversible finite state automaton (1rfa) [9]. Moreover, we define two–way reversible finite state automaton (2rfa) and show that 2rfa can recognize all regular languages.

In the definitions of 1gfa and 1rfa, Q is the set of states, Σ is the input alphabet, δ is the transition function, q_0 is the initial state, and F is the set of accepting states.

Definition 14. *A 1gfa is a 1dfa $M = (Q, \Sigma, \delta, q_0, F)$ with the restriction that for every state $q \in Q$ and every input symbol $\sigma \in \Sigma$ there exists exactly one state $q' \in Q$ such*

that $\delta(q^{'}, \sigma) = q$, i.e., δ is a complete[2] one–to–one function and the automaton derived from M by reversing all transitions is deterministic.

The set of languages recognized by 1gfa's is a proper subset of regular languages and they are called as group languages.

Definition 15. *A 1rfa is a 1dfa $M = (Q, \Sigma, \delta, q_0, F)$ such that for every state $q \in Q$ and every input symbol $\sigma \in \Sigma$ there is at most one state $q^{'} \in Q$ such that $\delta(q^{'}, \sigma) = q$, or, if there exist distinct states $q_1, q_2 \in Q$ and symbol $\sigma \in \Sigma$ such that $\delta(q_1, \sigma) = q = \delta(q_2, \sigma)$, then $\delta(q, \Sigma) = \{q\}$. The latter type of state is called a spin state because once a 1rfa enters it, it will never leave it.*

The set of languages recognized by 1rfa is also a subset of regular languages.

Definition 16. *A 2rfa is a 6–tuple $M = (Q, \Sigma, \delta, q_0, Q_{acc}, Q_{rej})$. Q is a finite set of states, $q_0 \in Q$ is the initial state, $Q_{acc} \subset Q$ is the set of accepting states, and $Q_{rej} \subset Q$ is the set of rejecting states. Q_{acc} and Q_{rej} are disjoint, and their union is the set of halting states (the machine stops its computation when entering halting states). The elements of $Q \backslash (Q_{acc} \cup Q_{rej})$ are non–halting states. Σ is the input alphabet. The symbols $\{¢, \$\} \notin \Sigma$ are used to mark the left and right ends of the input string, respectively. $\Gamma = \Sigma \cup \{¢, \$\}$ is the tape alphabet. δ is the transition function, which governs the behaviour of M, and is explained below..*

The configurations of a 2rfa running on a string w are pairs of the form (q, x), where q is the state and x is the head position. The initial configuration is $(q_0, 0)$. Suppose that every transition entering the same state drives the tape head moving in the same direction (left, right, or stationary). We represent this feature of a state using the appropriate one of the notations $\overleftarrow{q}$, $\overrightarrow{q}$, or $\downarrow q$ for this state in the machine description. We use δ_σ for the behaviour of the transition function δ for each symbol $\sigma \in \Sigma$. Because of reversibility, δ_σ is a permutation of Q.

[2]defined for all (state–input symbol) pairs

2.5.1. Simulation of 1dfa's by 2rfa's

By using a technique from [26], Watrous [2] showed that any 1dfa can be simulated by a 2rfa.

Let $A = (S, \Sigma, \mu, s_0, F)$ be a 1dfa, where S is the set of states, Σ is the input alphabet, μ is transition function, s_0 is the initial state, and F is the set of accepting states. We define a 2rfa $M = (Q, \Sigma, \delta, q_0, Q_{acc}, Q_{rej})$ which will recognize the same language as A.

In order to provide consistency between A and M (when reading the ¢ and \$ symbols), S and μ are extended to S' and μ' as follows. Let $S' = S \cup \{s_0', s_{acc}, s_{rej}\}$, where s_0', s_{acc}, and s_{rej} are not elements of S, and define

$$\mu'(s,\sigma) = \begin{cases} \mu(s,\sigma) & s \in S,\ \sigma \in \Sigma \\ s_0 & s = s_0',\ \sigma = ¢ \\ s_{acc} & s \in F,\ \sigma = \$ \\ s_{rej} & s \in S \setminus F,\ \sigma = \$ \end{cases}$$

For all other values, let μ' be undefined. Also define

$$\begin{aligned} I_{s,\sigma} &= \{s' \in S \mid \mu'(s',\sigma) = \mu'(s,\sigma)\}, \\ J_{s,\sigma} &= \{s' \in S \mid \mu'(s',\sigma) = s\}, \end{aligned}$$

and fix some ordering of the set S' (i.e., label each element of S' from 1 to $|S'|$). Let *max* and *min* denote the maximum and minimum functions relative to this ordering, and for any subset $T \subseteq S'$, let $succ(s,T)$ be the least element larger than s in T (assuming there is such an element).

The details of M are as follows: Let $Q = \{\overleftarrow{(s,-1)} \mid s \in S'\} \cup \{\overrightarrow{(s,+1)} \mid s \in S'\}$, and let $q_0 = \overrightarrow{(s_0',+1)}$, $Q_{acc} = \{\overrightarrow{(s_{acc},+1)}\}$ and $Q_{rej} = \{\overrightarrow{(s_{rej},+1)}\}$. For each $s \in S'$ and

$\sigma \in \Gamma$ for which $\mu^{'}(s,\sigma)$ is defined, let

$$\delta_\sigma \overrightarrow{(s,+1)} = \begin{cases} \overleftarrow{(succ(s, I_{s,\sigma}), -1)} & s \neq max(I_{s,\sigma}) \\ \overrightarrow{(\mu^{'}(s,\sigma), +1)} & s = max(I_{s,\sigma}), \end{cases}$$

and for every $s \in S^{'}$ and $\sigma \in \Gamma$ let

$$\delta_\sigma \overleftarrow{(s,-1)} = \begin{cases} \overrightarrow{(s,+1)} & J_{s,\sigma} = \emptyset \\ \overleftarrow{(min(J_{s,\sigma}), -1)} & J_{s,\sigma} \neq \emptyset. \end{cases}$$

Theorem 6. *For any 1dfa A, let M be the 2rfa defined above. For any $w \in \Sigma^*$, if A accepts w then M accepts w in $O(|w|)$ steps, and if A does not accept w, then M rejects w in $O(|w|)$ steps.*

Proof. Suppose that $A = (S, \Sigma, \mu, s_0, F)$ and let $A^{'} = (S^{'}, \Sigma, \mu^{'}, s_0^{'}, \{s_{acc}\})$ be a 2dfa, which is extended from A as described above. It is easily seen that $A^{'}$ only moves its tape head to the right and $L(A) = L(A^{'})$. Let $w^{'}$ represent the tape content ($\¢ w \$$) of $A^{'}$ on any input w of length n and so the set of configurations of $A^{'}$ on this input is $\{(s,x) \mid s \in S^{'}, x \in \mathbb{Z}_{n+2}\}$[3] where s is a state of $A^{'}$ and x is a tape head position.

Let G be an undirected graph as follows:

1. The vertices of G is the set of configurations of $A^{'}$ on input w of length n.
2. There is an edge between vertices (s_1, k) and $(s_2, k+1)$ if and only if $\mu^{'}(s_1, w_k^{'}) = s_2$ where $0 \leq k < n+1$ and $w_k^{'}$ is the k^{th} tape symbol.

The topology of G can be thought of as a matrix:

1. Each row contains the configurations having the same tape head positions and tape head positions are increased from bottom to top;
2. Each column contains the configurations having the same state and states are

[3] $\mathbb{Z}_n = \{0, 1, \cdots, n-1\}$.

placed according to the fixed order from left to right;

3. The edges can only be placed between configurations of consecutive rows.

Let G_0 be the connected component of G which contains the initial configuration $(s_0^{'}, 0)$. The following properties hold:

1. There is no cycle in G_0. Suppose that there is a cycle in G_0, then, it has one top vertex and one bottom vertex since edges are only placed between consecutive rows (not within the rows). There is only one edge between the bottom vertex and the vertex in the upper row but for a cycle, the bottom vertex must have at least two edges connected with the upper row, so there cannot be any cycle in G_0.
2. G_0 must contain exactly one of two vertices of $\{(s_{acc}, k+1), (s_{rej}, k+1)\}$, since $\mu^{'}$ is designed in order to enter just these two states at the end of the computation.
3. G_0 can be seen as a tree with the last configuration as the root and the deepest leaf is the initial configuration.
4. G_0 contains a simple path between initial configuration and last configuration and this path shows the exact computation steps of $A^{'}$ on input w.

Let $G_0^{'}$ be a subtree of G_0 including the vertices and edges on the simple path (between the root and deepest leaf) and all vertices and edges on the right–hand side of the simple path. M simulates A on input w by traversing $G_0^{'}$ in a reversible manner: M begins with the initial configuration, traverses all vertices in $G_0^{'}$ (once or twice) and halts with the last configuration. For each configuration (s, k) of $A^{'}$, there correspond two configurations of M: $(\overrightarrow{(s, +1)}, k)$ and $(\overleftarrow{(s, -1)}, k-1)$, which are to be interpreted as follows:

1. When M is in configuration $(\overrightarrow{(s, +1)}, k)$, this indicates that the subtree of $G_0^{'}$ rooted at vertex (s, k) has just been traversed;
2. When M is in configuration $(\overleftarrow{(s, -1)}, k-1)$, the subtree of $G_0^{'}$ rooted at vertex (s, k) is now about to be traversed.

Figure 2.1. An example subtree of $G_0^{'}$ rooted at vertex i

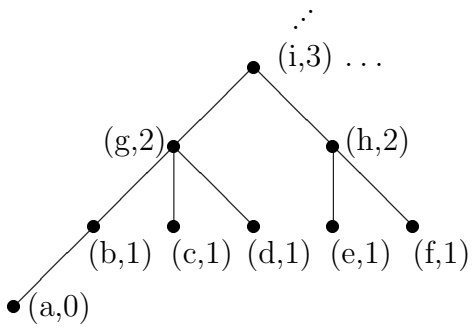

Figure 2.1 shows an example of a subtree of $G_0^{'}$ rooted at vertex i. M traverses the subtree as follows: $(\overrightarrow{(a,+1)},0)$, $(\overrightarrow{(b,+1)},1)$, $(\overleftarrow{(c,-1)},0)$, $(\overrightarrow{(c,+1)},1)$, $(\overleftarrow{(d,-1)},0)$, $(\overrightarrow{(d,+1)},1)$, $(\overrightarrow{(g,+1)},2)$, $(\overleftarrow{(h,-1)},1)$, $(\overleftarrow{(e,-1)},0)$, $(\overrightarrow{(e,+1)},1)$, $(\overleftarrow{(f,-1)},0)$, $(\overrightarrow{(f,+1)},1)$, $(\overrightarrow{(h,+1)},2)$, $(\overrightarrow{(i,+1)},3)$. There are mainly four cases:

1. When state is $\overleftarrow{(s,-1)}$:
   (a) if the vertex has a subtree ($J_{s,\sigma} \neq \emptyset$), traverse the leftmost child $\overleftarrow{(min(J_{s,\sigma}),-1)}$
   (b) if the vertex has no subtree ($J_{s,\sigma} = \emptyset$), traverse it $\overrightarrow{(s,+1)}$
2. When state is $\overrightarrow{(s,+1)}$:
   (a) if there is a vertex on the right–hand side ($s \neq max(I_{s,\sigma})$), traverse it $\overleftarrow{(succ(s,I_{s,\sigma}),-1)}$
   (b) if there is no vertex on the right–hand side ($s = max(I_{s,\sigma})$), traverse the parent $\overrightarrow{(\mu^{'}(s,\sigma),+1)}$

Note that $(a,0)$, $(b,1)$, $(g,2)$, and $(i,3)$ are on the simple path and the vertices and edges on lefthand side of the simple path in G_0 are not included in $G_0^{'}$ since they are not traversed by M.

As described above, M begins with the initial configuration, traverses all vertices and halts in the last configuration. If $w \in L(A)$ then the root of the simple path is $(s_{acc}, n+1)$ and so M accepts w. If $w \notin L(A)$ then the root of the simple path is $(s_{rej}, n+1)$ and so M rejects w. Therefore $L(A) = L(M)$. It is clear that M halts after $O(|w|)$ steps. □

Corollary 1. *For any regular language L, there exists a 2rfa which recognizes L in linear time.*

3. QUANTUM FINITE AUTOMATA

The idea of quantum computing begins with Feynman [27]. He argued that a quantum mechanical system can be simulated by a classical computer in exponential time. Therefore, a computer simulating a quantum mechanical system feasibly must be based on nonclassical features of quantum mechanics. In 1985, Deutsch [28] gave the first definition of a quantum Turing machine. [29–33] are milestones of quantum computing.

Quantum automata theory was introduced by [1, 3]. In this chapter, we give an overview of 1moqfa's (one–way measure–once quantum finite state automata), 1mmqfa's (one–way measure–many quantum finite state automata), 2qcfa's (two–way finite automata with quantum and classical states), 2qfa's (two–way quantum finite state automata), and 1.5qfa's (1.5–way quantum finite state automata).

3.1. Basics of Quantum Systems

In this section we give some basics of quantum systems (mainly from [5]) which will be used later.

Hilbert space is a mathematical framework suitable for describing the concepts, principles, processes and laws of quantum mechanics. Pure states of quantum systems are considered to be vectors of a Hilbert space. One can say that to each isolated quantum system corresponds a Hilbert space. Some even go farther by claiming that there is no reality on the quantum level; such a reality emerges only in the case of a measurement, and what we know about the quantum level are only computational procedures, expressed in terms of Hilbert space concepts, to compute evolutions of quantum systems and probabilities of the measurement outcomes.

Definition 17. *The conjugate of a complex number $c = a + bi$ is $a - bi$, denoted as $\overline{c}$ or c^*.*

Definition 18. *The conjugate of a matrix A is denoted as A^* and $A^*_{ij} = (A_{ij})^*$.*

Definition 19. *The hermitian conjugate or adjoint of a matrix A is $(A^T)^*$, denoted as $A^\dagger$.*

Definition 20. *A square matrix A is a unitary matrix if $A^\dagger = A^{-1}$.*

Definition 21. *An inner–product space H is a complex vector space, equipped with an inner product $\langle\cdot|\cdot\rangle : H \times H \to \mathbb{C}$ satisfying the following axioms for any vectors $\phi, \psi, \phi_1, \phi_2 \in H$, and any $c_1, c_2 \in \mathbb{C}$.*

$$\begin{aligned} \langle\psi|\phi\rangle &= \langle\phi|\psi\rangle^*, \\ \langle\psi|\psi\rangle &\geq 0 \text{ and } \langle\psi|\psi\rangle = 0 \text{ if and only if } \psi = 0, \\ \langle\psi|c_1\phi_1 + c_2\phi_2\rangle &= c_1\langle\psi|\phi_1\rangle + c_2\langle\psi|\phi_2\rangle, \end{aligned}$$

where $\langle\psi|\phi\rangle$ is the scalar product of ψ and ϕ and is defined by

$$\langle\psi|\phi\rangle = \sum_{i=1}^{n} \phi_i\psi_i^*, \quad |\psi\rangle = \begin{bmatrix} \psi_1 \\ \psi_2 \\ \vdots \\ \psi_n \end{bmatrix}, |\phi\rangle = \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_n \end{bmatrix}$$

The inner product introduces on H the norm (length)

$$||\psi|| = \sqrt{\langle\psi|\psi\rangle}.$$

Definition 22. *For any countable set D, let $l_2(D)$ be the space of all complex valued functions on D bounded by the so–called l_2–norm, i.e.*

$$l_2(D) = \{x \mid x : D \to \mathbb{C}, \left(\sum_{i\in D} x(i)x^*(i)\right)^{\frac{1}{2}} < \infty\}.$$

We say that $l_2(D)$ is a Hilbert space with respect to the inner product $\langle\cdot|\cdot\rangle : l_2(D) \times l_2(D) \to \mathbb{C}$.

Definition 23. *Two vectors ϕ and ψ of a Hilbert space are called orthogonal, notation*

$\psi \perp \phi$, if $\langle\phi|\psi\rangle = 0$. A set $S \subseteq H$ is orthogonal if all its elements are mutually orthogonal. S is orthonormal if it is orthogonal and all its elements have norm 1.

Definition 24. *A set $\mathcal{B}$ of vectors of a Hilbert space H forms an (orthonormal) basis of H if it is orthonormal and each vector ψ in H can be uniquely expressed in the form*

$$\psi = \sum_{\phi \in \mathcal{B}} \alpha_\phi \phi, \text{ with } \alpha_\phi \in \mathbb{C}.$$

Definition 25. *A subspace G of an inner–product space H is a subset of H closed under addition and scalar multiplication.*

An important property of Hilbert spaces is their decomposability into mutually orthogonal subspaces.

Theorem 7. *For each closed subspace W_1 of a Hilbert space H there exists a unique subspace $W_1^\perp$ such that $\langle\phi|\psi\rangle = 0$, whenever $\phi \in W_1$ and $\psi \in W_1^\perp$ and each $\psi \in H$ can be uniquely expressed in the form $\psi = \phi_1 + \phi_2$, with $\phi_1 \in W_1$ and $\phi_2 \in W_1^\perp$. In such a case we write $H = W_1 \oplus W_1^\perp$ and we say that W_1 and $W_1^\perp$ form a orthogonal decomposition of H.*

In a natural way we can make a generalization of an orthogonal decomposition

$$H = W_1 \oplus W_2 \oplus \cdots \oplus W_n,$$

of H into mutually orthogonal subspaces $W_1, W_2, \cdots, W_n$ such that each $\psi \in H$ has a unique representation as $\psi = \phi_1 + \phi_2 + \cdots + \phi_n$, with $\phi_i \in W_i, 1 \leq i \leq n$.

Definition 26. *A linear operator on a Hilbert space H is a linear mapping $A : H \to H$. The set of all linear operators of a Hilbert space H will be denoted $\mathcal{L}(H)$. $\mathcal{L}(H_1, H_2)$ will stand for the set of all linear operators from a Hilbert space H_1 into the Hilbert space H_2.*

An application of a linear operator A to a vector $|\psi\rangle$ is denoted by $A|\psi\rangle$.

In quantum physics in order to extract quantum information from a quantum system we have to observe the system – to perform a measurement of the system. Two basic types of observables are the so–called sharp observables and unsharp observables. To the sharp observables correspond the so–called projection measurements, or PV measurements, and to the unsharp observables correspond the so–called positive operator valued (POV) measurements (POVM).

An observable is a property of the physical system that can be measured. In quantum theory a (sharp) observable is a self–adjoint operator[4] .

In quantum mechanics, we consider the measurement of states mainly with respect to sharp observables. The numerical outcome of the measurement of a pure state $|\psi\rangle$ with respect to an observable A is one of the eigenvalues of A and the side effect of such a measurement is a "collapse" of $|\psi\rangle$ into a state $|\psi'\rangle$. In the measurement the eigenvalue λ_i is obtained with probability

$$Pr(\lambda_i) = ||P_i|\psi\rangle||^2 = \langle\psi|P_i|\psi\rangle,$$

and the new state $|\psi'\rangle$, into which $|\psi\rangle$ collapses, has the form

$$|\psi'\rangle = \frac{P_i|\psi\rangle}{\sqrt{\langle\psi|P_i|\psi\rangle}}.$$

This means that a measurement of $|\psi\rangle$ with respect to A irreversibly destroys $|\psi\rangle$, unless $|\psi\rangle$ is an eigenvector of A.

Definition 27. *The set $\mathcal{B} = \{i \mid i \text{ denotes a state}\}$ is a set of basis states, if for all $i, j \in \mathcal{B}, \langle i|j\rangle = \delta_{ij}$, i.e.*

$$\langle i|j\rangle = \begin{cases} 1 & \text{if } i = j; \\ 0 & \text{otherwise.} \end{cases}$$

[4] An operator T is self–adjoint if $T = T^*$.

We select all different configurations of a given quantum automaton as the set of basis states. The dimension of the corresponding Hilbert space equalst the maximum number of different configurations. Because of its quantum nature, any quantum automaton may be in more than one configuration with different amplitudes during its computation (i.e., the automaton may be in superpositions of more than one configuration).

Definition 28. *Let H_n be a Hilbert space of quantum automaton M with n different configurations and $\mathcal{B} = \{|c\rangle \mid c$ denotes a configuration$\}$ be a basis for H_n. A superposition of M can be expressed as*

$$|\psi\rangle = \sum_{i=1}^{k} \alpha_i |c_i\rangle, \quad 1 < k \leq n,$$

where c_i's are different configurations of M and α_i's are the coefficients of them. The probability of observing configuration c_i is $|\alpha_i|^2$ and the total probability is 1 ($|\alpha_1|^2 + |\alpha_2|^2 + \cdots + |\alpha_k|^2 = 1$).

Definition 29. *Let H_n be a Hilbert space of quantum automaton M with n different configurations. An observable $\mathcal{O} = \{E_1, E_2, \cdots, E_k\}$ is a set of disjoint and mutually orthogonal subspaces such that*

$$H_n = E_1 \oplus E_2 \oplus \cdots \oplus E_k$$

and an injection mapping $\mu : \{E_1, \cdots, E_k\} \to \mathbb{R}$.

Let $|\psi\rangle$ be a superposition of automaton M and $\mathcal{O} = \{E_1, E_2, \cdots, E_k\}$ be an observable. $|\psi\rangle$ can be expressed uniquely, as a linear superposition of its components (projections) along each of E_i's:

$$|\psi\rangle = \sum_{i=1}^{k} \alpha_i |\psi_{E_i}\rangle,$$

where $|\psi_{E_i}\rangle$ is the projection of $|\psi\rangle$ into E_i, and $\langle\psi_{E_i}|\psi_{E_i}\rangle = 1$ for all i.

An observation (measurement) of $|\psi\rangle$ by $\mathcal{O}$ has the following consequences:

1. One of the subspaces $E_1, \cdots, E_k$, say E_i, is selected and the value $\mu(E_i)$ is produced. The probability that a subspace E_i is selected is $|\alpha_i^2|$.
2. After observation, the superposition $|\psi\rangle$ "collapses" into the (renormalized) superposition $|\psi_{E_i}\rangle$.
3. The only classical information given by $\mathcal{O}$ is the value of the function μ. In the case $\mu(E_i) = i$, this is just information about which of the subspaces $E_1, \cdots, E_k$ was selected (or into which of subspaces the superposition $|\psi\rangle$ was projected). All information not in $|\psi_{E_i}\rangle$ is irreversibly lost.

During the computation of a quantum automaton, configurations may interfere with themselves. Since their coefficients may be positive or negative, the absolute value of summed coefficients may increase or decrase. If the interference leads the absolute value of coefficients to increase (or decrease), we name it constructive (or destructive) interference. The following example shows both kinds of interference:

Example 2. *Suppose that* $|\psi\rangle = \frac{1}{\sqrt{2}}|c_1\rangle + \frac{1}{\sqrt{2}}|c_2\rangle$ *and* L *is a linear operator such that*

$$\begin{aligned} L|c_1\rangle &= \tfrac{1}{\sqrt{2}}|c_3\rangle + \tfrac{1}{\sqrt{2}}|c_4\rangle \\ L|c_2\rangle &= \tfrac{1}{\sqrt{2}}|c_3\rangle - \tfrac{1}{\sqrt{2}}|c_4\rangle. \end{aligned}$$

So the calculation of $L|\psi\rangle$ *is*

$$\begin{aligned} L|\psi\rangle &= \tfrac{1}{\sqrt{2}}(\tfrac{1}{\sqrt{2}}|c_3\rangle + \tfrac{1}{\sqrt{2}}|c_4\rangle) + \tfrac{1}{\sqrt{2}}(\tfrac{1}{\sqrt{2}}|c_3\rangle - \tfrac{1}{\sqrt{2}}|c_4\rangle) \\ &= \tfrac{1}{2}|c_3\rangle + \tfrac{1}{2}|c_4\rangle + \tfrac{1}{2}|c_3\rangle - \tfrac{1}{2}|c_4\rangle \\ &= 1.|c_3\rangle + 0.|c_4\rangle \\ &= |c_3\rangle. \end{aligned}$$

As seen above after applying linear operator L *to* $|\psi\rangle$*, configuration* $|c_3\rangle$ *has interfered constructively and* $|c_4\rangle$ *has interfered destructively.*

3.2. One–Way Quantum Finite State Automata

3.2.1. Language Acceptance of One–Way Quantum Finite State Automata

It is defined as follows [9]:

1. A one–way quantum finite state automaton (1qfa) M is said to accept a language L with cut–point λ if for all $w \in L$ the probability of M accepting w greater than λ and for all $w \notin L$ the probability of M accepting w is at most λ.
2. A 1qfa M accepts L with bounded error if there exists an $\epsilon > 0$ such that for all $w \in L$ the probability of M accepting w greater than $\lambda + \epsilon$ and for all $w \notin L$ the probability of M accepting w is less than $\lambda - \epsilon$. We call ϵ the margin.

3.2.2. One–Way Measure–Once Quantum Finite State Automata

The definition of 1moqfa is as follows [9]:

Definition 30. *A measure–once quantum finite state automaton is defined by a 5–tuple*

$$M = (Q, \Sigma, \delta, q_0, F)$$

where Q is a finite set of states, Σ is a finite input alphabet with an end marker symbol \$, δ is the transition function

$$\delta : Q \times \Sigma \times Q \to \mathbb{C}$$

that represents the probability density amplitude that flows from state q to state $q^{'}$ upon reading symbol σ, the state q_0 is the initial configuration of the system, and F is the set of accepting states. For all states $q_1, q_2 \in Q$ and the symbol $\sigma \in \Sigma$ the function δ must satisfy the condition

$$\sum_{q^{'} \in Q} \overline{\delta(q_1, \sigma, q^{'})} \delta(q_2, \sigma, q^{'}) = \begin{cases} 1 & q_1 = q_2 \\ 0 & q_1 \neq q_2, \end{cases} \tag{3.1}$$

to ensure a unitary transition (see below).

Every input ends with the symbol \$. M measures its configuration at the end of the input; if it observes an accepting state then it accepts, otherwise it rejects.

The configuration of M is a linear superposition of states and is represented by an n–dimensional complex unit vector, where $n = |Q|$. This vector is denoted by

$$|\Psi\rangle = \sum_{i=1}^{n} \alpha_i |q_i\rangle \tag{3.2}$$

where $\{|q_i\rangle\}$ is the set of orthonormal basis vectors corresponding to the states of M. The coefficient α_i is the probability density amplitude of M being in state q_i. Since $|\Psi\rangle$ is a unit vector, it follows that $\sum_{i=1}^{n} \mid \alpha_i \mid^2 = 1$.

The transition function δ is represented by a set of unitary matrices $\{U_\sigma\}_{\sigma\in\Sigma}$ where U_σ represents the unitary transitions of M upon reading symbol σ. If M is in configuration $|\Psi\rangle$ and reads symbol σ then the new configuration of M is denoted by

$$|\Psi'\rangle = U_\sigma|\Psi\rangle = \sum_{q_i,q_j\in Q} \alpha_i \delta(q_i, \sigma, q_j)|q_j\rangle. \tag{3.3}$$

Measurement is represented by a diagonal zero–one projection matrix P where $P_{ii} = [q_i \in F]$. The probability of M accepting string x is defined by

$$P_M(x) = \langle\Psi_x|P|\Psi_x\rangle = |||\Psi_x\rangle||^2 \tag{3.4}$$

where $|\Psi_x\rangle = U(x)|q_0\rangle = U_{x_n}U_{x_{n-1}}\cdots U_{x_1}|q_0\rangle$.

Theorem 8. *A language L can be accepted by a 1moqfa with bounded error if and only if it can be accepted by a 1gfa.*

Proof. See [9]. □

Theorem 9. *Let M be a 1moqfa that accepts L with cut–point λ then:*

1. *There exists a 1pfa that accepts L with some cut–point λ'.*
2. *If M accepts L with bounded error, then there exists a 1pfa that accepts L with bounded error.*

Proof. See [9]. □

3.2.3. One–Way Measure–Many Quantum Finite State Automata

The definition of 1mmqfa is as follows [9]:

Definition 31. *A measure–many quantum finite state automaton is defined by a 6–tuple*

$$M = (Q, \Sigma, \delta, q_0, Q_{acc}, Q_{rej})$$

where Q is a finite set of states, Σ is a finite input alphabet with an end marker symbol \$, δ is a unitary function of the same form as for an 1moqfa, and the state q_0 is the initial configuration of M. The set Q is partitioned into three subsets: Q_{acc} is the set of halting accepting states, Q_{rej} is the set of halting rejecting states, and Q_{non} is the set of non–halting states.

The operation of an 1mmqfa is similar to that of an 1moqfa except that after every transition, M measures its configuration with respect to the three subspaces that correspond to the three subsets Q_{non}, Q_{acc}, and Q_{rej} such that $E_{non} = Span(\{|q\rangle \mid q \in Q_{non}\})$, $E_{acc} = Span(\{|q\rangle \mid q \in Q_{acc}\})$, and $E_{rej} = Span(\{|q\rangle \mid q \in Q_{rej}\})$. If the configuration of M is in E_{non}, then the computation continues; if the configuration is in E_{acc}, then M accepts, otherwise, it rejects. After each measurement, the superposition collapses into the measured subspace and is renormalized.

Just like 1moqfa's, the configuration of an 1mmqfa is represented by a complex

n–dimensional vector, the transition function is represented by unitary matrices, and measurement is represented by diagonal zero–one projection matrices that project the vector onto the respective subspaces.

During its computation, M may halt after some transitions. Therefore, it is useful to keep track of the cumulative accepting and rejecting probabilities with the current state as a triple $(|\Psi\rangle, p_{acc}, p_{rej})$, where p_{acc} and p_{rej} are the cumulative probabilities of accepting and rejecting. The evolution of M on reading symbol σ is denoted by

$$(P_{non}|\Psi^{'}\rangle, p_{acc} + ||P_{acc}|\Psi^{'}\rangle||^2, p_{rej} + ||P_{rej}|\Psi^{'}\rangle||^2) \tag{3.5}$$

where $|\Psi^{'}\rangle = U_\sigma|\Psi\rangle$, and P_{acc}, P_{rej}, and P_{non} are the diagonal zero–one projection matrices that project the configuration onto the non–halting, accepting and rejecting subspaces.

Theorem 10. *Let L be any language recognized by a 1mmqfa with bounded error. Then L is regular.*

Proof. See [3]. □

Theorem 11. *The language $L = \{a, b\}^*a$ cannot be recognized by a 1mmqfa with bounded error.*

Proof. See [3]. □

Theorem 12. *Let p be a prime. Any 1dfa recognizing the language $L_p = \{0^i \mid i \text{ is divisible by } p\}$ has to have at least p states, but for any $\epsilon > 0$ there is a 1mmqfa $\mathcal{A}_{p,\epsilon}$ with $O(\log(p))$ states accepting L_p with probability $\frac{1}{2} + \epsilon$.*

Proof. See [4]. □

3.3. Two–Way Finite Automata with Quantum and Classical States

The two– way finite automaton with quantum and classical states (2qcfa) was defined by Ambainis and Watrous [10]. Informally, a 2qcfa can be seen as a 2dfa that has access to a fixed–size quantum register, upon which it may perform quantum transformations and measurements. The transformations and measurements are determined by local descriptions of the classical portion of the machine, and the result of the measurements may determine the manner in which the classical part of the machine evolves. The formal definition as follows:

Definition 32. *A two–way finite automaton with quantum and classical states is specified by a 9–tuple*

$$M = (Q, S, \Sigma, \Theta, \delta, q_0, s_0, S_{acc}, S_{rej}),$$

where Q and S are finite quantum and classical state sets (respectively), Σ is a finite alphabet, Θ and δ are functions that specify the behavior of M as will be described below, $q_0 \in Q$ is the initial quantum state, $s_0 \in S$ is the initial classical state, and $S_{acc}, S_{rej} \subseteq S$ are the set of classical accepting and rejecting states (respectively). $\Gamma = \Sigma \cup \{\text{¢}, \$\}$ $(\text{¢}, \$ \notin \Sigma)$ is the tape alphabet, where ¢ is the left end–marker and $ is the right end–marker.

The function Θ specifies the evolution of the quantum portion of the internal state: for each pair $(s, \sigma) \in S \backslash (S_{acc} \cup S_{rej}) \times \Gamma$, $\Theta(s, \sigma)$ is an action to be performed on the quantum portion of the internal state of M. Each action $\Theta(s, \sigma)$ corresponds to either a unitary transformation or an orthogonal measurement.

The function δ specifies the evolution of the classical part of M (i.e., the classical part of the internal state and the tape head). In case $\Theta(s, \sigma)$ is a unitary transformation, $\delta(s, \sigma)$ is an element of $S \times \{-1, 0, 1\}$ specifying a new classical state and a movement of the tape head. In case $\Theta(s, \sigma)$ is a measurement, $\delta(s, \sigma)$ is a mapping from the set of possible results of the measurement to $S \times \{-1, 0, 1\}$ (again specify-

ing a new classical state and a tape head movement, this time one such pair for each outcome of the observation). That is, on each step, the quantum part of the internal state is first changed acording to $\Theta(s, \sigma)$, where s is the current classical state and σ is the currently scanned tape symbol, and then the classical internal state and tape head position are changed according to $\delta(s, \sigma)$ (along with the particular result obtained from $\Theta(s, \sigma)$ in case $\Theta(s, \sigma)$ is a measurement). It is assumed that δ is defined so that the tape head never acrosses the boundaries $(¢, \$)$.

Since the results obtained from each measurement $\Theta(s, \sigma)$ are probabilistic, the transitions among the classical parts of a given 2qcfa may be probabilistic as well. For each input x, we may define a probability $p_{acc}(x)$ that a given 2qcfa M eventually enters a classical accepting state, and a probability $p_{rej}(x)$ that M eventually enters a rejecting state. A given computation is assumed to halt when either an accepting or rejecting classical state is reached, so they are mutually exclusive.

3.3.1. A 2qcfa for the language L_{eq}

In this section, we will show that for any bound $\epsilon > 0$, there exists a 2qcfa which recognizes language L_{eq} with error bounded by ϵ in polynomial time [10]. Let M be a 2qcfa with a quantum register having two quantum states q_0 and q_1. Figure 3.1 is a high level description of M, where U_α means that the quantum state is rotated by angle $\alpha = \sqrt{2}\pi$ and $U_{-\alpha}$ means that the quantum state is rotated by angle $-\alpha$.

Lemma 5. *If the input w is of the form $a^m b^n$ and $m \neq n$, then M rejects after loop (I) (in Figure 3.1) with probability at least $\frac{1}{2(m-n)^2}$.*

Proof. Since the quantum state is set to $|q_0\rangle$ and U_α and $U_{-\alpha}$ are performed m and n times respectively, the superposition of the quantum register on symbol \$ is

$$\cos(\sqrt{2}(m-n)\pi)|q_0\rangle + \sin(\sqrt{2}(m-n)\pi)|q_1\rangle$$

The probability of observing $|q_1\rangle$ is $\sin^2(\sqrt{2}(m-n)\pi)$. We bound this probability from

Figure 3.1. Description of the 2qcfa M, given the input string w

Check (classically) whether the input w is of form $a^m b^n; m, n > 0$. If not, REJECT.
Otherwise, repeat ad infinitum:
Move the tape head on symbol ¢ and set the quantum state to $\|q_0\rangle$.
While the currently scanned symbol is not \$, do the following: (I)
If the currently scanned symbol is "a", perform U_α on the quantum register
If the currently scanned symbol is "b", perform $U_{-\alpha}$ on the quantum register
Measure the quantum state (on symbol \$). If the result is $\|q_1\rangle$, REJECT.
Two times repeat: (II)
Move the tape head on symbol ¢.
Move the tape head one square to the right
While the currently scanned symbol is not ¢ or \$, do the following (III)
Simulate a coin flip. If the result is "heads", move right. Otherwise, move left.
If both times the process ends at the right end–marker \$, do:
Simulate $k(= 1 + \lceil \log \frac{1}{\epsilon} \rceil)$ coin–flips. If all results are "heads", ACCEPT.

below.

Let d be the closest integer to $\sqrt{2}(m-n)$. Assume that $d < \sqrt{2}(m-n)$ (the other case is symmetric). Then, $d \leq \sqrt{2(m-n)^2 - 1}$ since d^2 is an integer and $2(m-n)^2 - 1$ is the largest integer that is smaller than $(\sqrt{2}(m-n))^2$. We have

$$\begin{aligned}
&(\sqrt{2}(m-n) - \sqrt{2(m-n)^2-1})(\sqrt{2}(m-n) + \sqrt{2(m-n)^2-1}) \\
&= 2(m-n)^2 - 2(m-n)^2 + 1 \\
&= 1
\end{aligned}$$

and

$$\begin{aligned}
&\sqrt{2}(m-n) - d \\
&\geq \sqrt{2}(m-n) - \sqrt{2(m-n)^2-1} \\
&= \frac{1}{\sqrt{2}(m-n) + \sqrt{2(m-n)^2-1}} \\
&> \frac{1}{2\sqrt{2}(m-n)}
\end{aligned}$$

Since d is the closest integer, we have $0 < \sqrt{2}(m-n) - d < \frac{1}{2}$. It is a well–known fact

that for any $x \in [0, \frac{1}{2}]$, $\sin(x\pi) \geq 2x$ (sin is concave in this interval). Therefore,

$$\begin{aligned}
&\sin^2(\sqrt{2}(m-n)\pi) \\
&= \sin^2((\sqrt{2}(m-n)-d)\pi) \\
&\geq 4(\sqrt{2}(m-n)-d)^2 \\
&\geq 4(\tfrac{1}{2\sqrt{2}(m-n)})^2 \\
&= \tfrac{1}{2(m-n)^2}.
\end{aligned}$$

□

Lemma 6. *Each execution of (II) (in Figure 3.1) leads to acceptance with probability* $\frac{1}{2^k(m+n+1)^2}$.

Proof. Each loop (III) (in Figure 3.1) is just a random walk starting at tape position 1 (the first a of $a^m b^n$) and ending at position 0 (the left end marker ¢) or position $m+n+1$ (the right end–marker \$). It is a standard result in probability theory (see Chapter 14.2 of [34]) that the probability of reaching position $m+n+1$ is exactly $\frac{1}{m+n+1}$. The probability of successfully repeating it twice and getting k "heads" from k coin flips is $\frac{1}{2^k(m+n+1)^2}$. □

If w is of the form $a^m b^n$ and $m = n$, then the loop (I) always results in $|q_0\rangle$ (in quantum state) and so the probability of M accepting after $c|w|^2$ executions of loop (II) is

$$1 - \left(1 - \frac{1}{2^k(m+n+1)^2}\right)^{c|m+n|^2} \tag{3.6}$$

and this can be made arbitrarily close to 1 by selecting an appropriate constant c.

If $|w|$ is of the form $a^m b^n$ and $m \neq n$, then M rejects after (I) with probability $p_{rej} \geq \frac{1}{2(m-n)^2}$ and accepts after (II) with probability $p_{acc} = \frac{1}{2^k(m+n+1)^2} \leq \frac{\epsilon}{2(m+n+1)^2}$.

$$\frac{p_{acc}}{p_{rej}} \leq \frac{\epsilon(m-n)^2}{(m+n+1)^2}$$

$$\frac{p_{acc}}{p_{rej}} < \epsilon \tag{3.7}$$

So, the probability of rejecting is

$$\begin{aligned}
&\textstyle\sum_{k\geq 0}(1p_{acc})^k(1p_{rej})^k p_{rej} \\
&= \textstyle\sum_{k\geq 0}(1-(p_{acc}+p_{rej}-p_{acc}p_{rej}))^k p_{rej} \\
&= \tfrac{p_{rej}}{p_{acc}+p_{rej}-p_{acc}p_{rej}} \\
&> \tfrac{p_{rej}}{p_{acc}+p_{rej}} \\
&= \tfrac{1}{\frac{p_{acc}}{p_{rej}}+1} \\
&> \tfrac{1}{1+\epsilon} \\
&> 1-\epsilon.
\end{aligned}$$

In both cases, the expected number of iterations of (I) and (II) before M accepts or rejects is $O(|w|^2)$ (because, in every iteration, M accepts or rejects with probability at least $\frac{c}{|w|^2}$). Loop (I) takes $O(|w|)$ time and each random walk in (II) takes $O(|w|^2)$ time. Hence, the expected running time of M is $O(|w|^4)$.

Theorem 13. *For any $\epsilon > 0$, there is a 2qcfa that accepts any $w \in L_{eq}$ with certainty and rejects $w \notin L_{eq}$ with probability at least $1-\epsilon$ and halts in expected time $O(|w|^4)$.*

Proof. By setting $k = 1+\lceil \log(\frac{1}{\epsilon}) \rceil$, the 2qcfa M described above is such an automaton. □

3.4. Two–Way Quantum Finite State Automata

Like their classical counterparts, two–way quantum finite state automata (2qfa's) have finite state controls and read–only input tapes with two–way tape heads.

Definition 33. *A 2qfa is a 6–tuple $M = (Q, \Sigma, \delta, q_0, Q_{acc}, Q_{rej})$. Q is a finite set of states, $q_0 \in Q$ is the initial state, $Q_{acc} \subset Q$ is the set of accepting states, and $Q_{rej} \subset Q$ is the set of rejecting states. Q_{acc} and Q_{rej} are disjoint, and their union is the set of halting states. The elements of $Q \setminus (Q_{acc} \cup Q_{rej})$ are non–halting states. Σ is the input*

alphabet. The symbols $\{\¢, \$\} \notin \Sigma$ *are used to mark the left and right ends of the input string, respectively.* $\Gamma = \Sigma \cup \{\¢, \$\}$ *is the tape alphabet.* δ *is the transition function, which governs the behaviour of* M*, and is explained below.*

The configurations of a 2qfa running on a string w are pairs of the form $|q, x\rangle$, where q is the state and x is the head position. Initially, the head is on the left end–marker, and the machine is in the configuration $|q_0, 0\rangle$. At later steps of the computation, due to its quantum nature, the machine may exist in superpositions of more than one configuration. It is sometimes useful to visualize such superpositions of multiple configurations as a snapshot of the machine running in multiple parallel computational paths or branches. Since the coefficients (amplitudes) in these superpositions can be negative or even complex numbers, these paths may interfere with each other in ways quite unlike what can happen with classical probabilistic machines. In each step, the machine makes two linear operations: The first one is a unitary transformation of the current superposition according to δ, and the second one is an orthogonal measurement of the new configuration to see whether the machine has accepted, rejected, or not halted yet. In all 2qfa's described here, every transition entering the same state involves the tape head moving in the same direction (left, right, or stationary). We represent this feature of a state q using the appropriate one of the notations $\overleftarrow{q}$, $\overrightarrow{q}$, or $\downarrow q$ for this state in the machine description. With this simplification, a syntactically correct 2qfa can be specified easily by just providing a unitary operator $V_\sigma : l_2(Q) \rightarrow l_2(Q)$ for each symbol $\sigma \in \Gamma$. More formally,

$$\delta(q, \sigma, q', d) = \langle q'|V_\sigma|q\rangle$$

is the amplitude with which the machine currently in state q and scanning symbol σ will jump to state q' and move the head in direction d. Here, $\langle q'|V_\sigma|q\rangle$ denotes the coefficient of $|q'\rangle$ in $V_\sigma|q\rangle$ and $d \in \{-1, 0, 1\}$ is the direction of the tape head determined by q'.

The observable describing the measurement process that is executed at each step

of the computation is designed so that the outcome of any observation is "accept", "reject", or "non–halting". The machine continues running from a normalized superposition of non–halting configurations until no non–halting configurations remain. The overall acceptance probability of the input string is the sum of the probabilities of the computational branches which end with accepting configurations.

Theorem 14. *For any regular language L, there exists a 2qfa which recognizes L in linear time.*

Proof. 2rfa's are a special case of 2qfa's whose transition amplitudes only take the values 0 and 1. See Section 2.5.1. □

3.4.1. A 2qfa for the language L_{eq}

In this section, we will show that for any bound $\epsilon > 0$, there exists a 2qfa which recognizes the non–regular language L_{eq} with error bounded by ϵ in linear time [3].

For each integer N, we define $M_N = (Q, \Sigma, \delta, q_0, Q_{acc}, Q_{rej})$ as follows:
$Q = \{ \overrightarrow{q}_0, \overleftarrow{q}_1, \overrightarrow{q}_2, \downarrow q_3\} \cup \{ \overleftarrow{r}_{j,0} \mid 1 \leq j \leq N\}$
$\cup \{ \downarrow r_{j,k} \mid 1 \leq j \leq N, 1 \leq k \leq max(j, N-j+1)\}$
$\cup \{ \downarrow s_j \mid 1 \leq j \leq N\}$,
$Q_{acc} = \{s_n\}$ and $Q_{rej} = \{q_3\} \cup \{s_j \mid 1 \leq j < N\}$. $\Sigma = \{a, b\}$. Let each V_σ act as indicated in Figure 3.2 and extend each to be unitary.

According to this construction, the machine first attempts to validate that the input string is of the form $\{a^m b^n | \ m, n > 0\}$, rejecting and halting otherwise. If this first stage is passed without rejection, the tape head is over the right end–marker \$, and the computation branches into $N = \lceil \frac{1}{\epsilon} \rceil$ paths, in each of which the head travels with different speeds on a's and b's towards the left end–marker: In path number j, the head waits j steps over each a and $N - j + 1$ steps over each b before moving left. When the tape head arrives at the end symbol ¢, each path splits according to the Quantum Fourier Transform (QFT), jumping to a set consisting of $N - 1$ reject states

Figure 3.2. Specification of the transition function of M_N

$V_{¢}\|\overrightarrow{q}_0\rangle = \|\overrightarrow{q}_0\rangle$	$V_{\$}\|\overrightarrow{q}_0\rangle = \|\downarrow q_3\rangle$
$V_{¢}\|\overleftarrow{q}_1\rangle = \|\downarrow q_3\rangle$	$V_{\$}\|\overrightarrow{q}_2\rangle = \frac{1}{\sqrt{N}}\sum_{j=1}^{N}\|\overleftarrow{r}_{j,0}\rangle, 1 \leq j \leq N$
$V_{¢}\|\overleftarrow{1}_{j,0}\rangle = \frac{1}{\sqrt{N}}\sum_{l=1}^{N}(e^{\frac{2\pi i}{N}jl})\|\downarrow s_l\rangle, 1 \leq j \leq N$	
$V_a\|\overrightarrow{q}_0\rangle = \|\overrightarrow{q}_0\rangle$	$V_b\|\overrightarrow{q}_0\rangle = \|\overleftarrow{q}_1\rangle$
$V_a\|\overleftarrow{q}_1\rangle = \|\overrightarrow{q}_2\rangle$	$V_b\|\overrightarrow{q}_2\rangle = \|\overrightarrow{q}_2\rangle$
$V_a\|\overrightarrow{q}_2\rangle = \nexists q_3$	$V_b\|\overleftarrow{r}_{j,0}\rangle = \|\downarrow r_{j,N-j+1}\rangle, 1 \leq j \leq N$
$V_a\|\overleftarrow{r}_{j,0}\rangle = \|\downarrow r_{j,j}\rangle, 1 \leq j \leq N$	$V_b\|\downarrow r_{j,k}\rangle = \|\downarrow r_{j,k-1}\rangle, 1 < k \leq N-j+1, 1 \leq j \leq N$
$V_a\|\downarrow r_{j,k}\rangle = \|\downarrow r_{j,k-1}\rangle, 1 < k \leq j, 1 \leq j \leq N$	$V_b\|\downarrow r_{j,1}\rangle = \|\overleftarrow{r}_{j,0}\rangle, 1 \leq j \leq N$
$V_a\|\downarrow r_{j,1}\rangle = \|\overleftarrow{r}_{j,0}\rangle, 1 \leq j \leq N$	

and one accept state. If the numbers of a's and b's are not equal, each computation path arrives at ¢ at a different time, and therefore branches to the accept state with $\frac{1}{N}$ probability, and to some reject state with $1 - \frac{1}{N}$ probability. Therefore, the overall acceptance probability is also $\frac{1}{N}$, and the overall rejection probability is $1 - \frac{1}{N}$. If the numbers of a's and b's are equal, all paths reach ¢ at the same time. The transitions implementing the QFT are arranged in such a way that, in this case, all the rejecting configurations have amplitude zero, and the machine accepts with probability 1.

For error bound ϵ and input string w, a machine built according to this method has $O((\frac{1}{\epsilon})^2)$ states and halts within $O(\frac{1}{\epsilon}|w|)$ steps.

3.5. 1.5–Way Quantum Finite State Automata

The 1.5–way quantum finite state automaton (1.5qfa) was defined by Amano and Iwama [35]. It is a special case of 2qfa which cannot move its head to the left. Amano and Iwama conjectured that, contrary to the case of 2qfa's, there are regular languages that cannot be accepted by any 1.5qfa and this is still an open problem [36]. On the other hand, it was shown that 1.5qfa's can recognize some non–context free languages [37].

Definition 34. *A 1.5–way quantum finite state automaton is defined as a 6–tuple*

$$M = (Q, \Sigma, \delta, q_0, Q_{acc}, Q_{rej})$$

where Q is the set of quantum states, Σ is the set of input symbols including the left and right end–marker $\{\text{¢}, \$\}$ respectively, δ is the quantum state transition function ($\delta : (Q \times \Sigma \times Q \times \{0, 1\}) \rightarrow \mathbb{C}$), q_0 is the initial quantum state, Q_{acc} is the set of accepting states, and Q_{rej} is the set of rejecting states, where $Q_{acc} \cap Q_{rej} = \emptyset$.

Theorem 15. *The non–context–free language $L = \{a^n d b^n d c^n \mid n \geq 0\}$ can be recognized by a 1.5qfa with probability less than $\frac{2}{3}$.*

Proof. See [37]. □

4. EFFICIENT CONSTRUCTION OF TWO–WAY QUANTUM FINITE AUTOMATA

4.1. Previous Work

4.1.1. N–way Branching with a Single Pass

This is the method described in Section 3.4.1. Machines constructed with respect to this method have $O((\frac{1}{\epsilon})^2)$ states and halt after $O((\frac{1}{\epsilon})|w|)$ steps, where w is the input string.

4.1.2. Two–way Branching in Multiple Passes

Watrous [2] presents an alternative approach to recognize L_{eq}: After making sure that the input is of the form $\{a^m b^n |\ m, n > 0\}$, the head moves to the last a. At that point, the computation branches to two paths. One path starts on the first b and walks over the b's to the right end–marker, while the other path starts on the last a and walks over the a's to the left end–marker. In both paths, the head then returns to the "border" between the a's and the b's. Upon crossing the border, both paths perform a QFT with $N = 2$ as in Section 3.4.1. If the two targets of this QFT were both halting states, as described in Section 3.4.1, this machine would clearly have the unacceptably high (one–sided) error bound of $\frac{1}{2}$, so one target is a reject state, whereas the other one (which is observed with probability 1 when the two paths perform the QFT at the same time) is a non–halting state. Each path again splits into two in this state, and the process described above takes place again and again to decrease ϵ to any desired positive value.

For error bound ϵ and input string w, a machine built according to this method has $O((\frac{1}{\epsilon})^2)$ states and halts within $O((\frac{1}{\epsilon})^2|w|)$ steps.

4.1.3. Two–way Branching in Multiple Passes with Collision Avoidance

One source of inefficiency in the algorithm described in Section 4.1.2 is the fact that different computation paths with the same length may interfere even during the processing of non–members of L_{eq}. For example, the path which walks over the a's in the first round and the b's in the second round would reach the second QFT at exactly the same time as the path which walks over the b's in the first round and the a's in the second round, and they would therefore avoid the reject state. Atak [38] modified Watrous' algorithm to prevent this interference by incorporating waiting states which cause the head to move slower and slower in each new round to every round except the first. We calculated the general formula for the required number of waiting steps for any value of ϵ: If the number of waiting steps in each round is taken from the list $W = (0, 1, 2^2 - 2, ..., 2^{p-1} - 2)$ (where $p > 1$), the value of ϵ is decreased to $\frac{1}{2^p}$. (See Section 4.3 for a detailed analysis of a similar issue.) Machines built according to this method have $O(\frac{1}{\epsilon})$ states, and their runtime is $O((\frac{1}{\epsilon})|w|)$ for input string w. This approach yields more succinct machines than [3] and [2].

4.2. The Quotient Comparison Method

In this section, we describe a method to construct 2qfa's which recognize L_{eq}, and whose runtimes do not depend on the error bound ϵ.

For each integer $N > 1$ and $k > 1$, we define $M_{k,N} = (Q, \Sigma, \delta, q_0, Q_{acc}, Q_{rej})$ as follows:

$Q = \{\overrightarrow{q}_0, \overleftarrow{q}_1, \overrightarrow{q}_2, \downarrow q_3\}$

$\cup \{\overleftarrow{m}_0, \overleftarrow{m}_1, ..., \overleftarrow{m}_{k-1}\} \cup \{\Downarrow r_1, \downarrow r_2, ..., \downarrow r_{k-1}\}$

$\cup \{\overrightarrow{m}_{j,0}, \overrightarrow{m}_{j,1}, ..., \overrightarrow{m}_{j,k-1}\} \cup \{\Downarrow w_{j,1}, \downarrow w_{j,2}, ..., \downarrow w_{j,l}\}$

$\cup \{\Downarrow s_1, \downarrow s_2, ..., \downarrow s_N\}$, where $1 \leq j \leq N, 1 \leq l \leq max(j, N - j + 1)$

$Q_{acc} = \{s_N\}$ and $Q_{rej} = \{q_3\} \cup \{r_i | 1 \leq i \leq k - 1\} \cup \{s_j | 1 \leq j \leq N - 1\}$. $\Sigma = \{a, b\}$. Let each V_σ act as indicated in Figure 4.1, and extend each to be unitary. Figure 4.2 is a high level description of the working of the machine.

Theorem 16. *The 2qfa $M_{k,N}$ recognizes the language $L_{eq} = \{a^n b^n | \ n > 0\}$ with one–*

Figure 4.1. Specification of the transition function of $M_{k,N}$

Stage 1	Stage 2	Stage 3
$V_{¢}\|\overrightarrow{q}_0\rangle = \|\overrightarrow{q}_0\rangle$	$V_{¢}\|\overleftarrow{m}_i\rangle = \|\downarrow r_i\rangle,\ 1 \le i \le k-1$	$V_{¢}\|\overrightarrow{m}_0\rangle = \frac{1}{\sqrt{N}}\sum_{j=1}^{N}\|\overrightarrow{m}_{j,0}\rangle$
$V_{¢}\|\overleftarrow{q}_1\rangle = \|\downarrow q_3\rangle$	$V_a\|\overleftarrow{m}_i\rangle = \|\overleftarrow{m}_{i-1(\mathrm{mod}\ k)}\rangle$	$V_a\|\overrightarrow{m}_{j,i}\rangle = \|\overrightarrow{m}_{j,i+1}\rangle,\ 0 \le i \le k-2$
$V_a\|\overrightarrow{q}_0\rangle = \|\overrightarrow{q}_0\rangle$	$V_b\|\overleftarrow{m}_i\rangle = \|\overleftarrow{m}_{i+1(\mathrm{mod}\ k)}\rangle$	$V_a\|\overrightarrow{m}_{j,k-1}\rangle = \|\downarrow w_{j,1}\rangle$
$V_a\|\overleftarrow{q}_1\rangle = \|\overrightarrow{q}_2\rangle$		$V_a\|\downarrow w_{j,i}\rangle = \|\downarrow w_{j,i+1}\rangle,\ 1 \le i \le N-j$
$V_a\|\overrightarrow{q}_2\rangle = \|\downarrow q_3\rangle$		$V_a\|\downarrow w_{j,N-j+1}\rangle = \|\overrightarrow{m}_{j,0}\rangle$
$V_b\|\overrightarrow{q}_0\rangle = \|\overleftarrow{q}_1\rangle$		$V_b\|\overrightarrow{m}_{j,i}\rangle = \|\overrightarrow{m}_{j,i-1}\rangle,\ 1 \le i \le k-1$
$V_b\|\overrightarrow{q}_2\rangle = \|\overrightarrow{q}_2\rangle$		$V_b\|\overrightarrow{m}_{j,0}\rangle = \|\downarrow w_{j,1}\rangle$
$V_{\$}\|\overrightarrow{q}_0\rangle = \|\downarrow q_3\rangle$		$V_b\|\downarrow w_{j,i}\rangle = \|\downarrow w_{j,i+1}\rangle,\ 1 \le i \le j-1$
$V_{\$}\|\overrightarrow{q}_2\rangle = \|\overleftarrow{m}_0\rangle$		$V_b\|\downarrow w_{j,j}\rangle = \|\overrightarrow{m}_{j,k-1}\rangle$
		$V_{\$}\|\overrightarrow{m}_{j,0}\rangle = \frac{1}{\sqrt{N}}\sum_{l=1}^{N}(e^{\frac{2\pi i}{N}jl})\|\downarrow s_l\rangle$

sided error bound $\frac{1}{N}$ *for any* k*. If* $k = N$*, then the machine halts in* $O(|w|)$ *steps, and the size of its state set is* $O(N^2)$*.*

Proof. We will give the detailed specification of the machine. For simplicity, we divide the machine into three stages (as seen in Figure 4.2).

States q_0 through q_3 check the membership of the input in $\{a^m b^n |\ m, n > 0\}$, in Stage 1. States m_0 through m_{k-1} and r_1 through r_{k-1} check $m = n \pmod k$, in Stage 2. Note that the quantum nature of the machine requires $k-1$ different reject states to be used in this stage, in contrast to the single reject state that is sufficient in a classical machine.

In the beginning of Stage 3, if the machine is still running, we know that $m = n \pmod k$, so m and n can be written as $m = xk + c$ and $n = yk + c$ where $(0 \le c \le k-1)$. We will use the fact that $x = y$ if and only if $m = n$, and therefore a comparison of the smaller values of x and y is sufficient to determine membership of the input in

Figure 4.2. Description of the quotient comparison method, given the input string w

---

1 Scan the input from ¢ to \$ to check whether it is of the form $a^m b^n$; $m, n > 0$. If not, REJECT.

2 Scan the input from \$ to ¢ to check whether $m = n \pmod{k}$. If not, REJECT.

3 While on ¢, branch the computation to N different paths (with amplitude $\frac{1}{\sqrt{N}}$). For each computation path j $(1 \leq j \leq N)$, scan the input from ¢ to \$ as follows:

   While scanning the a's, enter an $(N - j + 1)$–step waiting cycle at every k^{th} symbol (totally $\lfloor \frac{m}{k} \rfloor$ times), counting the a's modulo k simultaneously. Let c be the value of this count at the end of the a region.

   While scanning the b's, enter a j–step waiting cycle at the $(c+1)^{st}$ b, and at every k^{th} b after that (totally $\lfloor \frac{n}{k} \rfloor$ times).

   Upon reaching \$, branch to N halting states according to the QFT as in [3].

L_{eq}.

When walking over the a's, the machine enters the wait states at the squares numbered $(k, 2k, ..., xk)$ $(\lfloor \frac{m}{k} \rfloor = x$ times) for each path j. While traversing the b's, the machine enters the wait states at squares numbered $(m + c + 1 + 0k, m + c + 1 + 1k, ..., m + c + 1 + (y-1)k)$ $(\lfloor \frac{n}{k} \rfloor = y$ times) for each path j. Upon reaching the right end–marker, it is easily seen that each path j is in the state $m_{j,0}$.

At this point, each path again branches to halting states according to the QFT. As in [3], if all paths make the QFT at the same time, rejecting configurations cancel each other, and only the accept configuration remains, so the string is accepted with probability 1. If all paths make the QFT at different times, then the input string is accepted with probability $\frac{1}{N}$, and rejected with probability $1 - \frac{1}{N}$.

The number of steps taken by the j^{th} path in this stage is:

$$steps_j = |w| + xN + yj - xj + x + 1 \tag{4.1}$$

If $m = n$, then $x = y$, and $steps_j = 2m + xN + x + 1$. Since the number of steps in this case is independent of j, all computation paths make the QFT simultaneously. Therefore, the input string is accepted with probability 1.

If $m \neq n$, then $x \neq y$, and the length of path j depends on j. If the lengths of paths j_1 and j_2 are equal, we have

$$\begin{aligned} steps_{j_1} &= steps_{j_2} \\ |w| + xN + yj_1 - xj_1 + x + 1 &= |w| + xN + yj_2 - xj_2 + x + 1 \\ (y - x)j_1 &= (y - x)j_2, \end{aligned}$$

which is true only if $j_1 = j_2$ since $y - x \neq 0$, so all different paths make the QFT in different steps when the input is not a member of L_{eq}. Therefore, the input is accepted with probability $\frac{1}{N}$ and rejected with probability $1 - \frac{1}{N}$.

As for an analysis of the runtime; Stage 1 and Stage 2 take a total of at most $2|w| + 5$ steps. Stage 3 takes at most $|w| + xN + max(j(y - x)) + x + 1$ steps. So, the overall running time is at most $3|w| + 6 + max(x, y)N + min(x, y)$. When we set $k = N$, this value is bounded from above by $4|w| + 6$; therefore, the running time of $M_{N,N}$ is $O(|w|)$. Moreover, setting $N = \lceil \frac{1}{\epsilon} \rceil$ for any desired positive error bound ϵ, it is easily calculated that the size of the state set of $M_{N,N}$ is $O((\frac{1}{\epsilon})^2)$. □

Clearly, one can reduce the constant hidden in the O–notation for the runtime to 2 by simply running the three stages of the algorithm described above in parallel.

4.3. State–efficient Construction Methods

The methods to be described in this section produce machines which are superior in both the description size and runtime measures to those of [3]. They involve the repeated application of the quotient comparison procedure in multiple passes over the input string, with the aim of reducing the error probability to the desired value. As mentioned in Chapter 1, the divisors used in these passes are selected carefully to avoid unwanted interference between computational paths.

The three constructions to be described here can be seen as different instantiations of the following common template, parameterized with two lists of integers, $K = (k_1, k_2, \cdots, k_r)$ and $B = (N_1, N_2, \cdots, N_r)$: (Consult Figures 4.1 and 4.2 to recall the details of the stages of the basic quotient comparison procedure named in the description below.)

I Start as usual by checking whether w is of the form $a^m b^n$; $m, n > 0$. If not, REJECT. (Stage 1 from the previous section)

II Zig–zag on the tape to perform r rounds of Stage 2 to check whether $m = n \pmod{k_i}$ in the i^{th} round. If any check fails, REJECT.

III Perform r rounds of Stage 3. The i^{th} round branches to N_i paths and uses a divisor of value k_i. In all of these rounds (except the last), of the N_i targets of the QFT at the end, the state whose observation probability is 1 if the input is a member of L_{eq}, and $\frac{1}{N_i}$ otherwise, is not an accept state, but a non–halting state. Computation paths that arrive at this state split into N_{i+1}, and Stage 3 is performed again, this time with the head moving in the reverse direction and with the divisor set to k_{i+1}. The QFT at the end of the last round branches to one accept and $N_r - 1$ reject states as dictated in [3], ending the computation.

It is easily seen that if the input is a member of L_{eq}, then all computation paths in the same pass reach ¢ (\$) at the same step, and the input is accepted with probability 1 at the end. On the other hand, if the input is not a member of L_{eq} and Stages I and II are passed without rejection, we would like all pairs (CP_1, CP_2) of distinct

computational paths to finish each round of Stage III at different times in order to avoid interference which would increase the overall incorrect acceptance probability. If we could manage to avoid all such collisions, the one–sided error bound (ϵ) would equal $\frac{1}{N_1 N_2 \cdots N_r}$.

CP_1 and CP_2 are distinguished by the branches that they follow in the splits at the beginning of each round of Stage III. Let us say that CP_1 and CP_2 select the $j_{l,1}^{th}$ and $j_{l,2}^{th}$ branches respectively in round l (where $1 \leq j_{l,1}, j_{l,2} \leq N_l$). The difference between the total lengths (i.e. the number of steps since the beginning of the computation) of CP_1 and CP_2 when they reach the end of round i can be found easily, making use of Equation 4.1 from Section 4.2, to be:

$$steps_{CP_1} - steps_{CP_2} = (n-m)(\frac{j_{1,1} - j_{1,2}}{k_1} + \frac{j_{2,1} - j_{2,2}}{k_2} + \cdots + \frac{j_{i,1} - j_{i,2}}{k_i}) \qquad (4.2)$$

Since we are working on the case where $m \neq n$, in order to avoid unwanted interference, one has to select B and K so that the second parenthesis in the expression above cannot equal 0 for any round i. Some of the terms in that parenthesis may be 0, since some $j_{l,1}$ may equal the corresponding $j_{l,2}$, but this cannot be the case for all $(j_{l,1}, j_{l,2})$ pairs, since CP_1 and CP_2 are different computational paths. Concentrating on the nonzero terms, we see that the sum of all the positive terms must not equal the absolute value of the sum of all the negative terms for this expression to have a nonzero value. The following lemma will be used to ensure this condition for all the methods to be presented in this section.

Lemma 7. *Let $P = \{p_1, p_2, \cdots, p_s\}$ and $Q = \{q_1, q_2, \cdots, q_t\}$ be any two disjoint sets of integers greater than one, such that all members of $P \cup Q$ are relatively prime. Then $\sum_{i=1}^{s} \frac{c_i}{p_i} \neq \sum_{j=1}^{t} \frac{d_j}{q_j}$, whenever the $c_i's$, and the $d_j's$ are integers, and $0 < c_i < p_i$, $0 < d_j < q_j$.*

Proof. Define F, G, F_i, and G_j as follows ($1 \leq s, 1 \leq t$):

$$F = p_1 p_2 \cdots p_s \text{ and } F_i = \frac{F}{p_i}$$

$$G = q_1 q_2 \cdots q_t \text{ and } G_j = \frac{G}{q_j}$$

Suppose that $\sum_{i=1}^{s} \frac{c_i}{p_i} = \sum_{j=1}^{t} \frac{d_j}{q_j}$. Rewriting, one obtains

$$\begin{gathered}(\tfrac{c_1}{p_1} + \tfrac{c_2}{p_2} + \cdots + \tfrac{c_s}{p_s}) = (\tfrac{d_1}{q_1} + \tfrac{d_2}{q_2} + \cdots + \tfrac{d_t}{q_t}), \\ (\tfrac{c_1F_1 + c_2F_2 + \cdots + c_sF_s}{F}) = (\tfrac{d_1G_1 + d_2G_2 + \cdots + d_tG_t}{G}),\end{gathered}$$

which yields

$$G(c_1F_1 + c_2F_2 + \cdots + c_sF_s) = F(d_1G_1 + d_2G_2 + \cdots + d_tG_t).$$

Dividing both sides by p_1, we get

$$(\tfrac{Gc_1F_1}{p_1}) + G(\tfrac{c_2F_2 + \cdots + c_sF_s}{p_1}) = F_1(d_1G_1 + d_2G_2 + \cdots + d_tG_t).$$

The right–hand side of this equation is an integer, but the left–hand side is not, which is a contradiction. □

So one way of guaranteeing that a multi–pass 2qfa matching the template described at the beginning of this section avoids all unwanted collisions of computational paths is to select K such that all its elements are relatively prime (and greater than 1), and then to make sure that no element N_i of B is greater than the corresponding element k_i of K. It is easy to see that Lemma 7 ensures that the path length difference of Equation 4.2 will never equal zero when $m \neq n$ under these conditions.

K and B are set to values that satisfy the constraints described in the previous paragraph in our state–efficient constructions that will be presented in the following subsections.

4.3.1. N–way Branching in a Constant Number of Passes

Given a positive error bound ϵ and an integer $c > 1$, select K as the one with the smallest value of k_c among the infinitely many ascending lists of c integers that satisfy the following constraints:

1. All the k_i are relatively prime and greater than 1.
2. $\prod_{i=1}^{c} k_i \geq \frac{1}{\epsilon}$.

B is set to equal K.

Since the 2qfa's built according to this prescription are little more than cascades of c machines of the type described in Section 4.2, it is easy to see that they recognize L_{eq} with error less than or equal to ϵ, and their runtime is $O(|w|)$, since c does not depend on ϵ.

For a simple analysis of the state complexity, we consider slightly different machines whose state set sizes can be used as pessimistic estimates of the ones described above: Let us say that we choose K (and B) as the list of the first c primes greater than or equal to $(\frac{1}{\epsilon})^{(\frac{1}{c})}$. The resulting machine cannot possibly have fewer states than our construct for the same (ϵ, c) pair. Recalling that the state complexity of the basic machine $M_{N,N}$ from Section 4.2 is $O(N^2)$, we see that the size of the multi–pass machine is bounded by the sum of the squares of the c primes mentioned above, which is easily found to be $O((\frac{1}{\epsilon})^{\frac{2}{c}})$.

4.3.2. Two–way Branching in Multiple Passes with Collision Avoidance Using Primes

Formula 1. *The sum of the first k primes is*

$$\sum_{i=1}^{k} p_i \sim \frac{1}{2} k^2 \ln(k). \tag{4.3}$$

For a given ϵ, select K as the list of the first r_1 primes, where r_1 is the smallest integer such that $2^{r_1} \geq \frac{1}{\epsilon}$. Set all r_1 elements of B to 2.

Since $\lceil \log(\frac{1}{\epsilon}) \rceil$ passes are performed in each of Stages II and III, this method yields machines with runtime $O(\log(\frac{1}{\epsilon})|w|)$. The state complexity, which is mainly due to the modulo–k_i counting in this case, can be calculated asymptotically by using the well–known formula for the sum of the first r_1 primes (4.3), and turns out to be $O(\log^2(\frac{1}{\epsilon}) \log(\log(\frac{1}{\epsilon})))$.

4.3.3. N–way Branching in Multiple Passes with Collision Avoidance Using Primes

Formula 2. *The product of primes less than or equal to the k^{th} prime (primorial function)*

$$p_k\# \sim e^{(1+o(1))k \log(k)}. \tag{4.4}$$

Formula 3. *The sum of the squares of the first k primes is*

$$\sum_{i=1}^{k} p_i^2 \sim \frac{1}{3} k^3 \log^2(k). \tag{4.5}$$

Formula 4. *The solution of the equation $x \log(x) = a$ is*

$$x = \frac{a}{\log(a)} e^{O(\frac{\log(log(a))}{\log(a)})} \tag{4.6}$$

or

$$x = O(\frac{a}{\log(a)}). \tag{4.7}$$

For a given ϵ, select K (and B) as the list of the first r_2 primes, where r_2 is the

smallest integer such that the product of all the elements of K is greater than or equal to $\frac{1}{\epsilon}$.

Using the well–known formula for the asymptotic growth of the primorial function (4.4), one can calculate r_2 as follows:

$$\begin{aligned} p_{r_2}\# \sim e^{(1+o(1))r_2 \log(r_2)} &\geq \tfrac{1}{\epsilon} \\ (1+o(1))r_2 \log(r_2) &\geq \ln(\tfrac{1}{\epsilon}) \\ r_2 \log(r_2) &\geq \tfrac{\ln(\frac{1}{\epsilon})}{(1+o(1))} \end{aligned}$$

since we want to find the smallest integer value of r_2 (by using 4.7)

$$r_2 = O(\frac{\log(\frac{1}{\epsilon})}{\log(\log(\frac{1}{\epsilon}))}). \tag{4.8}$$

Since r_2 passes are performed in each of Stages II and III, this method yields machines with runtime $O(\frac{\log(\frac{1}{\epsilon})}{\log(\log(\frac{1}{\epsilon}))}|w|)$, which is faster than the construction described in Subsection 4.3.2. The state complexity is proportional to the sum of the squares of the first r_2 primes (4.5), and turns out to be $O(\frac{\log^3(\frac{1}{\epsilon})}{\log^3(\log(\frac{1}{\epsilon}))} \log^2(\frac{\log(\frac{1}{\epsilon})}{\log(\log(\frac{1}{\epsilon}))}))$, exceeding that of the machines of Subsection 4.3.2.

5. CONCLUSIONS

In this thesis, we have presented several methods for constructing efficient 2qfa's to recognize the non–regular language $L_{eq} = \{a^n b^n | \ n > 0\}$. Table 5.1 contains a quick comparison of our methods with previous work. One of our techniques produces machines which halt in $O(|w|)$ time (i.e. the running time does not depend on the error bound) and which have $O((\frac{1}{\epsilon})^{\frac{2}{c}})$ states for any given constant $c > 1$. Other methods, yielding machines whose state complexities are polylogarithmic in $\frac{1}{\epsilon}$, and which halt in $O((\log \frac{1}{\epsilon})|w|)$ time, are also presented, demonstrating a trade–off between time and state complexities in this highly space–efficient framework.

Table 5.1. Comparisons of methods w.r.t. runtime and size of state set

method	**runtime**	**size of state set**
N–way Branching with a Single Pass	$O(\frac{1}{\epsilon}\|w\|)$	$O((\frac{1}{\epsilon})^2)$
Two–way Branching in Multiple Passes	$O((\frac{1}{\epsilon})^2\|w\|)$	$O((\frac{1}{\epsilon})^2)$
Two–way B.M.P[a] with Collision Avoidance	$O(\frac{1}{\epsilon}\|w\|)$	$O(\frac{1}{\epsilon})$
Quotient Comparison	$O(\|w\|)$	$O((\frac{1}{\epsilon})^2)$
N–way Branching in a Constant Number of Passes	$O(\|w\|)$	$O((\frac{1}{\epsilon})^{\frac{2}{c}})$
Two–way BMP with Collision Avoidance UP[b]	$O(\log(\frac{1}{\epsilon})\|w\|)$	$O(\log^2(\frac{1}{\epsilon})\log(\log(\frac{1}{\epsilon})))$
N–way BMP with Collision Avoidance UP	$O(\frac{\log(\frac{1}{\epsilon})}{\log(\log(\frac{1}{\epsilon}))}\|w\|)$	$O(\frac{\log^3(\frac{1}{\epsilon})}{\log^3(\log(\frac{1}{\epsilon}))}\log^2(\frac{\log(\frac{1}{\epsilon})}{\log(\log(\frac{1}{\epsilon}))}))$

[a] Branching in Multiple Passes
[b] Using Primes

REFERENCES

1. Moore, C. and J. P. Crutchfield, "Quantum Automata and Quantum Grammars", 1997, http://www.citebase.org/abstract?id=oai:arXiv.org:quant-ph/9707031.

2. Watrous, J., "On the Power of 2-Way Quantum Finite State Automata", Technical Report CS-TR-1997-1350, University of Wisconsin, 1997, http://citeseer.ist.psu.edu/article/watrous97power.html.

3. Kondacs, A. and J. Watrous, "On the Power of Quantum Finite State Automata", *IEEE Symposium on Foundations of Computer Science*, pp. 66–75, 1997, http://citeseer.ist.psu.edu/kondacs97power.html.

4. Ambainis, A. and R. Freivalds, "1-way quantum finite automata: strengths, weaknesses and generalizations", *The Computing Research Repository*, Vol. quant-ph/9802062, 1998.

5. Gruska, J., *Quantum Computing*, McGraw-Hill, 1999, http://www.fi.muni.cz/usr/gruska/quantum/.

6. Ambainis, A., A. Ķikusts, and M. Valdats, "On the Class of Languages Recognizable by 1-Way Quantum Finite Automata.", *Symposium on Theoretical Aspects of Computer Science*, pp. 75–86, 2001.

7. Gruska, J., "Descriptional Complexity Issues in Quantum Computing", *Journal of Automata, Languages and Combinatorics*, Vol. 5, No. 3, pp. 191–218, 2000.

8. Bertoni, A. and M. Carpentieri, "Regular Languages Accepted by Quantum Automata", *Information and Computation*, Vol. 165, No. 2, pp. 174–182, 2001.

9. Brodsky, A. and N. Pippenger, "Characterizations of 1-Way Quantum Finite Au-

tomata", *SIAM Journal on Computing*, Vol. 31, No. 5, pp. 1456–1478, 2002.

10. Ambainis, A. and J. Watrous, "Two-way finite automata with quantum and classical states", *Theoretical Computer Science*, Vol. 287, No. 1, pp. 299–311, 2002.

11. Qiu, D. and M. Ying, "Characterizations of Quantum Automata", *Theoretical Computer Science*, Vol. 312, No. 2-3, pp. 479–489, 2004.

12. Shepherdson, J. C., "The reduction of two-way automata to one-way automata", *IBM Journal of Research and Development*, Vol. 3, pp. 198–200, 1959.

13. Freivalds, R., "Probabilistic Two-Way Machines", *Proceedings on Mathematical Foundations of Computer Science*, pp. 33–45, Springer-Verlag, London, UK, 1981.

14. Dwork, C. and L. Stockmeyer, "Finite state verifiers I: the power of interaction", *Journal of the ACM*, Vol. 39, No. 4, pp. 800–828, 1992.

15. Rabin, M. O. and D. Scott, "Finite Automata and Their Decision Problems", Moore, E. F. (editor), *Sequential Machines*, chapter 2, pp. 63–91, Addison-Wesley Publishing Company, Inc., 1964.

16. Sipser, M., *Introduction to the Theory of Computation*, Thomson Course Technology, United States of America, 2 edition, 2006.

17. Nerode, A., "Linear Automaton Transformations", *Proceedings of the American Mathematical Society*, Vol. 9, pp. 264–268, 1958.

18. Rabin, M. O., "Probabilistic Automata", Moore, E. F. (editor), *Sequential Machines*, chapter 4, pp. 98–114, Addison-Wesley Publishing Company, Inc., 1964.

19. Rasščevskis, Z., "The Complexity of Probabilistic versus Deterministic Finite Automata", 2000, `http://www.mdh.se/ima/personal/rbr01/courses/sundbyholmproc/zigmars.pdf`.

20. Bonner, R., R. Freivalds, and Z. Rasščevskis, "Size of Finite Probabilistic and Quantum Automata", 2002, http://citeseer.ist.psu.edu/bonner02size.html.

21. Kuklin, Y. I., "Two-way Probabilistic Automata", *Avtomatika i vyčistitelnaja tekhnika*, Vol. 5, pp. 36–36, 1973, (Russian).

22. Dwork, C. and L. Stockmeyer, "The Time Complexity Gap for 2-Way Probabilistic Finite-State Automata", *SIAM Journal on Computing*, Vol. 19, No. 6, pp. 1011–1023, 1990, http://citeseer.ist.psu.edu/424259.html.

23. Condon, A., "Bounded Error Probabilistic Finite State Automata", Panos Pardalos, J. R. and J. Rolim (editors), *Handbook on Randomized Computing, Volume II*, chapter 1, pp. 509–532, Kluwer, 2001.

24. Kaŋeps, J. and R. Freivalds, "Minimal nontrivial space complexity of probabilistic one-way turing machines", *MFCS'90: Proceedings on Mathematical foundations of computer science 1990*, pp. 355–361, Springer-Verlag New York, Inc., New York, NY, USA, 1990.

25. Pin, J.-E., "On Reversible Automata", *LATIN '92: Proceedings of the 1st Latin American Symposium on Theoretical Informatics*, pp. 401–416, Springer-Verlag, London, UK, 1992.

26. Lange, K.-J., P. McKenzie, and A. Tapp, "Reversible Space Equals Deterministic Space", *CCC'97: Proceedings of the 12th Annual IEEE Conference on Computational Complexity*, p. 45, IEEE Computer Society, Washington, DC, USA, 1997.

27. Feynman, R., "Simulating Physics with Computers", *International Journal of Theoretical Physics*, Vol. 21, No. 6&7, pp. 467–488, 1982, http://citeseer.ist.psu.edu/feynman82simulating.html.

28. Deutsch, D., "Quantum theory, the Church-Turing principle and the universal

quantum computer", *Proceedings of the Royal Society of London Series A*, Vol. A400, pp. 97–117, 1985, http://citeseer.ist.psu.edu/deutsch85quantum.html.

29. Deutsch, D. and R. Jozsa, "Rapid Solution of Problems by Quantum Computation", Technical report, Bristol, UK, 1992.

30. Bernstein, E. and U. Vazirani, "Quantum complexity theory", *Proceedings of the 25th ACM Symposium on the Theory of Computation*, pp. 11–20, ACM Press, New York, 1993, http://citeseer.ist.psu.edu/article/bernstein97quantum.html.

31. Simon, D. R., "On the Power of Quantum Computation", *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, pp. 116–123, Institute of Electrical and Electronic Engineers Computer Society Press, Los Alamitos, CA, 1994, http://citeseer.ist.psu.edu/simon94power.html.

32. Shor, P. W., "Algorithms for Quantum Computation: Discrete Logarithms and Factoring", *IEEE Symposium on Foundations of Computer Science*, pp. 124–134, 1994, http://citeseer.ist.psu.edu/14533.html.

33. Grover, L. K., "A fast quantum mechanical algorithm for database search", 1996, http://www.citebase.org/abstract?id=oai:arXiv.org:quant-ph/9605043.

34. Feller, W., *An Introduction to Probability Theory and Its Applications*, Vol. 1, Wiley, January 1968.

35. Amano, M. and K. Iwama, "Undecidability on quantum finite automata", *STOC '99: Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pp. 368–375, ACM Press, New York, NY, USA, 1999.

36. Gruska, J., "Algebraic Methods in Quantum Informatics", 2007, http://web.auth.gr/cai07/Lectures/GruskaCAI07.pdf.

37. Masaki Nakanishi, K. H., Takao Indoh and T. Kashiwabara, "On the Power of Quantum Pushdown Automata with a Classical Stack and 1.5-way Quantum Finite Automata", Computing Science Technical Report no. 2001005, Nara Institute of Science and Technology, March 2001, `http://isw3.naist.jp/IS/TechReport/report/2001005.ps`.

38. Atak, F. M., *Design and Simulation of Two-Way Quantum Finite Automata*, Master's thesis, Boğaziçi University, İstanbul, Turkey, 2006.