CLASSICAL AND QUANTUM COMPUTATION WITH SMALL SPACE BOUNDS

by

Abuzer Yakaryılmaz

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

M.S., Computer Engineering, Boğaziçi University, 2007

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

Graduate Program in Computer Engineering

Boğaziçi University

2011

CLASSICAL AND QUANTUM COMPUTATION WITH SMALL SPACE BOUNDS

APPROVED BY:

Prof. Ahmet Celal Cem Say (Thesis Supervisor)	.....
Assist. Prof. Ali Taylan Cemgil	.....
Assoc. Prof. Can Özturan	.....
Assoc. Prof. Ferit Öztürk	.....
Assist. Prof. Hüsnü Yenigün	.....

DATE OF APPROVAL: 25.11.2010

To My Sister Fatma...

ACKNOWLEDGEMENTS

First of all, I would like to thank my PhD advisor A. C. Cem Say. It has been a great experience for me to work with Cem Say and it is an honour for me to be his first PhD graduate. With his guidance, I have learned many things not only about "theory" but also about how to be a researcher. His research excitement has always motivated me and they were very special moments to see his eyes' shining whenever we came to a new finding.

I also would like to express my gratitude to Ali Taylan Cemgil, Kıvanç Mıhçak, Can Özturan, Ferit Öztürk, and Hüsnü Yenigün, my thesis committee members, who devoted their time and energy to this research.

I would like to thank Dilek Çankaya for her great support at first period of this research. During that time, I had completed the main framework of the thesis.

I would like to thank Rūsiņš Freivalds for his collaborative works and his efforts to our joint works. I would like to thank Andris Ambainis for offering to write a chapter on "quantum automata" together, for his great advice and for many helpful discussions and for kindly accepting my short visit to Riga. I would like to thank John Watrous for his great advice and for many helpful discussions and for kindly accepting my short visit to Waterloo. I would like to thank Farid Ablayev, Ali Taylan Cemgil, Flavio D'Alessandro, Juraj Hromkovič, Maris Ozols, Ferit Öztürk, Pascal Tesson, Mikahil Volkov for their helpful answers to my questions.

I would like to thank the organizing committee of SATA 2008 (Lisbon), TQC 2009 (Waterloo), AutoMathA 2009 (Liège), CSR 2009 (Novosibirsk), Randomized and Quantum Computation 2010 (Brno), QIP 2010 (Zurich), and QIP 2011 (Singapore).

I am grateful to my colleagues from the department of Computer Engineering at Boğaziçi University, with whom we have not only shared many academic experiences

but also participated in many social activities. I would like especially to thank all participants of our regularly scheduled basketball games.

This research has been partially supported by Boğaziçi University Scientific Research Projects with grants 07A103 and 08A102 and the Scientific and Technological Research Council of Turkey (TÜBİTAK) with grant 108E142.

The most beautiful finding during this research is Gönül. The proofs are written in my heart.

I would like to thank my family for their continuous encouragement and motivation not only with this work but also with much else.

ABSTRACT

CLASSICAL AND QUANTUM COMPUTATION WITH SMALL SPACE BOUNDS

In this thesis, we introduce a new quantum Turing machine model that supports general quantum operators, together with its pushdown, counter, and finite automaton variants, and examine the computational power of classical and quantum machines using small space bounds in many different cases. The main contributions are summarized below.

Firstly, we consider quantum Turing machines in the unbounded error setting: (i) in some cases of sublogarithmic space bounds, the class of languages recognized by quantum Turing machines is shown to be strictly larger than that of classical ones; (ii) in constant space bounds, the same result can still be obtained for restricted quantum Turing machines; (iii) the complete characterization of the class of languages recognized by realtime constant space nondeterministic quantum Turing machines is given.

Secondly, we consider constant space-bounded quantum Turing machines in the bounded error setting: (i) we introduce a new type of quantum and probabilistic finite automata with a special two-way input head which is not allowed to be stationary or move to the left but has the capability to reset itself to its starting position; (ii) the computational power of this type of quantum machine is shown to be superior to that of the probabilistic machine; (iii) based on these models, two-way probabilistic and two-way classical-head quantum finite automata are shown to be more succinct than two-way nondeterministic finite automata and their one-way variants; (iv) we also introduce probabilistic and quantum finite automata with postselection with their

bounded error language classes, and give many characterizations of them.

Thirdly, the computational power of realtime quantum finite automata augmented with a write-only memory is investigated by showing many simulation results for different kinds of counter automata. Parallelly, some results on counter and pushdown automata are obtained.

Finally, some lower bounds of realtime classical Turing machines in order to recognize a nonregular language are shown to be tight. Moreover, the same question is investigated for some other kinds of realtime machines and several nonregular languages recognized by them in small space bounds are presented.

ÖZET

AZ BELLEĞE SAHİP KLASİK VE KUANTUM HESAPLAMA

Bu tezde genel kuantum operatörlerini destekleyen yeni bir kuantum Turing makine modeli ile birlikte onun yığıt-bellekli, sayaçlı ve sonlu bellekli modelleri tanımlandı ve az belleğe sahip klasik ve kuantum makinelerin hesaplama güçleri bir çok durum için incelendi. Temel katkılarımız aşağıda özetlenmiştir.

İlk olarak kuantum Turing makineleri sınırlı olmayan hata açısından ele alındı: (i) logaritma-altı bellek kullanılan bazı durumlarda hesaplama gücü açısından kuantum Turing makinelerinin klasik muadillerinden üstün olduğu gösterildi; (ii) aynı sonuç sonlu belleğe sahip kısıtlı kuantum Turing makineleri için de elde edildi; (iii) gerçek-zamanlı sonlu belleğe sahip belirlenimci olmayan kuantum Turing makinelerinin tanıdığı dil ailesi belirlendi.

İkinci olarak, sonlu belleğe sahip kuantum Turing makineleri sınırlı hata açısından ele alındı: (i) sola gitme veya durma hakkı yasaklanmış fakat kendisini başlangıç noktasına taşıyabilen özel kafaya sahip yeni çift-yönlü kuantum ve olasılıksal sonlu durumlu makineler tanımlandı; (ii) hesaplama gücü açısından bu tür kuantum makinelerin olasılıksal olanlardan daha güçlü olduğu gösterildi; (iii) bu modeller temelinde, çift-yönlü olasıksal ve klasik kafaya sahip kuantum sonlu durum makinelerin, çift-yönlü belirlenimci olmayan sonlu durum makineler ile kendi tek-yönlü muadillerinden daha az sonlu bellek kullandıkları gösterildi; (iv) ayrıca olasılıksal ve kuantum sonseçimli sonlu durumlu makineler ile sınırlı hata payı ile tanıdıkları dil sınıfları tanımlandı ve bir çok özellikleri gösterildi.

Üçüncü olarak, sadece yazma hakkı olan bir hafıza eklenen gerçek-zamanlı kuantum sonlu durumlu makenelerin hesaplama gücü, farklı türdeki sayaçlı makineler üzerinden yapılan bir çok benzetim ile incelendi. Paralel olarak, sayaçlı ve yığıt-bellekli makinelere dair bazı sonuçlar elde edildi.

Son olarak, literatürde geçen bazı alt sınırların, düzenli olmayan bir dili tanıyan gerçek-zamanlı klasik Turing makineler için mümkün olan en iyi sınırlar oldukları gösterildi. Ek olarak, benzer soru diğer tür gerçek-zamanlı makineler için araştırıldı ve onlar tarafından az bellek ile tanınan birçok düzenli olmayan dilin varlığı gösterildi.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS iv
ABSTRACT vi
ÖZET viii
LIST OF FIGURES xiii
LIST OF TABLES xv
1. INTRODUCTION 1
2. SPACE-BOUNDED COMPUTATION 5
2.1. Preliminaries 5
2.1.1. Basic Notation 5
2.1.2. Basic Terminology 6
2.1.3. Language Recognition 10
2.1.3.1. Cutpoint 10
2.1.3.2. One-Sided Cutpoint 10
2.1.3.3. Exclusive and Inclusive Cutpoint 11
2.1.3.4. Unbounded Error 11
2.1.3.5. One-Sided Unbounded Error 11
2.1.3.6. Isolated Cutpoint or Bounded Error 11
2.1.3.7. One-Sided Bounded Error 12
2.2. Turing Machines 13
2.3. Pushdown Automata 17
2.4. Counter Automata 19
2.5. Formal Definition of Classical Machines 22
2.6. Computation of Machines 23
2.6.1. Deterministic Computation 23
2.6.2. Nondeterministic Computation 23
2.6.3. Alternating Computation 24
2.6.4. Probabilistic Computation 24
2.6.5. Quantum Computation 25
2.7. Space Classes 27

3. A NEW KIND OF QUANTUM MACHINE 31
3.1. The General Framework for Quantum Machines 31
3.2. Quantum Turing Machines 34
3.2.1. Two-Way Quantum Finite Automata 34
3.2.2. Unidirectional Quantum Turing Machines 35
3.2.3. Quantum Turing Machines with Classical Heads 36
3.2.4. Realtime Quantum Finite Automata 39
3.2.5. Quantum Turing Machines with Restricted Measurements . . . 42
3.3. Quantum Pushdown Automata 45
3.4. Quantum Counter Automata 46
3.5. Nondeterministic Quantum Machines 48
4. SUBLOGARITHMIC-SPACE UNBOUNDED-ERROR COMPUTATION . . 49
4.1. Unbounded Error Results 49
4.1.1. Probabilistic versus Quantum Computation with Sublogarithmic Space 49
4.1.2. Languages Recognized by Kondacs-Watrous Quantum Finite Automata 58
4.2. Nondeterministic Quantum Computation 63
4.2.1. Basic Facts 63
4.2.2. Languages Recognized with One-Sided Error 64
4.2.3. Space Efficiency of Nondeterministic Quantum Finite Automata 68
4.2.4. Languages Recognized with Two-Sided Error 69
4.2.5. Closure Properties 73
5. CONSTANT-SPACE BOUNDED-ERROR COMPUTATION 76
5.1. Probabilistic and Quantum Automata with Resetting 76
5.1.1. Definitions 76
5.1.2. Basic Facts 78
5.1.3. Computational Powers of Realtime Probabilistic Finite Automata with Restart 81
5.1.4. Computational Powers of Realtime Quantum Finite Automata with Restart 83
5.2. Succinctness of Two-Way Models 90

5.3. Probability Amplification 95
5.3.1. Improved Algorithms for UPAL Language 96
5.3.2. A Realtime Quantum Finite Automata with Restart Algorithm for UPAL Language 98
5.3.3. An Improved Algorithm for PAL Language 99
5.4. Probabilistic and Quantum Automata with Postselection 101
5.4.1. Definitions 102
5.4.2. Characterization of Realtime Postselection Automata 106
5.4.3. Latvian Postselection Automata 109
6. WRITE-ONLY MEMORY 112
6.1. Definitions 112
6.2. Basic Facts 114
6.3. Realtime Quantum Finite Automata with Incremental-Only Counters . 117
6.3.1. Bounded Error 118
6.3.2. Unbounded Error 123
6.4. Realtime Quantum Finite Automata with Push-Only Stacks 128
6.5. Realtime Quantum Finite Automata with Write-Only Memories 130
7. SUBLINEAR-SPACE REALTIME TURING MACHINES 133
8. RELATED WORK 139
8.1. Counter and Pushdown Automata 139
8.2. Promise Problems 143
8.3. Multihead Finite Automata 144
9. CONCLUSION 145
APPENDIX A: LOCAL CONDITIONS FOR QUANTUM MACHINES 148
A.1. Quantum Turing Machines 148
A.2. Quantum Pushdown Automata 150
APPENDIX B: QUANTUM COMPUTATION 155
REFERENCES 162

LIST OF FIGURES

Figure 2.1. Generalized finite automaton 26

Figure 3.1. The matrix representation of superoperators (E) 33

Figure 3.2. The local conditions for unidirectional QTM wellformedness 35

Figure 3.3. The local conditions for unidirectional 2QFA wellformedness . . . 35

Figure 3.4. The definition and properties of *vec* 41

Figure 4.1. Specification of the transition function of the 1KWQFA presented in the proof of Theorem 4.3 (I) 51

Figure 4.2. Specification of the transition function of the 1KWQFA presented in the proof of Theorem 4.3 (II) 52

Figure 4.3. A new family of nonstochastic languages 57

Figure 4.4. General template to build a unitary matrix (I) 59

Figure 5.1. General template to build a unitary matrix (II) 87

Figure 5.2. Specification of the transition function of the RT-KWQFA presented in the proof of Theorem 5.23 98

Figure 5.3. Specification of the transition function of the RT-KWQFA presented in the proof of Theorem 5.24 100

Figure 6.1. Probabilistic zero-checking of multiple counters by one counter . . 116

Figure 6.2. The description of N-way quantum Fourier transform used by the machines having a WOM 120

Figure 6.3. The details of the transition function of the RT-QFA-IOC presented in the proof of Theorem 6.14 (I) 125

Figure 6.4. The details of the transition function of the RT-QFA-IOC presented in the proof of Theorem 6.14 (II) 126

Figure 6.5. The transitions of the RT-QFA-POS of Theorem 6.16 129

Figure 9.1. The relationships among classical and quantum constant space-bounded classes 146

Figure B.1. The matrix obtained by deleting all rows of $U^\dagger$ except the ones in $\mathcal{C}\times\omega_1$. 160

Figure B.2. Re-partitioning of the matrix in Figure B.1 161

LIST OF TABLES

Table 2.1. The list of the abbreviations of Turing machines 15

Table 2.2. The list of the abbreviations of finite automata 16

Table 2.3. The list of the abbreviations of pushdown automata 18

Table 2.4. The list of the abbreviations of counter automata 20

Table 7.1. Lower bounds of {D,N,A}TMs in order to recognize a nonregular language 133

1. INTRODUCTION

Computation with small space-bounds has always had a special emphasis in the field of theoretical computer science. In this thesis, we examine both classical and quantum small space complexity classes. Our main tools are Turing machines (TM) and some restricted variants of them, i.e. one-way or realtime TMs, finite automata, and counter and pushdown automata. Since quantum computation is relatively a young research field (and so it has been less investigated than classical computation), our main focus is on quantum machines and their space-bounded classes.

Moreover, we think that examining small space complexity classes or the models working in small space-bounds is useful for understanding and comparing different kinds of computational resources, such as *nondeterminism*, *randomization*, *quantumness*, etc. Therefore, we also try to investigate the cases in which quantumness (or randomization) has advantages and the cases having no advantage.

Our interest of "space" can be classified into four general cases, which can also be seen as the understanding of what we mean by small space-bounds:

i. constant space,
ii. sublogarithmic space,
iii. sublinear space, and
iv. linear space.

Note that, such a classification may not be meaningful for some models since their computational power increases only when using logarithmic or linear space [1, 2]. On the other hand, even constant space can be sub-classified for some models in terms of language recognition [3–5].

In the following part, based on the outline of the thesis, we give a short description of each chapter by highlighting the main results.

Chapter 2 begins with the preliminaries in which the basic notations, terminologies, and language recognition settings used throughout the thesis are presented. After that, the conventional TM model for space bounded computations and its classical variants, i.e. finite automata (FA), counter automata (CA), and pushdown automata (PDA), are given with their formal definitions and computational specifications. The chapter ends with the definitions of many space classes.

In Chapter 3, we introduce a new kind of quantum Turing machine (QTM) allowing to implement general quantum operators and its FA, PDA, and CA variants. Since the previously defined QTM models for space-bounded computations [6–8] did not reflect the full generality of quantum mechanics, our new QTM model is one of the contributions of the thesis. We also present several well-formedness conditions, i.e. a list of constraints obeyed by the transition function of the machine, for the quantum models[1] . Moreover, we present the standard technique for quantum machines in order to simulate their classical counterparts exactly, such that both machines agree on the accepting value for any given input string. Lastly, the "linearization" technique of a quantum finite automaton (QFA) operating in realtime[2] is given.

In Chapter 4, we focus on our results related with unbounded error, both in the general case and in one-sided (equivalently, nondeterministic) case. Firstly, we show that one-way[3] QFAs (1QFAs), that is, one-way QTMs (1QTMs) with constant space, can recognize some nonstochastic languages that were shown to be not recognizable by a probabilistic Turing machine (PTM) (resp., one-way PTMs (1PTMs)) in space $o(\log\log n)$ (resp., $o(\log n)$). This is one of our superiority results of quantum computation over classical computation in $o(\log\log n)$ and $o(\log n)$ spaces. Thus, we partially solve an open problem addressed in [6], i.e. the relationship between QTMs and PTMs for sublogarithmic space bounds. In the next part, we solve another open problem introduced in [9], i.e. the complete characterization of a restricted type realtime QFA (RT-QFA), namely realtime Kondacs-Watrous QFA [4] (RT-KWQFA),

[1]Some of them are given in Appendix A.

[2]The input head of the machine is allowed to move only to the right in each step.

[3]The input head of the machine is allowed either to be stationary or to move to the right in each step.

in the unbounded error setting. We show that the language class recognized by RT-KWQFAs is equal to that of realtime probabilistic FAs (RT-PFAs) in this setting, using a simulation technique. Moreover, this technique is also used in order to demonstrate some other results. Thirdly, we present the complete characterization of the class of languages recognized by realtime nondeterministic QFAs (RT-NQFAs) and then show some non-trivial properties of this class.

In Chapter 5, we shift our focus on the results related to constant space-bounds and the bounded-error cases. Firstly, we define a new type of QFAs and PFAs with a special two-way input head which is not allowed to be stationary or move to the left, but has the capability to reset the input head to its initial position and to change the internal state to a specified one. If the specified internal state is set to be the initial one, these machines are called realtime QFAs and PFAs with restart, denoted by RT-QFA$^{\circlearrowleft}$s and RT-PFA$^{\circlearrowleft}$s. Secondly, we show that the class of the languages recognized by this kind of QFAs is a proper superset of that of PFAs in bounded error setting. Moreover, based on these models, we show that two-way QFAs and two-way PFAs (2QFAs and 2PFAs) can be more concise than two-way nondeterministic FAs (2NFAs) and their one-way variants. Additionally, in a special setting, we also show that 2QFAs can be more concise than 2PFAs. Thirdly, we present more space- and time-efficient algorithms for 2QFAs in order to recognize some nonregular languages. In the last part of this chapter, we introduce the FAs endowed with an unrealistic theoretical capability, i.e. *postselection*. By postselection, it is meant that the final decision on the input is given by a specified subset of the computation outcomes and so the rest of the computation outcomes are discarded [10]. We introduce both RT-QFAs and RT-PFAs with postselection (RT-PostQFA and RT-PostPFA) and then give some characterizations of the classes of the languages recognized by them. As an interesting observation, the class of languages recognized by RT-PostQFA and RT-QFA$^{\circlearrowleft}$ (resp., RT-PostPFA and RT-PFA$^{\circlearrowleft}$) are the same in both bounded and unbounded error settings. Additionally, in [11, 12], a somewhat different QFA model with postselection was defined based on RT-KWQFA. We also examine probabilistic and quantum versions of these models, and present some characterizations of the classes of languages recognized by them.

In Chapter 6, we present our results that are extensions of the ones we gave in [13] and [14], that is, a RT-QFA augmented with a *write-only memory* (WOM) (RT-QFA-WOM). Contrary to classical computation, due to the interference of the configurations, the computational power of a quantum machine can be increased by using a WOM. Throughout the chapter, we investigate the computational power of RT-QFA with IOC (*increment-only counter*), POS (*push-only stack*), and WOM by showing some simulations of classical machines, such as blind and reversal counter automata, and giving some example languages.

The minimum space usage required by a TM in order to recognize a nonregular language has been determined for many cases [2], but realtime TMs have not been considered. In Chapter 7, we show that RT-TMs can share the same lower bounds with one-way TMs (1TMs) by using a general simulation technique. Moreover, the same question is investigated for some other kinds of realtime machines and several nonregular languages recognized by them in small space bounds are presented.

In Chapter 8, we present the remaining results of our work that we chose not to organize as a separate chapter.

Section 9 is the conclusion of the thesis. We present a technical summary of all results. A short review of the mathematical formalism used for quantum computation in this thesis is given in Appendix B.

2. SPACE-BOUNDED COMPUTATION

This chapter forms the foundation of the thesis. In Section 2.1, we list basic notations and terminologies and language recognition settings used throughout the thesis. The details of generic TMs convenient for space bounded computations are presented in Section 2.2. Consequently, the details of generic PDAs and CAs are given in Sections 2.3 and 2.4, respectively. Then, the formal definitions of classical (deterministic, nondeterministic, alternating, and probabilistic) machines are presented in Section 2.5. The computational specifications of all types of machines that we cover (classical and quantum) are described in Section 2.6. In the last section (2.7), the generic space complexity classes and some specific space classes introduced in this thesis are presented.

2.1. Preliminaries

2.1.1. Basic Notation

i. For a given set S, $|S|$ is the size of the set.

ii. For a given alphabet A, A^* is the set of *the empty string* (or *the empty symbol*), denoted as ε, and all strings obtained by finitely concatenating the symbols of A.

iii. For a given string w, $|w|$ is the length of w, w_i is the i^{th} symbol of w.

iv. For a given vector (row or column) v, $v[i]$ is the i^{th} entry of v.

v. For a given matrix A, $A[i,j]$ is the $(i,j)^{th}$ entry of A.

vi. Let $\{A_i \mid 1 \leq i \leq n\}$ be a set. Then, $\bar{A}$ denotes $A_1 \times \cdots \times A_n$ and any instance of $\bar{A}$ is denoted by small letter, i.e. $\bar{a}$ or $\bar{a}_{i \in \{1,\ldots,|\bar{A}|\}}$.

vii. Some fundamental conventions in Hilbert space are as follows:

- v and its conjugate transpose are denoted as $|v\rangle$ and $\langle v|$, respectively;
- the multiplication of $\langle v_1|$ and $|v_2\rangle$ is shortly written as $\langle v_1|v_2\rangle$;
- the tensor product of $|v_1\rangle$ and $|v_2\rangle$ can also be written as $|v_1\rangle|v_2\rangle$ instead of $|v_1\rangle \otimes |v_2\rangle$,

where v, v_1, and v_2 are vectors.

2.1.2. Basic Terminology

Σ (*input alphabet*): Σ is a finite set of symbols, i.e. $\Sigma = \{\sigma_1, \ldots, \sigma_{|\Sigma|}\}$. As a convention, Σ never contains the symbol # (*the blank symbol*), ¢ (*the left end-marker of the input*), and \$ (*the right end-marker of the input*). $\tilde{\Sigma}$ denotes the set $\Sigma \cup \{¢, \$\}$. Additionally, $\tilde{w}$ denotes the string $¢w\$$, for any given input string $w \in \Sigma^*$.

Γ (*work tape, stack, or memory alphabet*): Γ is a finite set of symbols, i.e. $\Gamma = \{\gamma_1, \ldots, \gamma_{|\Gamma|}\}$. In case of work tape, Γ contains # and in case of memory, Γ additionally contains ε. On the other hand, in case of stack, Γ contains $\vdash$ (*the left end-marker of stack*) but not # and $\tilde{\Gamma} = \Gamma \cup \{\#\}$.

Δ (*the set of outcomes*) and Ω (*the finite register alphabet*): In quantum computation, in order to implement general quantum operators, the main system is subjected to interact with a writeable finite register (See Appendix B for details). Ω represents the alphabet of this finite register, i.e. $\Omega = \{\omega_1, \ldots, \omega_{|\Omega|}\}$. In some cases, a selective (projective) measurement is done on the finite register. In those contexts, Δ represents the finite set of outcome results of the measurement, i.e. $\Delta = \{\tau_1, \ldots, \tau_{|\Delta|}\}$; Ω is partitioned into $|\Delta|$ pairwise disjoint parts, i.e.

$$\Omega = \bigcup_{\tau \in \Delta} \Omega_\tau; \tag{2.1}$$

and the projective measurement P is specified as

$$P = \{P_{\tau \in \Delta} \mid P_\tau = \sum_{\omega \in \Omega_\tau} |\omega\rangle\langle\omega|\}. \tag{2.2}$$

In its *standard usage*, Δ contains three elements, i.e. $\Delta = \{n, a, r\}$, and the following actions are associated with the measurement outcomes:

- "n": the computation continuous;
- "a": the computation halts, and the input is accepted;

- "r": the computation halts, and the input is rejected.

Q (*the set of internal states*): Q is a finite set of internal states, i.e. $Q = \{q_1, \ldots, q_{|Q|}\}$. Unless otherwise specified, we assume the following:

- q_1 is *the initial state*;
- for classical computational models except realtime finite automata (see Section 2.5), Q is partitioned into three disjoint subsets, i.e. Q_a (the set of *the accepting states*), Q_r (the set of *the rejection states*), and $Q_n = Q \setminus \{Q_a \cup Q_r\}$ (the set of *the nonhalting states*);
- for realtime finite automata including quantum ones, only one subset of Q is defined, Q_a.

$<>$ (*the set of head directions*): $<>$ is set $\{\leftarrow, \downarrow, \rightarrow\}$, where "$\leftarrow$" means that the (corresponding) head moves one square left, "$\downarrow$" means that the head stays on the same square, and "$\rightarrow$" means that the head moves one square right. When the tape is used as a stack, the arrows are also associated with the following stack operations: "$\leftarrow$" means that the top symbol is popped from the stack, "$\downarrow$" means that the top symbol is popped from the stack and a new symbol is pushed into the stack, and "$\rightarrow$" means that a symbol is pushed into the stack. As a special case, $\rhd$ is set $\{\downarrow, \rightarrow\}$. Additionally, D_i, D_w, and D_s are some functions from Q to $<>$ (or to $\rhd$). As will be seen later, the subscripts "i", "w", and "s" correspond to input tape, work tape, and stack, respectively.

$\Diamond$ (*the counter operations*): $\Diamond$ is set $\{-1, 0, +1\}$, where "-1" means that the value of the counter is decreased by 1, "0" means that the value of the counter is not changed, and "$+1$" means that the value of the counter is increased by 1. As a special case, $\triangle$ is set $\{0, +1\}$. Additionally, D_c is a function from Q to $\Diamond^k$ (or to $\triangle^k$), where k, a nonnegative integer, denotes the number of the counter(s).

Θ (*the status of a counter*): Θ is set $\{0, 1\}$, where 1 means that the counter value is nonzero and 0 means that the counter value is zero.

δ (*the transition function*): The behavior of a machine is specified by its transition function. The domain and the range of a transition function may vary with respect to the capabilities of the model. For a general and formal definition, δ, which is called "having m domain components and n range components" throughout the thesis, is a function

$$\delta : X_1 \times X_2 \times \cdots \times X_m \to Z^{Y_1 \times Y_2 \times \cdots \times Y_n}, \tag{2.3}$$

where m and n depend on the model; $X_{1 \leq i \leq m}$'s (*the domain components*), which decide the updates in a single step, can be

- the current internal state,
- the symbols read by the tape heads, or
- the status of a counter;

$Y_{1 \leq j \leq n}$'s (*the range components*), which are updates done in a single step, can be

- the next internal state,
- the symbols written on the tapes, memories, etc.,
- the movements of the tape heads, or
- an update operation on a stack or a counter;

and Z (*the set of the transition values*) is a set of either discrete or continuous numbers. More specifically, for each $\bar{x} \in \bar{X}$,

$$\delta(\bar{x}) = \sum_{\bar{y}_i \in \bar{Y}} z_i \bar{y}_i, \tag{2.4}$$

and $z_i \in Z$ is called *the transition value* and by overloading notation δ, z_i is also represented as

$$z_i = \delta(\bar{x}; \bar{y}_i) = \delta(\bar{x}[1], \ldots, \bar{x}[m], \bar{y}_i[1], \ldots, \bar{y}_i[n]), \tag{2.5}$$

where

$$\delta : X_1 \times \cdots \times X_m \times Y_1 \times \cdots \times Y_n \to Z. \tag{2.6}$$

Note that, independent of the context, $z_i = 0$ always corresponds to the case where there is no update defined from $\bar{x}$ to $\bar{y}_i$. For *nondeterministic and alternating* machines, Z is set to $\{0, 1\}$, where the transitions are defined only for values 1. If there is exactly one transition defined for each domain elements, we obtain a *deterministic* machine. The transition function of a deterministic machine can be written in the simplified form as follows:

$$\delta : \bar{X} \to \bar{Y}. \tag{2.7}$$

For *probabilistic* machines, Z is set to $[0, 1]$ (or a subset of a $[0, 1]$), where the transitions are defined for nonzero values and each transition value is called as *the transition probability*. For each domain value, all related transition probabilities must always be summed up to 1, i.e. for each $\bar{x} \in \bar{X}$,

$$\sum_{\bar{y}_i \in \bar{Y}} \delta(\bar{x}; \bar{y}_i) = 1. \tag{2.8}$$

This condition is known as *the local condition for PTM wellformedness*. For *quantum* machines, Z is set to a subset of complex numbers whose euclidean norm is at most 1. (The remaining details are given in Section 3.)

$f^a_{\mathcal{M}}(w)$ (*the accepting probability*): For a given machine $\mathcal{M}$ and an input string $w \in \Sigma^*$, $f^a_{\mathcal{M}}(w)$, or shortly $f_{\mathcal{M}}(w)$, is the accepting probability of w associated by $\mathcal{M}$. If the range of $f_{\mathcal{M}}(w)$ is extended to real number domain, it is called *the accepting value*. Moreover, $f^r_{\mathcal{M}}(w)$ is used in order to represent *the rejecting probability* of w associated by $\mathcal{M}$. Note that, for deterministic, nondeterministic, and alternating machines, the value of these functions is either 0 or 1.

2.1.3. Language Recognition

2.1.3.1. Cutpoint. The language $L \subseteq \Sigma^*$ recognized by machine $\mathcal{M}$ with (strict) cutpoint $\lambda \in \mathbb{R}$ is defined [15, 16] as

$$L = \{w \in \Sigma^* \mid f_{\mathcal{M}}(w) > \lambda\}. \tag{2.9}$$

The language $L \subseteq \Sigma^*$ recognized by machine $\mathcal{M}$ with nonstrict cutpoint $\lambda \in \mathbb{R}$ is defined [16] as

$$L = \{w \in \Sigma^* \mid f_{\mathcal{M}}(w) \geq \lambda\}. \tag{2.10}$$

The language $L \subseteq \Sigma^*$ is recognized by machine $\mathcal{M}$ with strict (resp., nonstrict) cutpoint if there exists a cutpoint $\lambda \in \mathbb{R}$ such that L is recognized by machine $\mathcal{M}$ with strict (resp., nonstrict) cutpoint λ.

2.1.3.2. One-Sided Cutpoint. The language $L \subseteq \Sigma^*$ is recognized by machine $\mathcal{M}$ with (positive) one-sided cutpoint $\lambda \in \mathbb{R}$ if L is recognized with cutpoint λ and $f_{\mathcal{M}}(w) = \lambda$ for all $w \notin L$.

The language $L \subseteq \Sigma^*$ is recognized by machine $\mathcal{M}$ with negative one-sided cutpoint $\lambda \in \mathbb{R}$ if $\overline{L}$ is recognized by machine $\mathcal{M}$ with positive one-sided cutpoint λ

The language $L \subseteq \Sigma^*$ is recognized by machine $\mathcal{M}$ with positive (resp., negative) one-sided cutpoint if there exists a cutpoint $\lambda \in \mathbb{R}$ such that L (resp., $\overline{L}$) is recognized by machine $\mathcal{M}$ with positive one-sided cutpoint $\lambda \in \mathbb{R}$

2.1.3.3. Exclusive and Inclusive Cutpoint. The language $L \subseteq \Sigma^*$ is recognized by machine $\mathcal{M}$ with exclusive cutpoint $\lambda \in \mathbb{R}$ if L is defined as

$$L = \{w \in \Sigma^* \mid f_{\mathcal{M}}(w) \neq \lambda\}. \tag{2.11}$$

The language $L \subseteq \Sigma^*$ is recognized by machine $\mathcal{M}$ with inclusive cutpoint $\lambda \in \mathbb{R}$ if $\overline{L}$ is recognized by machine $\mathcal{M}$ with exclusive cutpoint λ.

2.1.3.4. Unbounded Error. The language $L \subseteq \Sigma^*$ is recognized by machine $\mathcal{M}$ with unbounded error if L is recognized by $\mathcal{M}$ with either strict or nonstrict cutpoint [17].

2.1.3.5. One-Sided Unbounded Error. Let $f_{\mathcal{M}}(w) \in [0, 1]$.

The language $L \subseteq \Sigma^*$ recognized by machine $\mathcal{M}$ with *(positive) one-sided unbounded error* is defined as

$$L = \{w \in \Sigma^* \mid f_{\mathcal{M}}(w) > 0\}. \tag{2.12}$$

The language $L \subseteq \Sigma^*$ recognized by machine $\mathcal{M}$ with *negative one-sided unbounded error* is defined as

$$L = \{w \in \Sigma^* \mid f_{\mathcal{M}}(w) = 1\}. \tag{2.13}$$

2.1.3.6. Isolated Cutpoint or Bounded Error. Let $f_{\mathcal{M}}(w) \in [0, 1]$.

The language $L \subseteq \Sigma^*$ is said to be recognized by machine $\mathcal{M}$ with isolated cutpoint [15] $\lambda \in (0, 1)$ if there exists a $\delta > 0$ such that

- $f_{\mathcal{M}}(w) \geq \lambda + \delta$ when $w \in L$ and
- $f_{\mathcal{M}}(w) \leq \lambda - \delta$ when $w \notin L$,

where δ is known as the isolation gap.

The language $L \subseteq \Sigma^*$ is said to be recognized by machine $\mathcal{M}$ with bounded error if there exists a $p \in (\frac{1}{2}, 1]$ such that

- $f_{\mathcal{M}}(w) \geq p$ when $w \in L$ and
- $f_{\mathcal{M}}(w) \leq 1 - p$ when $w \notin L$.

Equivalently, it can be said that $L \subseteq \Sigma^*$ is recognized by machine $\mathcal{M}$ with error bound ϵ, where $\epsilon = 1 - p$ (and so $\epsilon \in [0, \frac{1}{2})$).

2.1.3.7. One-Sided Bounded Error. The language $L \subseteq \Sigma^*$ is said to be recognized by machine $\mathcal{M}$ with (positive) one-sided bounded error if there exists a $p \in (0, 1]$ such that

- $f_{\mathcal{M}}(w) \geq p$ when $w \in L$ and
- $f_{\mathcal{M}}(w) = 0$ when $w \notin L$.

Equivalently, it can be said that $L \subseteq \Sigma^*$ is recognized by machine $\mathcal{M}$ with (positive) one-sided error bound ϵ, where $\epsilon = 1 - p$ (and so $\epsilon \in [0, 1)$).

The language $L \subseteq \Sigma^*$ is said to be recognized by machine $\mathcal{M}$ with negative one-sided bounded error if there exists a $p \in (0, 1]$ such that

- $f_{\mathcal{M}}(w) = 1$ when $w \in L$ and
- $f_{\mathcal{M}}(w) \leq 1 - p$ when $w \notin L$.

Equivalently, it can be said that $L \subseteq \Sigma^*$ is recognized by machine $\mathcal{M}$ with negative one-sided error bound ϵ, where $\epsilon = 1 - p$ (and so $\epsilon \in [0, 1)$).

2.2. Turing Machines

The Turing machine (TM) models used throughout the thesis consist of a read-only input tape with a two-way tape head, a read/write work tape with a two-way tape head, and a finite state control. (The quantum versions also have a finite register that plays a part in the implementation of general quantum operations, and is used to determine whether the computation has halted, and if so, with which decision. For reasons of simplicity, this register is not included in the definition of the classical machines, since its functionality can be emulated by a suitable partition of the set of internal states without any loss of computational power.) Unless otherwise specified, both tapes are assumed to be two-way infinite and indexed by $\mathbb{Z}$.

Let w be an input string. On the input tape, $\tilde{w}$ is placed in the squares indexed by $1, \ldots, |\tilde{w}|$, and all remaining squares contain #. When the computation starts, the internal state is q_1, the input tape head and the work tape head are placed on the squares indexed by 1 and 0, respectively. Additionally, we assume that the input tape head never visits the squares indexed by 0 or $|\tilde{w}| + 1$.

The transition function of a classical TM, i.e.

$$\delta : Q \times \tilde{\Sigma} \times \Gamma \to Z^{Q \times \Gamma \times <> \times <>}, \tag{2.14}$$

or

$$z = \delta(q, \sigma, \gamma, q', \gamma', d_i, d_w) \in Z, \tag{2.15}$$

has 3 domain components and 4 range components such that whenever $z \neq 0$,

the TM – that is in state $q \in Q$ and reads symbols $\sigma \in \tilde{\Sigma}$ and $\gamma \in \Gamma$ on the input tape and the work tape, respectively – changes its state to $q' \in Q$, writes $\gamma' \in \Gamma$ on the work tape, and then updates the position of the input and work tapes with respect to $d_i \in <>$ and $d_w \in <>$, respectively, with transition value z.

In the quantum case, there is also a fifth range component Ω in order to implement general quantum operations, i.e. a symbol is written on the register ($\omega \in \Omega$). (The details are given in Section 3.).

By restricting the movement of the input tape head to $\{\downarrow, \rightarrow\}$ (and so by replacing range component corresponding to the input tape head $<>$ with $\rhd$), we obtain a one-way TM, denoted as 1TM.

If the input tape head is restricted to move only to the right ("$\rightarrow$") (and so range component $<>$ corresponding to the input tape head is completely removed from the transition function and the input tape head is automatically moved one square to the right after each transition), we obtain a realtime TM, denoted as RT-TM. Note that, after reading \$, the computation of a RT-TM must be terminated.

If a 1TM is allowed to be stationary on a symbol for only at most some fixed steps, say $d > 0$, it is called RT-TM with delay d, denoted as dRT-TM. However, note that, by allowing it to be stationary, the input tape head of a quantum Turing machine (QTMs) may be put in superposition (see [6, 7, 18] and also Chapter 3), i.e. each head in the superposition may be positioned on a different symbol of the input. This violates the idea behind the realtime computation. Therefore, for realtime QTMs with delay d, the input head is required to be classical, where $d > 0$.

In space-bounded computation, the models having more than one work tape can generally be simulated by the ones having one work tape with the same amount of space usage. On the other hand, this is not the case for RT-TMs [19, 20]. Therefore, the number of the work tapes is a parameter for RT-TMs. However, all RT-TMs presented

in this thesis have only one work tape. Additionally, we assume that the work tape of RT-TMs are one-way infinite and their left most square is indexed by 0.

Table 2.1. The list of the abbreviations of Turing machines

Types of TMs	**TM**	**1TM**	**(d-)RT-TM**
deterministic	DTM	1DTM	(d-)RT-DTM
nondeterministic	NTM	1NTM	(d-)RT-NTM
probabilistic	PTM	1PTM	(d-)RT-PTM
alternating	ATM	1ATM	(d-)RT-ATM
quantum	QTM	1QTM	(d-)RT-QTM
nondeterministic quantum	NQTM	1NQTM	(d-)RT-NQTM
classical head quantum	CQTM	1CQTM	(d-)RT-CQTM
classical head nondeterministic quantum	CNQTM	1CNQTM	(d-)RT-CNQTM

If we remove the work tape of a TM (and so domain component Γ and range component Γ with its related range component $\diamond$ are completely removed from the transition function), we obtain a finite automaton, denoted as FA.

In classical and quantum FA domains, prefix "1" has been sometimes used instead of prefix "realtime" [4,9,15,21]. This is generally not a problem for classical FAs due to their equivalence. However, this is not the case for quantum FAs (QFAs) since allowing the head to be stationary increases the power of QFAs, i.e. in bounded and unbounded error language recognition [4, 22, 23]. In the current literature, QFAs are commonly denoted using the prefix "1" and "1.5" in order to present "realtime" and "one-way" input heads, respectively [4,22–24]. In this thesis, we adopt the prefix notation of TMs for one-way and realtime models:

i. two-way finite automaton (2FA), TM with no work tape,
ii. one-way finite automaton (1FA), 1TM with no work tape,
iii. realtime finite automaton with delay d (dRT-FA), dRT-TM with no work tape, where $d > 0$, and,
iv. realtime finite automaton (RT-FA), RT-TM with no work tape.

Note that, in FA domain, two-wayness of the input head has been clearly indicated.

Table 2.2. The list of the abbreviations of finite automata

Types of FAs	**2FA**	**1FA**	**(d-)RT-FA**
deterministic	2DFA	1DFA	(d-)RT-DFA
nondeterministic	2NFA	1NFA	(d-)RT-NFA
probabilistic	2PFA	1PFA	(d-)RT-PFA
alternating	2AFA	1AFA	(d-)RT-AFA
quantum	2QFA	1QFA	(d-)RT-QFA
nondeterministic quantum	2NQFA	1NQFA	(d-)RT-NQFA
classical head quantum	2CQFA	1CQFA	(d-)RT-CQFA
classical head nondeterministic quantum	2CNQFA	1CNQFA	(d-)RT-CNQFA

A TM is said to be *unidirectional* (or *simple*) if the movements of input and work tape heads are fixed for each internal state to be entered in any transition. That is, for a unidirectional TM, denoted as uni-TM, D_i and D_w determine respectively the movements of the input and work tape heads in any transition. Thus, the corresponding range component(s) of δ's are dropped. For simplicity, each internal state of a unidirectional FA, say q, can be represented as follows:

- $\overleftarrow{q}$ if $D_i(q) =$ " $\leftarrow$",
- $\downarrow q$ if $D_i(q) =$ " $\downarrow$", and
- $\overrightarrow{q}$ if $D_i(q) =$ " $\rightarrow$".

All classical computational models presented throughout thesis are equivalent to their unidirectional counterparts. Therefore, each of those machines is considered with default unidirectional.

A configuration of a TM is the collection of

- the internal state of the machine,
- the position of the input tape head,
- the contents of the work tape, and the position of the work tape head.

$\mathcal{C}^w$, or shortly $\mathcal{C}$, denotes the set of all configurations, which is a finite set in our case of space bounded computations. Let c_i and c_j be two configurations. The value of the transition from c_i to c_j is given by the transition function δ if c_i is reachable from c_j in one step, and is zero otherwise. A *configuration matrix* is a square matrix whose rows and columns are indexed by the configurations. The $(j,i)^{th}$ entry of the matrix denotes the value of the transition from c_i to c_j. We assume c_1 as *the initial configuration*, in which, as described previously, the heads are placed on the square indexed by 1 and the internal state is q_1 (in quantum case, the finite register is prepared by the initial symbol as well).

For classical TMs, a configuration is called an *accepting configuration* (resp., a *rejecting configuration*) if its internal state belongs to the set of the accepting states (resp., the set of the rejecting states). Moreover, a configuration is called a *halting configuration* if it is an accepting or rejecting configuration or (in nondeterministic and alternating cases) there is no defined transition from it. The computation does not continue from a halting configuration. However, note that, for the models making their decisions after reading the whole input, such as RT-FAs, the configurations can be associated with adjectives "accepting" or "rejecting" only after reading symbol \$.

2.3. Pushdown Automata

A pushdown automaton (PDA) is a TM using its work tape as a stack, on which three operations are applied: *pop*, *push*, and *pop-and-push*. As a special note, the push operation on a stack can be implemented by a TM in at least two steps, however, it is a single step for a PDA. We assume that the stacks are one-way infinite and bounded from the left. The squares of a stack are indexed by nonnegative integers, where the left-most square is indexed by 0 on which $\vdash$ is placed throughout the computation. At the beginning of the computation, all squares indexed by positive integers contain symbol #. The stack head is assumed not to drop out from left. For PDAs, Γ contains $\vdash$ and not # and $\tilde{\Gamma} = \Gamma \cup \{\#\}$.

For a given input string $w \in \Sigma$, a configuration of a PDA is composed of the

following elements:

- the current internal state,
- the position of the input head, and
- the contents of the stacks.

Throughout the thesis, it is assumed that the PDAs have one stack. A two-way pushdown automaton (2PDA) has a two-way input tape head, i.e. in each transition, the position of the input tape head can be updated with respect to $\lhd\rhd$, and a one-way pushdown automaton (1PDA) is a restricted 2PDA in which the input tape head cannot move to the left, i.e. in each transition, the position of the input tape head can be updated with respect to $\rhd$. By not allowing the input tape head of a 1PDA to be stationary, we obtain a realtime pushdown automaton (RT-PDA).

The transition function of a PDA can be defined as a special case of a TM after making some small modifications for the push operation. That is, syntactically, there is no difference between the transition function of a TM and a PDA. However, a PDA has some restrictions on its transition function and the push operation has capability to change the next square (on the right side) on the stack. In the following, the details of the operations implemented on the stack are presented. Note that, all the remaining

Table 2.3. The list of the abbreviations of pushdown automata

Types of PDAs	**2PDA**	**1PDA**	**(d-)RT-PDA**
deterministic	2DPDA	1DPDA	(d-)RT-DPDA
nondeterministic	2NPDA	1NPDA	(d-)RT-NPDA
probabilistic	2PPDA	1PPDA	(d-)RT-PPDA
alternating	2APDA	1APDA	(d-)RT-APDA
quantum	2QPDA	1QPDA	(d-)RT-QPDA
nondeterministic quantum	2NQPDA	1NQPDA	(d-)RT-NQPDA
classical head quantum	2CQPDA	1CQPDA	(d-)RT-CQPDA
classical head nondeterministic quantum	2CNQPDA	1CNQPDA	(d-)RT-CNQPDA

part of the transition function is exactly the same as a TM.

Let $d_s \in <>$ be the direction of the stack head and $\gamma \in \Gamma$ and $\gamma' \in \tilde{\Gamma}$ be the symbols to be read and to be written on the stack, respectively.

- In a pop operation, $\gamma \neq$ " $\vdash$ " is overwritten by "#" and the head is moved one square to the left. (For symbol $\vdash$, the pop operation is forbidden.) Therefore, the pop operation is associated with setting of d_s to " $\leftarrow$ " and the values of all transitions having the setting of $\gamma =$ " $\vdash$ " or the setting of $\gamma' \neq$ "#" are always zero.
- In a pop-and-push operation, γ is overwritten by $\gamma' \neq$ "#" and the head stays in the same square. (It is not allowed to write symbol #.) Therefore, the pop-and-push operation is associated with setting of d_s to " $\downarrow$ " and the values of all transitions having the setting of $\gamma' =$ "#" or the setting of $\gamma =$ " $\vdash$ " and $\gamma' \neq$ " $\vdash$ " are always zero.
- In a push operation, the following two consecutive operations of a TM are combined: (i) γ is overwritten by γ and the head is moved one square to the right, (ii) symbol # is overwritten by $\gamma' \neq$ "#" and the head stays in the same square. (It is not allowed to write symbol "#".) Therefore, the push operation is associated with setting of d_s to " $\rightarrow$ " and the values of all transitions having the setting of $\gamma' =$ "#" are always zero.

As a last remark, *unidirectional* PDAs, denoted as uni-PDAs, are defined exactly in the same way as TMs. Thus, $D_s(q')$ determines the stack operation of the transition in which q' is the state to be entered.

2.4. Counter Automata

A counter automaton (CA) can be seen as a PDA with multiple stacks whose alphabets are restricted to contain two symbols, i.e. $\vdash$ and a *counting symbol*. That is, a CA can count by using its stacks and can check whether their values are zero or not. In order to make a counting with negative numbers, we also assume that the stack tapes

are two-way infinite by requiring that the squares indexed by 0 always contains $\vdash$ and the remaining squares contain either the blank symbol or counting symbol throughout the computation. Note that, there is no blank symbol between the counting symbols.

Formally, we follow the domain specific conventions of counter automata. A $k \in \mathbb{Z}^+$ counter automaton (kCA) is a finite state automaton augmented with k counters which can be modified by some amount from $\Diamond$, and where the status of these counters is also taken into account during transitions.

For a given input string $w \in \Sigma$, a configuration of a kCA is composed of the following elements:

- the current internal state,
- the position of the input head, and
- the contents of the counters.

Similar to PDAs, we define two-way k-counter automaton ($2k$CA), one-way k-counter automaton ($1k$CA), and realtime k-counter automaton (RT-kCA).

Table 2.4. The list of the abbreviations of counter automata

Types of kCAs	**2kCA**	**1kCA**	**(d-)RT-kCA**
deterministic	2DkCA	1DkCA	(d-)RT-DkCA
nondeterministic	2NkCA	1NkCA	(d-)RT-NkCA
probabilistic	2PkCA	1PkCA	(d-)RT-PkCA
alternating	2AkCA	1AkCA	(d-)RT-AkCA
quantum	2QkCA	1QkCA	(d-)RT-QkCA
nondeterministic quantum	2NQkCA	1NQkCA	(d-)RT-NQkCA
classical head quantum	2CQkCA	1CQkCA	(d-)RT-CQkCA
classical head nondeterministic quantum	2CNQkCA	1CNQkCA	(d-)RT-CNQkCA

Let k be a nonnegative integer. The transition function of a classical $2k$CA, i.e.

$$\delta : Q \times \tilde{\Sigma} \times \{\Theta^k\} \to Z^{Q \times \diamond \times \{\Diamond^k\}}, \tag{2.16}$$

or

$$z = \delta(q, \sigma, \bar{\theta}, q', d_i, \bar{c}) \in Z, \tag{2.17}$$

has 3 domain components and 3 range components such that whenever $z \neq 0$,

> the $2k$CA – that is in state $q \in Q$, reads symbols $\sigma \in \tilde{\Sigma}$ on the input tape, and has $\bar{\theta} \in \Theta^k$ on the counter(s), i.e. $\bar{\theta}[i]$ represents the status of the i^{th} counter, – changes its state to $q' \in Q$, updates the position of the input head with respect to d_i, and updates the values of the counter(s) with respect to $\bar{c}$, i.e. the value of j^{th} counter is updated by $\bar{c}[j]$, with transition value z, where $1 \leq i, j \leq k$.

For $1k$CA, the range component $<>$ is replaced by $\triangleright$ and for RT-kCA, the range component $<>$ is completely removed. In the quantum case, there is also an additional range component Ω in order to implement general quantum operations, i.e. a symbol is written on the register ($\omega \in \Omega$).

Similar to TMs, a CA is said to be *unidirectional* or *simple*, denoted as uni-CA, if the movement of the input tape head and the updates of the counters are fixed for each internal state to be entered in any transition. (The function determining the updates of the counters is D_c.) Thus, the corresponding range component(s) of δ's are dropped.

An r-reversal kCA [25], denoted as r-rev-kCA, is a kCA such that the number of alternations from increasing to decreasing and vice versa on each counter is restricted by r, where r is a nonnegative integer.

A blind kCA, denoted as a kBCA, is a kCA never knowing the status of its counter(s) (and so the domain component Θ^k is completely removed) and it requires that the value of the each counter must be zero in order to accept a given input string.

Lastly, a kCA(m) is a kCA with the capability of updating its counter by a value

from the set $\{-m,\ldots,0,\ldots,m\}$ in a single step, where $m>1$. Indeed, for any given kCA(m), an isomorphic kCA can easily be built and so such a capability does not increase the computational power of CAs. We present this fact explicitly for realtime quantum CAs in Lemma 3.4.

2.5. Formal Definition of Classical Machines

Formally, a classical TM or a classical PDA with any type of the input tape head has is a 7-tuple

$$\mathcal{P}=(Q,\Sigma,\Gamma,\delta,q_1,Q_a,Q_r). \tag{2.18}$$

For two-way or one-way classical FAs or kCAs, we use a 6-tuple

$$\mathcal{P}=(Q,\Sigma,\delta,q_1,Q_a,Q_r). \tag{2.19}$$

For the above machines, $Q_n=Q\setminus\{Q_a\cup Q_r\}$ and additionally in case of alternating machines, Q_n is composed of two disjoint subsets: Q_e, the set of *existential* states; Q_u, the set of *universal* states.

A realtime classical FA or kCA is a 5-tuple

$$\mathcal{P}=(Q,\Sigma,\delta,q_1,Q_a). \tag{2.20}$$

When it describes an alternating machine, Q is composed of two disjoint subsets: Q_e and Q_u. Moreover, for probabilistic FAs, the transition function can also be defined as a finite set of (left) stochastic matrices, i.e. $A_{\sigma\in\tilde{\Sigma}}$ and $A_\sigma[j,i]$ represents the transition probability from q_i to q_j when reading σ. Throughout the thesis, we follow this convention for those machines: a two-way or one-way PFA is a 6-tuple

$$\mathcal{P}=(Q,\Sigma,\{A_{\sigma\in\tilde{\Sigma}}\},q_1,Q_a,Q_r) \tag{2.21}$$

and a RT-PFA is a 5-tuple

$$\mathcal{P} = (Q, \Sigma, \{A_{\sigma \in \tilde{\Sigma}}\}, q_1, Q_a). \tag{2.22}$$

Similarly, the transition function of a probabilistic kCA can be represented by a collection of (left) stochastic matrices defined for each $\sigma \in \tilde{\Sigma}$ and each $\bar{\theta} \in \Theta_k$.

2.6. Computation of Machines

Let $w \in \Sigma^*$ be a given input string and $\mathcal{M}$ be a machine. For all computational models, the computation begins with the initial configuration. We use mainly *the tree structure* in order to represent the computation of classical models with the following specifications:

- the root(s) of the tree(s) is (are) always the initial configuration;
- the nodes are the configurations and the halting configurations can only be placed on the leafs;
- the edges represent the one-step transitions between the configurations;
- any path from the root to a leaf is called a *halting path*

2.6.1. Deterministic Computation

The computation is traced by a finite path, i.e. the leaf of the path is either an accepting or rejecting configuration. w is accepted if and only if the path of $\mathcal{M}$ on w ends with an accepting configuration.

2.6.2. Nondeterministic Computation

The computation is traced by a finite or infinite tree. Contrary to the deterministic case, more than one outgoing edge may be defined for a configuration. A terminating path of the tree is called *an accepting path* (resp., *rejecting path*) if its leaf is an accepting (resp., rejecting configuration). w is accepted if and only if there exists

an accepting path in the tree of $\mathcal{M}$ on w.

2.6.3. Alternating Computation

This is a generalization of nondeterministic computation. The computation is traced by a forest, a set of trees. The tree structure is similar to that of nondeterministic computation. However, each tree is allowed to have only one outgoing edge from any node corresponding to a (nonhalting) configuration having an existential state. Therefore, different nondeterministic choices lead to different computation trees. A tree is called *an accepting tree* if each leaf of the tree correspond to an accepting configuration, that is, any terminating path of the tree is an accepting path. w is accepted if and only if there exists an accepting tree in the forest of $\mathcal{M}$ on w.

2.6.4. Probabilistic Computation

We can either use tree structure or *vectors*. The tree structure of a probabilistic computation is similar to that of nondeterministic computation, in which each edge has a weight, i.e. the transition probability. Therefore, each accepting or rejecting path can also be associated with a probability, calculated by multiplying all weights of the path. (Note that, the number of accepting or rejecting paths can be infinite.) The overall accepting (resp., rejecting) probability of $\mathcal{M}$ on w, $f^a_{\mathcal{M}}(w)$ (resp., $f^r_{\mathcal{M}}(w)$), is the summation of the probabilities associated to the all accepting (resp., rejecting) paths.

In the vector representation, a column vector, called *configuration vector* or *state vector*, whose i^{th} entry corresponds to the i^{th} configuration or state, represent the probability distribution of the configuration in any step of the computation, i.e. in each step of the computation, the current vector is multiplied by the configuration matrix from the left. Note that, in the configuration matrix, the $(i,i)^{th}$ entry is always assumed to be 1 if the i^{th} column (or vector) corresponds to a halting configuration. Thus, the probability of reaching a halting configuration can be cumulatively represented by the corresponding entry in any step of the computation.

As a special case, the computation of a RT-PFA can be traced by a stochastic state vector, say v, such that $v(i)$ corresponds to state $q_i \in Q$.

$$v_i = A_{\tilde{w}_i} v_{i-1}, \tag{2.23}$$

where $1 \leq i \leq |\tilde{w}|$ and v_0 is the initial state vector whose first entry is equal to 1. The transition matrices of a RT-PFA can be extended for any string as

$$A_{w\sigma} = A_\sigma A_w, \tag{2.24}$$

where $\sigma \in \tilde{\Sigma}$, $w \in (\tilde{\Sigma})^*$, and $A_\varepsilon = I$. The probability that w is accepted by 1PFA $\mathcal{P}$ is

$$f_{\mathcal{P}}(w) = \sum_{q_i \in Q_a} (A_{\tilde{w}} v_0)(i) = \sum_{q_i \in Q_a} v_{|\tilde{w}|}(i). \tag{2.25}$$

By generalizing the linearization of RT-PFA, Turakainen [26] defined a more general computational model, called *generalized finite automaton* (GFA). The details of a GFA are given in Figure 2.1.

2.6.5. Quantum Computation

In this part, we roughly explain quantum computation[4] by using the classical representation tools, tree structure and vectors together.

A *pure* quantum state, or shortly, a quantum state, is represented by a column vector, where each entry represents the amplitude of being in the classical state that is assumed as the configurations of the quantum machines throughout the thesis.

In quantum computation, as a part of the one-step transition, some symbols are written on the finite register (and then the register is discarded either after a

[4]The formal definitions and detailed explanations are in Chapter 3 and Appendix B.

A GFA is formally a 5-tuple

$$\mathcal{G} = (Q, \Sigma, \{A_{\sigma\in\Sigma}\}, v_0, f), \tag{2.26}$$

where

i. $A_{\sigma\in\Sigma}$ are $|Q| \times |Q|$-dimensional real valued transition matrices;

ii. v_0 and f are real valued *initial* (column) and *final* (row) vectors, respectively.

Similar to what we have for RT-PFAs, the transition matrices of a GFA can be extended for any string. For a given input string, $w \in \Sigma$, the acceptance value associated by GFA $\mathcal{G}$ to string w is

$$f_{\mathcal{G}}(w) = fA_{w_{|w|}} \cdots A_{w_1} v_0 = fA_w v_0. \tag{2.27}$$

Figure 2.1. Generalized finite automaton

measurement or without any measurement). For each written symbol, the current quantum state is transformed into a new quantum state with the probability of the square of the euclidean norm of the new quantum state. (Once the finite register is discarded, the new quantum states are normalized[5] .)

Similar to the probabilistic machines, the computation of a quantum machine can be represented by a tree with some exceptions: the nodes of the tree are the quantum states instead of the configurations (and so the root of the tree is the initial quantum state, where the entry of the initial configuration is 1 and the remaining entries are zeros); the edges are again associated with the nonzero probabilities and a finite register symbol; the leafs of the tree are the halting (accepting or rejecting) quantum states, i.e. if the incoming edge of a quantum state is associated with an accepting (resp., a rejecting) finite register symbol, then it is called an accepting (resp., a rejecting) quantum state. As a special remark, a symbol written on the finite register may

[5]Although only one of them survives and so is normalized according to any particular observer, we use a "God's eye view" and assume that all of them survive with related probabilities.

not always been associated with an edge in the tree due to destructive interference interference.

The overall accepting and rejecting probabilities of a quantum machine on an input can be calculated in the same way as that of a probabilistic machine.

2.7. Space Classes

Let s be a function of the form $s : \mathbb{N} \to \mathbb{R}$.

The space used by a path (a tree or a forest) is the difference between the leftmost visited square and the right most visited square when considering all the configurations in the structure. Note that, in the quantum case, we restrict ourselves only to the configurations having nonzero amplitude in the quantum states.

A machine is said to be strongly $s(|w|)$ space-bounded if the space used by the related computational structure (tree or forest) of w is no more than $s(|w|)$, for any input string $w \in \Sigma^*$.

A machine is said to be middle $s(|w|)$ space-bounded if the space used by the related computational structure (tree or forest) of w is no more than $s(|w|)$, for any accepted input string $w \in \Sigma^*$.

A nondeterministic or alternating machine is said to be weak $s(|w|)$ space-bounded if the space used by an accepting path or tree, respectively, is no more than $s(|w|)$, for any accepted input string $w \in \Sigma^*$.

The generic name of space classes is **X**SPACE, where **X** is replaced by suitable letters depending on the TM and/or error types. In probabilistic and quantum computation, the subscript of "SPACE", i.e. $\mathrm{SPACE}_{\mathbb{Y}}$, if it exists, specifically denotes the class of the numbers $\mathbb{Y} \in \{\mathbb{Q}, \mathbb{A}, \tilde{\mathbb{R}}, \tilde{\mathbb{C}}, \mathbb{R}, \mathbb{C}\}$, to which the transition values of the corresponding TMs are restricted, where $\tilde{\mathbb{R}}$ and $\tilde{\mathbb{C}}$ denote the computable real and complex

numbers, respectively. (Note that, a complex number (and so a real number) is computable if its real and imaginary parts are computed within the precision of 2^{-n} by a deterministic algorithm in time polynomial in n [18].) By default, probabilistic (resp., quantum) space classes are defined for $\mathbb{R}$ (resp., $\mathbb{C}$).

There may be two more prefixes before the class names as well (when used together, the order below is followed):

i. the kind of the space usage, i.e. "weak-" or "middle-", and
ii. the type of the input tape head, i.e. "one-way-" or "realtime-".

The list of the class names used throughout the thesis are as follows:

- DSPACE(s), NSPACE(s), and ASPACE(s) denote the classes of the languages recognized by DTMs, NTMs, and ATMs, respectively, in space s.
- PrSPACE(s) and PrQSPACE(s) denote the classes of the languages recognized by PTMs and QTMs, respectively, in space s with unbounded error.
- BPSPACE(s) and BQSPACE(s) denote the classes of the languages recognized by PTMs and QTMs, respectively, in space s with bounded error.
- NQSPACE(s) denote the classes of the languages recognized by QTMs in space s with one-sided unbounded error setting.
- $C_=$SPACE(s) and $C_=$QSPACE(s) denote the classes of the languages recognized by PTMs and QTMs, respectively, in space s, with inclusive cutpoint $\frac{1}{2}$ providing that each computation must be halted in finite step.

For the pushdown and the counter machines, we use "STACK" and "COUNTER" complexity classes, respectively, in order to represent the space usage of the machines on their storage devices with respect to the length of the input string, i.e. they are the counterparts of "SPACE" complexity class. Moreover, kSTACK and kCOUNTER denote the classes defined by the machines having k stacks and k counters, respectively. As a generic class name, CFL is the class of languages recognized by 1NPDAs.

In the finite automata domain, the classes also have domain specific names.

The class of the languages recognized by RT-DFAs (and 1DFAs) are regular languages, denoted as REG. In the finite automata domain, neither two-wayness nor nondeterminism increases the computational power of the deterministic machines [21, 27]. That is,

$$\text{DSPACE}(1)=\text{NSPACE}(1)=\text{REG}. \tag{2.28}$$

RT-PFAs, GFAs [26], and 2PFAs [28] recognize the same class of languages with (strict) cutpoint. This is the class of *stochastic languages*, denoted by S. The class of languages recognized by these machines with nonstrict cutpoint is denoted by coS. The class of languages recognized by RT-PFAs, GFAs, and 2PFAs with unbounded error is therefore $\text{S} \cup \text{coS}$, and is denoted by uS.

$$\text{PrSPACE}(1)=\text{uS}. \tag{2.29}$$

One-way-$\text{C}_=$SPACE(1) is also denoted as $\text{S}^=$. The complementary class of $\text{S}^=$, denoted as $\text{S}^{\neq}$, is known as exclusive stochastic languages [29].

The class of languages recognized by RT-QFAs (Section 3.2.4) with cutpoint is denoted by QAL. The class of languages recognized by these machines with nonstrict cutpoint is denoted by coQAL. $\text{QAL} \cup \text{coQAL}$ is denoted by uQAL. The class of the languages recognized by MCQFA (Section 3.2.5) with cutpoint is Moore-Crotchfield languages, denoted as MCL. coMCL and uMCL denote respectively complementary class of MCL and the class formed by $\text{MCL} \cup \text{coMCL}$.

Brodsky and Pippenger [9] defined the class of languages recognized by RT-KWQFAs (Section 3.2.5) with unbounded error, denoted as UMM, in a way that is slightly different than the standard approach: $L \in \text{UMM}$ if and only if there exists a

RT-KWQFA $\mathcal{M}$ such that

- $f_{\mathcal{M}}(w) > \lambda$ when $w \in L$ and
- $f_{\mathcal{M}}(w) < \lambda$ when $w \notin L$,

for some $\lambda \in [0, 1]$.

The languages recognized by RT-NQFAs are nondeterministic quantum automaton languages, denoted as NQAL. NMCL denotes the class of the languages recognized by nondeterministic MCQFAs (MCNQFAs).

In the bounded error setting, the class of the languages recognized by RT-PFAs (and 1PFAs) are regular languages [15]. On the other hand, 2PFAs can recognize some nonregular languages with bounded error [3].

$$\text{REG} = \text{one-way-BPSPACE}(1) \subsetneq \text{BPSPACE}(1). \tag{2.30}$$

The class of languages recognized by RT-PostPFAs and RT-PostQFAs (Section 5.4.1) with bounded error are PostS (*post-stochastic languages*) and PostQAL (*post-quantum automaton languages*), respectively. Additionally, the class of languages recognized by RT-LPostPFAs and RT-LPostQFAs (Section 5.4.3) with bounded error are LPostS and LPostQAL, respectively.

3. A NEW KIND OF QUANTUM MACHINE

After a general framework for the quantum machines with some introductory concepts (Section 3.1), we present firstly a new kind of quantum Turing machine[6] with its FA variants (Section 3.2), and then, based on it, quantum pushdown and stack machines (Sections 3.3 and 3.4). We also give some basic facts in this chapter.

3.1. The General Framework for Quantum Machines

In accordance with quantum theory, a quantum machine can be in a *superposition* of more than one configuration at the same time. The "weight" of each configuration in such a superposition is called its *amplitude*. Unlike the case with classical machines, these amplitudes are not restricted to being positive real numbers, and that is what gives quantum computers their interesting features. A superposition of configurations

$$|\psi\rangle = \alpha_1|c_1\rangle + \alpha_2|c_2\rangle + \cdots + \alpha_n|c_n\rangle \tag{3.1}$$

can be represented by a column vector $|\psi\rangle$ with a row for each possible configuration, where the i^{th} row contains the amplitude of the corresponding configuration in $|\psi\rangle$.

If our knowledge that the quantum system under consideration is in superposition $|\psi\rangle$ is certain, then $|\psi\rangle$ is called a *pure state*, and the vector notation described above is a suitable way of manipulating this information. However, in some cases (e.g. during classical probabilistic computation), we only know that the system is in state $|\psi_l\rangle$ with probability p_l for an ensemble of pure states $\{(p_l, |\psi_l\rangle)\}$,where $\sum_l p_l = 1$. A convenient representation tool for describing quantum systems in such *mixed states* is the density matrix. The *density matrix*[7] representation of $\{(p_l, |\psi_l\rangle) \mid 1 \leq l \leq M < \infty\}$ is

$$\rho = \sum_l p_l |\psi_l\rangle\langle\psi_l|. \tag{3.2}$$

[6]For descriptions of several other QTM variants, we refer the reader to [6–8, 18, 30, 31].

[7]The trace of a density matrix is 1, and each density matrix is positive semidefinite.

We use both these representations for quantum states in the thesis. We refer the reader to Appendix B for further details.

A quantum machine is distinguished from a classical machine by the presence of the items Ω (the finite register alphabet) and Δ (the set of possible outcomes associated with the measurements of the finite register). Note that, Ω is partitioned into $|\Delta| = k$ subsets $\Omega_{\tau_1}, \ldots, \Omega_{\tau_k}$. As a part of each transition, a quantum machine has the following phases:

i. *pre-transition phase*: initialize the finite register, i.e. the register is set to "ω_1";
ii. *transition phase*: in addition to the transition of the classical machines, update the content of the register;
iii. *post-transition phase*: make a selective measurement on the finite register with the outcome set Δ and then discard it[8] .

Since we do not consider the register content as part of the configuration, the register can be seen as the "environment" interacting with the "principal system" that is the rest of the quantum machine [32]. δ therefore induces a set of configuration transition matrices, $\{E_{\omega\in\Omega}\}$, where the $(i,j)^{th}$ entry of E_ω, the amplitude of the transition from c_j to c_i by writing $\omega \in \Omega$ on the register, is defined by δ whenever c_j is reachable from c_i in one step, and is zero otherwise. The $\{E_{\omega\in\Omega}\}$ form an operator $\mathcal{E}$, with operation elements $\mathcal{E}_{\tau_1} \cup \mathcal{E}_{\tau_2} \cup \cdots \cup \mathcal{E}_{\tau_k}$, where $\mathcal{E}_{\tau\in\Delta} = \{E_{\omega\in\Omega_\tau}\}$.

According to the modern understanding of quantum computation [33], a quantum machine is said to be *well-formed*[9] if $\mathcal{E}$ is a superoperator (selective quantum operator), i.e.

$$\sum_{\omega\in\Omega} E_\omega^\dagger E_\omega = I. \tag{3.3}$$

[8] In some realtime quantum machines, such a selective measurement can be done only at the end of the computation.

[9] We also refer the reader to [18] for a detailed discussion of the well-formedness of QTMs that evolve unitarily.

$\mathcal{E}$ can be represented by a $|\mathcal{C}||\Omega| \times |\mathcal{C}|$-dimensional matrix E (Figure 3.1) by concatenating each $E_{\omega \in \Omega}$ one under the other. It can be verified that $\mathcal{E}$ is a superoperator if and only if the columns of E form an orthonormal set.

	c_1	c_2	...	$c_{\|\mathcal{C}\|}$
c_1				
c_2				
$\vdots$		E_{ω_1}		
$c_{\|\mathcal{C}\|}$				
c_1				
c_2				
$\vdots$		E_{ω_2}		
$c_{\|\mathcal{C}\|}$				
c_1				
c_2				
$\vdots$		$\vdots$		
$c_{\|\mathcal{C}\|}$				
c_1				
c_2				
$\vdots$		$E_{\omega_{\|\Omega\|}}$		
$c_{\|\mathcal{C}\|}$				

(3.4)

Figure 3.1. The matrix representation of superoperators (E)

Let c_{j_1} and c_{j_2} be two configurations with corresponding columns v_{j_1} and v_{j_2} in E. For an orthonormal set to be formed, we must have

$$v_{j_1}^{\dagger} \cdot v_{j_2} = \begin{cases} 1 & j_1 = j_2 \\ 0 & j_1 \neq j_2 \end{cases} \tag{3.5}$$

for all such pairs. This constraint imposes some easily checkable restrictions on the transition function (δ).

The initial density matrix of a quantum machine is represented by $\rho_0 = |c_1\rangle\langle c_1|$, where c_1 is the initial configuration corresponding to the given input string. Note that, unless otherwise specified, Δ is set to $\{n, a, r\}$.

3.2. Quantum Turing Machines

We define a quantum Turing machine (QTM) $\mathcal{M}$ to be a 7-tuple

$$M = (Q, \Sigma, \Gamma, \Omega, \delta, q_1, \Delta). \tag{3.6}$$

We refer the reader to Appendix A.1 for the list of the local conditions for QTM wellformedness.

3.2.1. Two-Way Quantum Finite Automata

The two-way quantum finite automaton (2QFA) is obtained by removing the work tape of the QTM:

$$\mathcal{M} = (Q, \Sigma, \Omega, \delta, q_1, \Delta). \tag{3.7}$$

See below for a list of easily checkable local conditions for wellformedness of 2QFAs. Let x_1 and x_2 denote the positions of the input tape heads. In order to evolve to the same configuration in one step, the difference between x_1 and x_2 must be at most 2. Therefore, we obtain a total of 13 different cases, listed below, that completely define the restrictions on the transition function. Note that, by taking the conjugates of each summation, we handle the symmetric cases that are shown in the parentheses. For $q_1, q_2 \in Q; \sigma \in \tilde{\Sigma}$,

1. $x_1 = x_2$:

$$\sum_{q' \in Q, d \in <>, \omega \in \Omega} \overline{\delta(q_1, \sigma, q', d, \omega)} \delta(q_2, \sigma, q', d, \omega) = \begin{cases} 1 & q_1 = q_2 \\ 0 & \text{otherwise} \end{cases} \tag{3.8}$$

2. $x_1 = x_2 - 1$ $(x_1 = x_2 + 1)$:

$$\sum_{q' \in Q, \omega \in \Omega} \overline{\delta(q_1, \sigma, q', \rightarrow, \omega)} \delta(q_2, \sigma, q', \downarrow, \omega) + \overline{\delta(q_1, \sigma, q', \downarrow, \omega)} \delta(q_2, \sigma, q', \leftarrow, \omega) = 0. \tag{3.9}$$

3. $x_1 = x_2 - 2$ $(x_1 = x_2 + 2)$:

$$\sum_{q' \in Q, \omega \in \Omega} \overline{\delta(q_1, \sigma, q', \rightarrow, \omega)} \delta(q_2, \sigma, q', \leftarrow, \omega) = 0. \tag{3.10}$$

3.2.2. Unidirectional Quantum Turing Machines

If the QTM is unidirectional, wellformedness can be checked using the simpler conditions in Figure 3.2. Removing the reference to worktape symbols, we obtain the analogous constraints for unidirectional 2QFAs as shown in Figure 3.3.

For $q_1, q_2 \in Q; \sigma \in \tilde{\Sigma}; \gamma_1, \gamma_2 \in \Gamma$,

$$\sum_{q' \in Q, \gamma' \in \Gamma, \omega \in \Omega} \overline{\delta(q_1, \sigma, \gamma_1, q', \gamma', \omega)} \delta(q_2, \sigma, \gamma_2, q', \gamma', \omega) = \begin{cases} 1 & q_1 = q_2 \text{ and } \gamma_1 = \gamma_2 \\ 0 & \text{otherwise} \end{cases}. \tag{3.11}$$

Figure 3.2. The local conditions for unidirectional QTM wellformedness

For $q_1, q_2 \in Q; \sigma \in \tilde{\Sigma}$,

$$\sum_{q' \in Q, \omega \in \Omega} \overline{\delta(q_1, \sigma, q', \omega)} \delta(q_2, \sigma, q', \omega) = \begin{cases} 1 & q_1 = q_2 \\ 0 & \text{otherwise} \end{cases}. \tag{3.12}$$

Figure 3.3. The local conditions for unidirectional 2QFA wellformedness

As is the case with PTMs, the transition function of a uni-QTM can be specified easily by transition matrices of the form $\{E_{\sigma,\omega}\}$, whose rows and columns are indexed by (internal state, work tape symbol) pairs for each $\sigma \in \tilde{\Sigma}$ and $\omega \in \Omega$. It can be verified that the wellformedness condition is then equivalent to the requirement that,

for each $\sigma \in \tilde{\Sigma}$,

$$\sum_{\omega \in \Omega} E_{\sigma,\omega}^{\dagger} E_{\sigma,\omega} = I. \tag{3.13}$$

Similarly, for each $\sigma \in \tilde{\Sigma}$ and $\omega \in \Omega$, well-formed unidirectional 2QFAs can be described by transition matrices of the form $\{E_{\sigma,\omega}\}$, whose rows and columns are indexed by internal states, such that for each $\sigma \in \tilde{\Sigma}$,

$$\sum_{\omega \in \Omega} E_{\sigma,\omega}^{\dagger} E_{\sigma,\omega} = I. \tag{3.14}$$

Open Problem 1. Given a QTM (resp., 2QFA) $\mathcal{M}$, does there always exist a uni-QTM (resp., uni-2QFA) $\mathcal{M}'$ such that

$$f_{\mathcal{M}}(w) = f_{\mathcal{M}'}(w) \tag{3.15}$$

for all $w \in \Sigma^*$?

3.2.3. Quantum Turing Machines with Classical Heads

Although our definition of space usage as the number of work tape squares used during the computation is standard in the study of small as well as large space bounds [2, 34, 35], some researchers prefer to utilize QTM models where the tape head locations are classical (i.e. the heads do not enter quantum superpositions) to avoid the possibility of using quantum resources that increase with input size for the implementation of the heads [5, 8]. For details of this specialization of our model, which we call the QTM with classical heads (CQTM)

To specialize our general QTM model in order to ensure that the head positions are classical, we associate combinations of head movements with measurement

outcomes. There are 9 different pairs of possible movement directions, i.e.

$$<>^2 = \{\leftarrow, \downarrow, \rightarrow\} \times \{\leftarrow, \downarrow, \rightarrow\}, \tag{3.16}$$

for the input and work tape heads, and so we can classify register symbols with the function

$$D_r : \Omega \rightarrow <>^2. \tag{3.17}$$

We have $D_r(\omega) = (\downarrow, \downarrow)$ if $\omega \in \Omega_a \cup \Omega_r$. We split Ω_n into 9 parts, i.e.

$$\Omega_n = \bigcup_{d_i, d_w \in <>} \Omega_{n,d_i,d_w}, \tag{3.18}$$

where

$$\Omega_{n,d_i,d_w} = \{\omega \in \Omega_n \mid D_r(\omega) = (d_i, d_{_w})\}. \tag{3.19}$$

Therefore, the outcome set have 11 elements, represented as triples, specified as follows:

i. "(n, d_i, d_w)": the computation continuous and the positions of the input and work tape heads are updated with respect to d_i and d_w, respectively;
ii. "$(a, \downarrow, \downarrow)$": the computation halts and the input is accepted with no head movement;
iii. "$(r, \downarrow, \downarrow)$": the computation halts and the input is rejected with no head movement.

The transition function of CQTMs are specified so that when the CQTM is in state q and reads σ and γ respectively on the input and work tapes, it enters state q', and writes γ' and ω respectively on the work tape and the finite register with the

amplitude

$$\delta(q, \sigma, \gamma, q', \gamma', \omega). \tag{3.20}$$

Since the update of the positions of the input and work tape heads is performed classically, it is no longer a part of the transitions. Note that the transition function of 2QFAs with classical head (2CQFAs) [5] is obtained by removing the mention of the work tape from the above description.

Moreover, as with unidirectional QTMs (resp. unidirectional 2QFAs), for each $\sigma \in \tilde{\Sigma}$ and $\omega \in \Omega$, CQTMs (2CQFAs) can be described by transition matrices $\{E_{\sigma,\omega}\}$ satisfying the same properties.

As also argued in [8], CQTMs are sufficiently general for simulating any classical TM. We present a trivial simulation.

Lemma 3.1. *CQTMs can simulate any PTM exactly.*

Proof. Let $\mathcal{P} = (Q, \Sigma, \Gamma, \delta_{\mathcal{P}}, Q_a, Q_r)$ be a PTM and $\mathcal{M} = (Q, \Sigma, \Gamma, \Omega, \delta_{\mathcal{M}}, \Delta)$ be the CQTM simulating $\mathcal{P}$. For each $(q, \gamma, q', \gamma') \in Q \times \Gamma \times Q \times \Gamma$, we define a register symbol ω such that

i. if $q' \in Q_a$: $\omega \in \Omega_{(a,\downarrow,\downarrow)}$;
ii. if $q' \in Q_r$: $\omega \in \Omega_{(r,\downarrow,\downarrow)}$;
iii. if $q' \in Q_n$: $\omega \in \Omega_{(n,D_i(q'),D_w(q'))}$.

We conclude with setting

$$\delta_{\mathcal{M}}(q, \sigma, \gamma, q', \gamma', \omega) = \sqrt{\delta_{\mathcal{P}}(q, \sigma, \gamma, q', \gamma')}, \tag{3.21}$$

i.e. the quantum transitions behave exactly as if they are probabilistic. □

In fact, this result can be extended for other kind of machines, such as, PDAs, CAs, and FAs.

Watrous' QTM model in [8], which we call Wa03-QTM for ease of reference, is a CQTM variant that has an additional classical work tape and classical internal states. Every Wa03-QTM can be simulated exactly (i.e. preserving the same acceptance probability for every input) by CQTMs with only some time overhead. Note that Wa03-QTMs allow only algebraic transition amplitudes by definition.

3.2.4. Realtime Quantum Finite Automata

Let us consider realtime versions of 2QFAs, whose tape heads are forced by definition to have classical locations. If the quantum machine model used is sufficiently general, then the intermediate measurements can be postponed easily to the end of the algorithm in realtime computation. That final measurement can be performed on the set of internal states, rather than the finite register. Therefore, as with RT-FAs, we specify a subset of the internal states of the machine as the collection of accepting states, denoted as Q_a.

A realtime quantum finite automaton (RT-QFA) [36] is a 5-tuple

$$\mathcal{M} = (Q, \Sigma, \{\mathcal{E}_{\sigma\in\tilde{\Sigma}}\}, q_1, Q_a), \tag{3.22}$$

where each $\mathcal{E}_\sigma$ is an operator having elements $\{E_{\sigma,1}, \ldots, E_{\sigma,k}\}$ for some $k \in \mathbb{Z}^+$ satisfying

$$\sum_{i=1}^{k} E_{\sigma,i}^\dagger E_{\sigma,i} = I. \tag{3.23}$$

Additionally, we define the projector

$$P_a = \sum_{q\in Q_a} |q\rangle\langle q| \tag{3.24}$$

in order to check for acceptance. For any given input string $w \in \Sigma^*$, $\tilde{w}$ is placed on the tape, and the computation can be traced by density matrices

$$\rho_j = \mathcal{E}_{\tilde{w}_j}(\rho_{j-1}) = \sum_{i=1}^{k} E_{\tilde{w}_j,i}\rho_{j-1}E_{\tilde{w}_j,i}^{\dagger}, \tag{3.25}$$

where $1 \leq j \leq |\tilde{w}|$ and $\rho_0 = |q_1\rangle\langle q_1|$ is the initial density matrix. The transition operators can be extended easily for any string as

$$\mathcal{E}_{w\sigma} = \mathcal{E}_w \circ \mathcal{E}_\sigma, \tag{3.26}$$

where $\sigma \in \tilde{\Sigma}$, $w \in (\tilde{\Sigma})^*$, and $\mathcal{E}_\varepsilon = I$. Note that, if $\mathcal{E} = \{E_i \mid 1 \leq i \leq k\}$ and $\mathcal{E}' = \{E'_i \mid 1 \leq j \leq k'\}$, then

$$\mathcal{E}' \circ \mathcal{E} = \{E'_j E_i \mid 1 \leq i \leq k, 1 \leq j \leq k'\}. \tag{3.27}$$

The probability that RT-QFA $\mathcal{M}$ accepts w is

$$f_{\mathcal{M}}(w) = tr(P_a\mathcal{E}_{\tilde{w}}(\rho_0)) = tr(P_a\rho_{|\tilde{w}|}). \tag{3.28}$$

Lemma 3.2. *For a given RT-QFA $\mathcal{M}$ with n internal states, there exists a GFA $\mathcal{G}$ with n^2 internal states such that $f_{\mathcal{M}}(w) = f_{\mathcal{G}}(w)$ for all $w \in \Sigma^*$.*

Proof. Let $\mathcal{M} = (Q_1, \Sigma, \mathcal{E}_{\sigma\in\tilde{\Sigma}}, q_1, Q_a)$ be the RT-QFA with n internal states, and let each $\mathcal{E}_{\sigma\in\tilde{\Sigma}}$ have k elements, without loss of generality. We construct GFA $\mathcal{G} = (Q_2, \Sigma, \{A_{\sigma\in\Sigma}\}, v_0, f)$ with $2n^2$ internal states. We use mapping *vec* described in Figure 3.4 in order to linearize the computation of $\mathcal{M}$ so that it can be traced by GFA $\mathcal{G}$.

We define

$$v'_0 = vec(\rho_1), \tag{3.32}$$

(Page 73 in [8]) Let A, B, and C be $n \times n$ dimensional matrices. *vec* is a linear mapping from $n \times n$ matrices to n^2 dimensional (column) vectors defined as

$$vec(A)[(i-1)n + j] = A[i, j], \tag{3.29}$$

where $1 \leq i, j \leq N$. One can verify the following properties:

$$vec(ABC) = (A \otimes C^{\mathrm{T}})vec(B) \tag{3.30}$$

and

$$tr(A^{\mathrm{T}}B) = vec(A)^{\mathrm{T}}vec(B). \tag{3.31}$$

Figure 3.4. The definition and properties of *vec*

where

$$\rho_1 = \mathcal{E}_{¢}(\rho_0) = \sum_{i=1}^{k} E_{¢,i}\rho_0 E_{¢,i}^{\dagger}. \tag{3.33}$$

For each $\sigma \in \Sigma$, we define

$$A'_{\sigma} = \sum_{i=1}^{k} E_{\sigma,i} \otimes E_{\sigma,i}^{*} \tag{3.34}$$

and so we obtain (by Equation 3.30)

$$\rho' = \mathcal{E}_{\sigma}(\rho) = \sum_{i=1}^{k} E_{\sigma,i}\rho E_{\sigma,i}^{\dagger} \rightarrow vec(\rho') = A'_{\sigma}vec(\rho), \tag{3.35}$$

for any density matrix ρ. Finally, we define

$$f' = vec(P_a)^T \sum_{i=1}^{k} E_{\$,i} \otimes E_{\$,i}^{*}. \tag{3.36}$$

It can be verified by using Equation 3.31 that for any input string $w \in \Sigma^*$,

$$f' A'_{w_{|w|}} \cdots A'_{w_1} v'_0 = tr(P_a \mathcal{E}_{\$} \circ \mathcal{E}_w \circ \mathcal{E}_{¢}(\rho_0)) = f_{\mathcal{M}}(w). \tag{3.37}$$

The complex entries of v'_0, $\{A'_{\sigma \in \Sigma}\}$, and f' can be replaced [37] with 2×2 dimensional real matrices[10] , and so we obtain the equations

$$\begin{pmatrix} f_{\mathcal{M}}(w) & 0 \\ 0 & f_{\mathcal{M}}(w) \end{pmatrix} = f'' A''_{w_{|w|}} \cdots A''_{w_1} v''_0, \tag{3.38}$$

where the terms with double primes are obtained from the corresponding terms with single primes. We finish the construction of $\mathcal{G}$ by stating that

i. v_0 is the first column of v''_0,
ii. A_σ is equal to A''_σ, for each $\sigma \in \Sigma$, and
iii. f is the first row of f''.

We also refer the reader to [37, 38] presenting similar constructions for other types of realtime QFAs. An alternative demonstration, not using the density matrix formalism, can be found in [39]. □

Corollary 3.3. *$QAL = S$.*

We therefore have that realtime unbounded-error probabilistic and quantum finite automata are equivalent in power. We show in Section 4.1 that this equivalence does not carry over to the one-way and two-way cases.

3.2.5. Quantum Turing Machines with Restricted Measurements

In another specialization of the QTM model, the *QTM with restricted measurements*, the machine is unidirectional, the heads can enter quantum superpositions,

[10] $a + bi$ is replaced with $\begin{pmatrix} a & b \\ -b & a \end{pmatrix}$.

$\Delta = \{n, a, r\}$, and $|\Omega_n| = |\Omega_a| = |\Omega_r| = 1$. The first family of QTMs that was formulated for the analysis of space complexity issues [6, 7], which we call the Wa98-QTM, corresponds to such a model, with the added restriction that the transition amplitudes are only allowed to be rational numbers. The finite automaton versions of QTMs with restricted measurements[11] are known as Kondacs-Watrous quantum finite automata, and abbreviated as 2KWQFAs, 1KWQFA, or RT-KWQFAs, depending on the set of allowed directions of movement for the input head. These are pure state models, since the non-halting part of the computation is always represented by a single quantum state. Therefore, configuration or state vectors, rather than the density matrix formalism, can be used in order to trace the computation easily. To be consistent with the literature on 2KWQFAs, we specialize the 2QFA model by the following process:

i. The finite register does not need to be refreshed, since the computation continuous if and only if the initial symbol is observed.
ii. In fact, 2KWQFAs do not need to have the finite register at all, instead, similarly to 2PFAs, the set of internal states of the 2KWQFA is partitioned to sets of nonhalting, accepting, and rejecting states, denoted as Q_n, Q_a, and Q_r, respectively, which can be obtained easily by taking the tensor product of the internal states of the 2QFA and the set $\{n, a, r\}$.
iii. A configuration is designated as nonhalting (resp., accepting or rejecting), if its internal state is a member of Q_n (resp., Q_a or Q_r). Nonhalting (resp., accepting or rejecting) configurations form the set $\mathcal{C}_n^w$ (resp., $\mathcal{C}_a^w$ or $\mathcal{C}_r^w$) (for a given input string $w \in \Sigma^*$).
iv. The evolution of the configuration sets can be represented by a unitary matrix.
v. The measurement is done on the configuration set with projectors P_n, P_a, and P_r, defined as

$$P_{\tau \in \{n,a,r\}} = \sum_{c \in \mathcal{C}_\tau^w} |c\rangle\langle c| \tag{3.39}$$

for a given input string $w \in \Sigma^*$, where the standard actions are associated with

[11] These models, which also allow unrestricted transition amplitudes by the convention in automata theory, are introduced in the paper written by Kondacs and Watrous [4].

the outcomes "n", "a", and "r".

Formally, a 2KWQFA is a 6-tuple

$$\mathcal{M} = \{Q, \Sigma, \delta, q_1, Q_a, Q_r\}, \tag{3.40}$$

where $Q_n = Q \setminus \{Q_a \cup Q_r\}$ and $q_1 \in Q_n$. δ induces a unitary matrix U_σ, whose rows and columns are indexed by internal states for each input symbol σ. Note that, since all 2KWQFAs are unidirectional, we use the notations $\overleftarrow{q}$,$\downarrow q$, and $\overrightarrow{q}$ for internal state q in order to represent the value of $D_i(q)$ as $\leftarrow$, $\downarrow$, and $\rightarrow$, respectively.

A RT-KWQFA is a 6-tuple

$$\mathcal{M} = \{Q, \Sigma, \{U_{\sigma \in \tilde{\Sigma}}\}, q_1, Q_a, Q_r\}, \tag{3.41}$$

where $\{U_{\sigma \in \tilde{\Sigma}}\}$ are unitary transition matrices. In contrast to the other kinds of realtime finite automata, a RT-KWQFA is measured at each step during computation after the unitary transformation is applied. The projectors are defined as

$$P_{\tau \in \Delta} = \sum_{q \in Q_\tau} |q\rangle\langle q|. \tag{3.42}$$

The nonhalting portion of the computation of a RT-KWQFA can be traced by a state vector, say $|u\rangle$, such that $|u\rangle[i]$ corresponds to state q_i. The computation begins with $|u_0\rangle = |q_1\rangle$. For a given input string $w \in \Sigma^*$, at step j $(1 \leq j \leq |\tilde{w}|)$:

$$|u_j\rangle = P_n U_{\tilde{w}_j} |u_{j-1}\rangle, \tag{3.43}$$

the input is accepted with probability

$$||P_a U_{\tilde{w}_j} |u_{j-1}\rangle||^2, \tag{3.44}$$

and rejected with probability

$$||P_r U_{\tilde{w}_j}|u_{j-1}\rangle||^2. \tag{3.45}$$

The overall acceptance and rejection probabilities are accumulated by summing up these values at each step. Note that, the state vector representing the nonhalting portion is not normalized in the description given above.

The most restricted QFA model is the Moore-Crutchfield QFA (MCQFA) [37], which can be seen as a special case of RT-KWQFA such that only a unique measurement is done after reading symbol \$.

3.3. Quantum Pushdown Automata

Two-way quantum pushdown automaton (2QPDA), one-way quantum pushdown automaton (1QPDA), and realtime quantum pushdown automaton (RT-QPDA) can all be represented to be a 7-tuple

$$\mathcal{P} = (\Sigma, \Gamma, \Omega, Q, \delta, q_1, \Delta). \tag{3.46}$$

We refer the reader to Appendix A.2 for the list of the local conditions for 2QPDAs, 1QPDAs, and RT-QPDAs wellformedness.

If a 2QPDA (resp., 1QPDA or RT-QPDA) is unidirectional, then there is only one local condition for well-formedness: for any choice of $q_1, q_2 \in Q$, $\sigma \in \tilde{\Sigma}$, and $\gamma_1 \in \Gamma, \gamma_2 \in \tilde{\Gamma}$,

$$\sum_{q' \in Q, \gamma' \in \Upsilon, \omega \in \Omega} \overline{\delta(q_1, \sigma, \gamma_1, q', \gamma', \omega)} \delta(q_2, \sigma, \gamma_2, q', \gamma', \omega) = \begin{cases} 1 & q_1 = q_2 \text{ and } \gamma_1 = \gamma_2 \\ 0 & \text{otherwise} \end{cases}. \tag{3.47}$$

In the unidirectional case, we can define an admissible operator for each $\sigma \in \tilde{\Sigma}$, i.e. $\mathcal{E}_\sigma = \{E_{\sigma,\omega}\}$, where $\omega \in \Omega$ and $E_{\sigma,\omega}[j,i]$ represents the amplitude of the transition $\delta(q_i, \sigma, \gamma_i, q_j, \gamma_j, \omega)$, where (q_i, γ_i) and (q_j, γ_j) are the pairs corresponding to the i^{th} and j^{th} columns (rows), respectively, and $q_i, q_j \in Q; \gamma_i, \gamma_j \in \Upsilon$.

It is an open problem whether the computation powers of 2QPDAs (resp., 1QPDAs or RT-QPDAs) and uni-2QPDAs (resp., uni-1QPDAs or uni-RT-QPDAs) are the same or not.

3.4. Quantum Counter Automata

Since quantum CAs are a special case of quantum PDAs, we do not give the details for all kind of quantum CAs. Instead, we focus on the realtime counterpart of quantum counter automata.

A realtime quantum k-counter automaton (RT-QkCA) is a 6-tuple

$$\mathcal{M} = (Q, \Sigma, \Omega, \delta, q_1, Q_a). \tag{3.48}$$

We give the local conditions of well-formedness for RT-Q1CAs: For any choice of $q_1, q_2 \in Q$, $\sigma \in \tilde{\Sigma}$, and $\theta_1, \theta_2 \in \Theta$,

$$\sum_{q' \in Q, c \in \Diamond, \omega \in \Omega} \overline{\delta(q_1, \sigma, \theta_1, q', c, \omega)} \delta(q_2, \sigma, \theta_2, q', c, \omega) = \begin{cases} 1 & q_1 = q_2 \text{ and } \theta_1 = \theta_2 \\ 0 & \text{otherwise} \end{cases}, \tag{3.49}$$

$$\sum_{q' \in Q, \omega \in \Omega} \begin{array}{l} \overline{\delta(q_1, \sigma, \theta_1, q', +1, \omega)} \delta(q_2, \sigma, \theta_2, q', 0, \omega) \\ + \overline{\delta(q_1, \sigma, \theta_1, q', 0, \omega)} \delta(q_2, \sigma, \theta_2, q', -1, \omega) = 0 \end{array}, \tag{3.50}$$

and

$$\sum_{q' \in Q, \omega \in \Omega} \overline{\delta(q_1, \sigma, \theta_1, q', +1, \omega)} \delta(q_2, \sigma, \theta_2, q', -1, \omega) = 0. \tag{3.51}$$

If the RT-Q1CA (and RT-QkCA) is unidirectional, then there is only one local

condition for well-formedness: for any choice of $q_1, q_2 \in Q$, $\sigma \in \tilde{\Sigma}$, and $\bar{\theta_1}, \bar{\theta_2} \in \Theta^k$,

$$\sum_{q' \in Q, \omega \in \Omega} \overline{\delta(q_1, \sigma, \bar{\theta_1}, q', \omega)} \delta(q_2, \sigma, \bar{\theta_2}, q', \omega) = \begin{cases} 1 & q_1 = q_2 \text{ and } \bar{\theta_1} = \bar{\theta_2} \\ 0 & \text{otherwise} \end{cases} . \tag{3.52}$$

In the unidirectional case, we can define an admissible operator $\mathcal{E}_{\sigma,\bar{\theta}}$ for each $\sigma \in \tilde{\Sigma}$ and $\bar{\theta} \in \Theta^k$, which is described by a collection $\{E_{\sigma,\bar{\theta},\omega}\}$, where $\omega \in \Omega$ and $E_{\sigma,\bar{\theta},\omega}[j,i]$ represents the amplitude of the transition from state q_i to q_j when reading symbol σ, having counter signs $\bar{\theta}$, and writing ω on the finite register (and so the value of the counter is updated by $D_c(q')$). Note that, $\mathcal{E}_{\sigma,\bar{\theta}}$ is admissible if and only if

$$\sum_{\omega \in \Omega} E^{\dagger}_{\sigma,\bar{\theta},\omega} E_{\sigma,\bar{\theta},\omega} = I. \tag{3.53}$$

It is an open problem whether the computation powers of RT-QkCA and uni-RT-QkCA are the same or not.

We close the section by showing the isomorphism of RT-QkCA(m) and RT-QkCA.

Lemma 3.4. *Any RT-QkCA(m) can be exactly simulated by a RT-QkCA.*

Proof. Let $\mathcal{M} = (Q, \Sigma, \Omega, \delta, q_1, Q_a)$ be the RT-QkCA(m) and $\mathcal{M}' = (Q', \Sigma, \Omega, \delta', q_1', Q_a')$ be the RT-QkCA simulating $\mathcal{M}$ exactly. For each internal state of $\mathcal{M}$, say $q \in Q$, we define m^k internal states, i.e. $\langle q, i_1, \ldots, i_k \rangle \in Q'$ $(0 \leq i_j \leq m-1, 1 \leq j \leq k)$, for $\mathcal{M}'$. Moreover, $q_1' = \langle q_1, 0, \ldots, 0 \rangle$ and $Q_a' = Q_a \times \{0, \ldots, m-1\}^k$.

Let

$$\varphi : \mathbb{Z}^k \to \mathbb{Z}^k \times \{0, \ldots, m-1\}^k \tag{3.54}$$

be a bijection such that

$$\varphi(x_1, \ldots, x_k) = \left(\left\lfloor \frac{x_1}{m} \right\rfloor, \ldots, \left\lfloor \frac{x_k}{m} \right\rfloor, (x_1 \mod m), \ldots, (x_k \mod m) \right). \tag{3.55}$$

Hence, we can say that the counter values of $\mathcal{M}$, say $\bar{x} \in \mathbb{Z}^k$, can be equivalently represented by $\varphi(\bar{x})$, based on which we construct $\mathcal{M}'$, where $\varphi(\bar{x})[i]$ is stored by the i^{th} counter and $\varphi(\bar{x})[k+i]$ is stored by the internal state. That is, for any configuration of $\mathcal{M}$, say $(q, \bar{x})$, we have an equivalent configuration of $\mathcal{M}'$ as

$$\left(\langle q, \varphi(\bar{x})[k+1], \ldots, \varphi(\bar{x})[2k]\rangle, \varphi(\bar{x})[1], \ldots, \varphi(\bar{x})[k]\right). \tag{3.56}$$

Moreover, the transitions of $\mathcal{M}'$ can be obtained from those of $\mathcal{M}$ in the following way: for any $(i_1, \ldots, i_k) \in \{-m, \ldots, m\}^k$ and $(j_1, \ldots, j_k) \in \{0, \ldots, m-1\}^k$, the part of the transition

$$(q, \sigma) \xrightarrow{\delta} \alpha(q', i_1, \ldots, i_k, \omega) \tag{3.57}$$

of $\mathcal{M}$ is replaced by transition

$$\begin{array}{l} \left(\langle q, j_1, \ldots, j_k\rangle, \sigma\right) \xrightarrow{\delta'} \\ \quad \alpha\left(\langle q', (j_1 + i_1 \mod m), \ldots, (j_k + i_k \mod m)\rangle, \left\lfloor \frac{j_1+i_1}{m} \right\rfloor, \ldots, \left\lfloor \frac{j_k+i_k}{m} \right\rfloor, \omega\right) \end{array} \tag{3.58}$$

in $\mathcal{M}'$, where $q \in Q$, $\sigma \in \tilde{\Sigma}$, $\omega \in \Omega$, and $\alpha \in \mathbb{C}$ is the amplitude of the transition. Since φ is a bijection, the configuration matrix of $\mathcal{M}$ is isomorphic to the one of $\mathcal{M}'$ for any input string $w \in \Sigma^*$. (Similarly, $\mathcal{M}$ is well-formed if and only if $\mathcal{M}'$ is well-formed.) Therefore, they process exactly the same computation on a given input string, say $w \in \Sigma^*$, and so

$$f_{\mathcal{M}}(w) = f_{\mathcal{M}'}(w). \tag{3.59}$$

□

3.5. Nondeterministic Quantum Machines

A nondeterministic quantum machine is a quantum machine with the error setting of positive one-sided unbounded error. "N" is used before "Q" in the abbreviations of quantum machines.

4. SUBLOGARITHMIC-SPACE UNBOUNDED-ERROR COMPUTATION

In this chapter, we focus on unbounded error quantum computation in sublogatihmic space (Section 4.1) and nondeterminitic quantum computation in constant space (Section 4.2).

4.1. Unbounded Error Results

Watrous compared the unbounded-error probabilistic space complexity classes ($\mathrm{PrSPACE}_{\mathbb{Q}}(s)$ and $\mathrm{PrSPACE}_{\mathbb{A}}(s)$) with the corresponding classes for both Wa98-QTMs [6, 7] and Wa03-QTMs [8], respectively, for space bounds $s = \Omega(\log n)$, establishing the identity of the associated quantum space complexity classes with each other, and also with the corresponding probabilistic ones. The case of $s = o(\log n)$ was left as an open question [7]. In this section, we provide an answer to that question.

4.1.1. Probabilistic versus Quantum Computation with Sublogarithmic Space

We already know that QTMs allowing superoperators are at least as powerful as PTMs for any common space bound. We now exhibit a 1KWQFA which performs a task that is impossible for PTMs with small space bounds.

Consider the nonstochastic and context-free language [40]

$$L_{NH} = \{a^x b a^{y_1} b a^{y_2} b \cdots a^{y_t} b \mid x, t, y_1, \cdots, y_t \in \mathbb{Z}^+ \text{ and } \exists k\ (1 \leq k \leq t), x = \sum_{i=1}^{k} y_i\} \qquad (4.1)$$

over the alphabet $\Sigma = \{a, b\}$. Freivalds and Karpinski [35] have proven the following facts about L_{NH}:

Fact 4.1. *No PTM using space $o(\log\log n)$ can recognize L_{NH} with unbounded error.*

Fact 4.2. *No 1PTM using space $o(\log n)$ can recognize L_{NH} with unbounded error.*

There exists a one-way *deterministic* TM that recognizes L_{NH} within the optimal space bound $O(\log n)$ [35]. No (two-way) PTM which recognizes L_{NH} using $o(\log n)$ space is known as of the time of writing.

Theorem 4.3. *There exists a 1KWQFA that recognizes L_{NH} with unbounded error.*

Proof. Consider the 1KWQFA $\mathcal{M} = (Q, \Sigma, \delta, q_0, Q_a, Q_r)$, where $\Sigma = \{a, b\}$ and the state sets are as follows:

$$\begin{aligned} Q_n &= \{\overrightarrow{q_0}\} \cup \{\overrightarrow{q_i} \mid 1 \leq i \leq 6\} \cup \{\overrightarrow{p_i} \mid 1 \leq i \leq 6\} \cup \{\overrightarrow{a_i} \mid 1 \leq i \leq 4\} \\ &\quad \cup \{\overrightarrow{r_i} \mid 1 \leq i \leq 4\} \cup \{\Downarrow w_i \mid 1 \leq i \leq 6\}, \\ Q_a &= \{\Downarrow A_i \mid 1 \leq i \leq 18\},\ Q_r = \{\Downarrow R_i \mid 1 \leq i \leq 18\}. \end{aligned} \tag{4.2}$$

Let each U_σ induced by δ act as indicated in Figures 4.1 and 4.2, and extend each to be unitary.

Machine $\mathcal{M}$ starts computation on symbol ¢ by branching into two paths, path_1 and path_2, with equal amplitude. Each path and their subpaths, to be described later, check whether the input is of the form $(aa^*b)(aa^*b)(aa^*b)^*$. The different stages of the program indicated in Figures 4.1 and 4.2 correspond to the subtasks of this regular expression check. Stage I ends successfully if the input begins with (aa^*b). Stage II checks the second (aa^*b). Finally, Stage III controls whether the input ends with $(aa^*b)^*$.

The reader note that many transitions in the machine are of the form

$$U_\sigma|q_i\rangle = |\psi\rangle + \alpha|A_k\rangle + \alpha|R_k\rangle, \tag{4.3}$$

where $|\psi\rangle$ is a superposition of configurations such that $\langle\psi|\psi\rangle = 1 - 2\alpha^2$, $A_k \in Q_a$, $R_k \in Q_r$. The equal-probability transitions to the "twin halting states" A_k and R_k are included to ensure that the matrices are unitary, without upsetting the "accept/reject balance" until a final decision about the membership of the input in L_{NH} is reached. If the regular expression check mentioned above fails, each path in question splits

Stages	$U_{¢}, U_a$	$U_{\$}$
	$U_{¢}\|\overrightarrow{q_0}\rangle = \frac{1}{\sqrt{2}}\|\overrightarrow{q_1}\rangle + \frac{1}{\sqrt{2}}\|\overrightarrow{p_1}\rangle$	
I (path_1)	$U_a\|\overrightarrow{q_1}\rangle = \frac{1}{\sqrt{2}}\|\overrightarrow{q_2}\rangle + \frac{1}{2}\|\downarrow A_1\rangle + \frac{1}{2}\|\downarrow R_1\rangle$ $U_a\|\overrightarrow{q_2}\rangle = \frac{1}{\sqrt{2}}\|\overrightarrow{q_2}\rangle - \frac{1}{2}\|\downarrow A_1\rangle - \frac{1}{2}\|\downarrow R_1\rangle$	$U_{\$}\|\overrightarrow{q_1}\rangle = \frac{1}{\sqrt{2}}\|\downarrow A_1\rangle + \frac{1}{\sqrt{2}}\|\downarrow R_1\rangle$ $U_{\$}\|\overrightarrow{q_2}\rangle = \frac{1}{\sqrt{2}}\|\downarrow A_2\rangle + \frac{1}{\sqrt{2}}\|\downarrow R_2\rangle$ $U_{\$}\|\overrightarrow{q_3}\rangle = \frac{1}{\sqrt{2}}\|\downarrow A_3\rangle + \frac{1}{\sqrt{2}}\|\downarrow R_3\rangle$
I (path_2)	$U_a\|\overrightarrow{p_1}\rangle = \|\downarrow w_1\rangle$ $U_a\|\downarrow w_1\rangle = \frac{1}{\sqrt{2}}\|\overrightarrow{p_2}\rangle + \frac{1}{2}\|\downarrow A_2\rangle + \frac{1}{2}\|\downarrow R_2\rangle$ $U_a\|\overrightarrow{p_2}\rangle = \|\downarrow w_2\rangle$ $U_a\|\downarrow w_2\rangle = \frac{1}{\sqrt{2}}\|\overrightarrow{p_2}\rangle - \frac{1}{2}\|\downarrow A_2\rangle - \frac{1}{2}\|\downarrow R_2\rangle$	$U_{\$}\|\overrightarrow{p_1}\rangle = \frac{1}{\sqrt{2}}\|\downarrow A_4\rangle + \frac{1}{\sqrt{2}}\|\downarrow R_4\rangle$ $U_{\$}\|\overrightarrow{p_2}\rangle = \frac{1}{\sqrt{2}}\|\downarrow A_5\rangle + \frac{1}{\sqrt{2}}\|\downarrow R_5\rangle$ $U_{\$}\|\overrightarrow{p_3}\rangle = \frac{1}{\sqrt{2}}\|\downarrow A_6\rangle + \frac{1}{\sqrt{2}}\|\downarrow R_6\rangle$
II (path_1)	$U_a\|\overrightarrow{q_3}\rangle = \|\downarrow w_3\rangle$ $U_a\|\downarrow w_3\rangle = \frac{1}{\sqrt{2}}\|\overrightarrow{q_4}\rangle + \frac{1}{2}\|\downarrow A_3\rangle + \frac{1}{2}\|\downarrow R_3\rangle$ $U_a\|\overrightarrow{q_4}\rangle = \|\downarrow w_4\rangle$ $U_a\|\downarrow w_4\rangle = \frac{1}{\sqrt{2}}\|\overrightarrow{q_4}\rangle - \frac{1}{2}\|\downarrow A_3\rangle - \frac{1}{2}\|\downarrow R_3\rangle$	$U_{\$}\|\overrightarrow{q_4}\rangle = \frac{1}{\sqrt{2}}\|\downarrow A_7\rangle + \frac{1}{\sqrt{2}}\|\downarrow R_7\rangle$ $U_{\$}\|\overrightarrow{q_5}\rangle = \frac{1}{\sqrt{2}}\|\downarrow A_8\rangle + \frac{1}{\sqrt{2}}\|\downarrow R_8\rangle$
II (path_2)	$U_a\|\overrightarrow{p_3}\rangle = \frac{1}{\sqrt{2}}\|\overrightarrow{p_4}\rangle + \frac{1}{2}\|\downarrow A_4\rangle + \frac{1}{2}\|\downarrow R_4\rangle$ $U_a\|\overrightarrow{p_4}\rangle = \frac{1}{\sqrt{2}}\|\overrightarrow{p_4}\rangle - \frac{1}{2}\|\downarrow A_4\rangle - \frac{1}{2}\|\downarrow R_4\rangle$	$U_{\$}\|\overrightarrow{p_4}\rangle = \frac{1}{\sqrt{2}}\|\downarrow A_9\rangle + \frac{1}{\sqrt{2}}\|\downarrow R_9\rangle$ $U_{\$}\|\overrightarrow{p_5}\rangle = \frac{1}{\sqrt{2}}\|\downarrow A_{10}\rangle + \frac{1}{\sqrt{2}}\|\downarrow R_{10}\rangle$
III (path_1)	$U_a\|\overrightarrow{q_5}\rangle = \|\downarrow w_5\rangle$ $U_a\|\downarrow w_5\rangle = \frac{1}{\sqrt{2}}\|\overrightarrow{q_6}\rangle + \frac{1}{2}\|\downarrow A_5\rangle + \frac{1}{2}\|\downarrow R_5\rangle$ $U_a\|\overrightarrow{q_6}\rangle = \|\downarrow w_6\rangle$ $U_a\|\downarrow w_6\rangle = \frac{1}{\sqrt{2}}\|\overrightarrow{q_6}\rangle - \frac{1}{2}\|\downarrow A_5\rangle - \frac{1}{2}\|\downarrow R_5\rangle$	$U_{\$}\|\overrightarrow{q_6}\rangle = \frac{1}{\sqrt{2}}\|\downarrow A_{11}\rangle + \frac{1}{\sqrt{2}}\|\downarrow R_{11}\rangle$
III (path_2)	$U_a\|\overrightarrow{p_5}\rangle = \frac{1}{\sqrt{2}}\|\overrightarrow{p_6}\rangle + \frac{1}{2}\|\downarrow A_6\rangle + \frac{1}{2}\|\downarrow R_6\rangle$ $U_a\|\overrightarrow{p_6}\rangle = \frac{1}{\sqrt{2}}\|\overrightarrow{p_6}\rangle - \frac{1}{2}\|\downarrow A_6\rangle - \frac{1}{2}\|\downarrow R_6\rangle$	$U_{\$}\|\overrightarrow{p_6}\rangle = \frac{1}{\sqrt{2}}\|\downarrow A_{12}\rangle + \frac{1}{\sqrt{2}}\|\downarrow R_{12}\rangle$
III ($\mathsf{path}_{\mathsf{accept}}$)	$U_a\|\overrightarrow{a_1}\rangle = \frac{1}{\sqrt{2}}\|\overrightarrow{a_2}\rangle + \frac{1}{2}\|\downarrow A_7\rangle + \frac{1}{2}\|\downarrow R_7\rangle$ $U_a\|\overrightarrow{a_2}\rangle = \frac{1}{\sqrt{2}}\|\overrightarrow{a_2}\rangle - \frac{1}{2}\|\downarrow A_7\rangle - \frac{1}{2}\|\downarrow R_7\rangle$ $U_a\|\overrightarrow{a_3}\rangle = \frac{1}{\sqrt{2}}\|\overrightarrow{a_4}\rangle + \frac{1}{2}\|\downarrow A_8\rangle + \frac{1}{2}\|\downarrow R_8\rangle$ $U_a\|\overrightarrow{a_4}\rangle = \frac{1}{\sqrt{2}}\|\overrightarrow{a_4}\rangle - \frac{1}{2}\|\downarrow A_8\rangle - \frac{1}{2}\|\downarrow R_8\rangle$	$U_{\$}\|\overrightarrow{a_1}\rangle = \|\downarrow A_{17}\rangle$ $U_{\$}\|\overrightarrow{a_3}\rangle = \|\downarrow A_{18}\rangle$ $U_{\$}\|\overrightarrow{a_2}\rangle = \frac{1}{\sqrt{2}}\|\downarrow A_{13}\rangle + \frac{1}{\sqrt{2}}\|\downarrow R_{13}\rangle$ $U_{\$}\|\overrightarrow{a_4}\rangle = \frac{1}{\sqrt{2}}\|\downarrow A_{14}\rangle + \frac{1}{\sqrt{2}}\|\downarrow R_{14}\rangle$
III ($\mathsf{path}_{\mathsf{reject}}$)	$U_a\|\overrightarrow{r_1}\rangle = \frac{1}{\sqrt{2}}\|\overrightarrow{r_2}\rangle + \frac{1}{2}\|\downarrow A_9\rangle + \frac{1}{2}\|\downarrow R_9\rangle$ $U_a\|\overrightarrow{r_2}\rangle = \frac{1}{\sqrt{2}}\|\overrightarrow{r_2}\rangle - \frac{1}{2}\|\downarrow A_9\rangle - \frac{1}{2}\|\downarrow R_9\rangle$ $U_a\|\overrightarrow{r_3}\rangle = \frac{1}{\sqrt{2}}\|\overrightarrow{r_4}\rangle + \frac{1}{2}\|\downarrow A_{10}\rangle + \frac{1}{2}\|\downarrow R_{10}\rangle$ $U_a\|\overrightarrow{r_4}\rangle = \frac{1}{\sqrt{2}}\|\overrightarrow{r_4}\rangle - \frac{1}{2}\|\downarrow A_{10}\rangle - \frac{1}{2}\|\downarrow R_{10}\rangle$	$U_{\$}\|\overrightarrow{r_1}\rangle = \|\downarrow R_{17}\rangle$ $U_{\$}\|\overrightarrow{r_3}\rangle = \|\downarrow R_{18}\rangle$ $U_{\$}\|\overrightarrow{r_2}\rangle = \frac{1}{\sqrt{2}}\|\downarrow A_{15}\rangle + \frac{1}{\sqrt{2}}\|\downarrow R_{15}\rangle$ $U_{\$}\|\overrightarrow{r_4}\rangle = \frac{1}{\sqrt{2}}\|\downarrow A_{16}\rangle + \frac{1}{\sqrt{2}}\|\downarrow R_{16}\rangle$

Figure 4.1. Specification of the transition function of the 1KWQFA presented in the proof of Theorem 4.3 (I)

equiprobably to one rejecting and one accepting configuration, and the overall probability of acceptance of the machine turns out to be precisely $\frac{1}{2}$. If the input is indeed of the form $(aa^*b)(aa^*b)(aa^*b)^*$, whether the acceptance probability exceeds $\frac{1}{2}$ or not

Stages	U_b
I (path_1)	$U_b\|\overrightarrow{q_1}\rangle = \frac{1}{\sqrt{2}}\|\downarrow A_1\rangle + \frac{1}{\sqrt{2}}\|\downarrow R_1\rangle$ $U_b\|\overrightarrow{q_2}\rangle = \|\overrightarrow{q_3}\rangle$ $U_b\|\overrightarrow{q_3}\rangle = \frac{1}{\sqrt{2}}\|\downarrow A_2\rangle + \frac{1}{\sqrt{2}}\|\downarrow R_2\rangle$
I (path_2)	$U_b\|\overrightarrow{p_1}\rangle = \frac{1}{\sqrt{2}}\|\downarrow A_3\rangle + \frac{1}{\sqrt{2}}\|\downarrow R_3\rangle$ $U_b\|\overrightarrow{p_2}\rangle = \|\overrightarrow{p_3}\rangle$ $U_b\|\overrightarrow{p_3}\rangle = \frac{1}{\sqrt{2}}\|\downarrow A_4\rangle + \frac{1}{\sqrt{2}}\|\downarrow R_4\rangle$
II (path_1)	$U_b\|\overrightarrow{q_4}\rangle = \frac{1}{2}\|\overrightarrow{q_5}\rangle + \frac{1}{2\sqrt{2}}\|\overrightarrow{a_1}\rangle + \frac{1}{2\sqrt{2}}\|\overrightarrow{r_1}\rangle + \frac{1}{2}\|\downarrow A_{11}\rangle + \frac{1}{2}\|\downarrow R_{11}\rangle$ $U_b\|\overrightarrow{q_5}\rangle = \frac{1}{\sqrt{2}}\|\downarrow A_5\rangle + \frac{1}{\sqrt{2}}\|\downarrow R_5\rangle$
II (path_2)	$U_b\|\overrightarrow{p_4}\rangle = \frac{1}{2}\|\overrightarrow{p_5}\rangle + \frac{1}{2\sqrt{2}}\|\overrightarrow{a_1}\rangle - \frac{1}{2\sqrt{2}}\|\overrightarrow{r_1}\rangle + \frac{1}{2}\|\downarrow A_{12}\rangle + \frac{1}{2}\|\downarrow R_{12}\rangle$ $U_b\|\overrightarrow{p_5}\rangle = \frac{1}{\sqrt{2}}\|\downarrow A_6\rangle + \frac{1}{\sqrt{2}}\|\downarrow R_6\rangle$
III (path_1)	$U_b\|\overrightarrow{q_6}\rangle = \frac{1}{2}\|\overrightarrow{q_5}\rangle + \frac{1}{2\sqrt{2}}\|\overrightarrow{a_1}\rangle + \frac{1}{2\sqrt{2}}\|\overrightarrow{r_1}\rangle - \frac{1}{2}\|\downarrow A_{11}\rangle - \frac{1}{2}\|\downarrow R_{11}\rangle$
III (path_2)	$U_b\|\overrightarrow{p_6}\rangle = \frac{1}{2}\|\overrightarrow{p_5}\rangle + \frac{1}{2\sqrt{2}}\|\overrightarrow{a_1}\rangle - \frac{1}{2\sqrt{2}}\|\overrightarrow{r_1}\rangle - \frac{1}{2}\|\downarrow A_{12}\rangle - \frac{1}{2}\|\downarrow R_{12}\rangle$
III ($\mathsf{path}_{\mathsf{accept}}$)	$U_b\|\overrightarrow{a_2}\rangle = \frac{1}{\sqrt{2}}\|\overrightarrow{a_3}\rangle + \frac{1}{2}\|\downarrow A_{13}\rangle + \frac{1}{2}\|\downarrow R_{13}\rangle$ $U_b\|\overrightarrow{a_1}\rangle = \frac{1}{\sqrt{2}}\|\downarrow A_7\rangle + \frac{1}{\sqrt{2}}\|\downarrow R_7\rangle$ $U_b\|\overrightarrow{a_4}\rangle = \frac{1}{\sqrt{2}}\|\overrightarrow{a_3}\rangle - \frac{1}{2}\|\downarrow A_{13}\rangle - \frac{1}{2}\|\downarrow R_{13}\rangle$ $U_b\|\overrightarrow{a_3}\rangle = \frac{1}{\sqrt{2}}\|\downarrow A_8\rangle + \frac{1}{\sqrt{2}}\|\downarrow R_8\rangle$
III ($\mathsf{path}_{\mathsf{reject}}$)	$U_b\|\overrightarrow{r_2}\rangle = \frac{1}{\sqrt{2}}\|\overrightarrow{r_3}\rangle + \frac{1}{2}\|\downarrow A_{14}\rangle + \frac{1}{2}\|\downarrow R_{14}\rangle$ $U_b\|\overrightarrow{r_1}\rangle = \frac{1}{\sqrt{2}}\|\downarrow A_9\rangle + \frac{1}{\sqrt{2}}\|\downarrow R_9\rangle$ $U_b\|\overrightarrow{r_4}\rangle = \frac{1}{\sqrt{2}}\|\overrightarrow{r_3}\rangle - \frac{1}{2}\|\downarrow A_{14}\rangle - \frac{1}{2}\|\downarrow R_{14}\rangle$ $U_b\|\overrightarrow{r_3}\rangle = \frac{1}{\sqrt{2}}\|\downarrow A_{10}\rangle + \frac{1}{\sqrt{2}}\|\downarrow R_{10}\rangle$

Figure 4.2. Specification of the transition function of the 1KWQFA presented in the proof of Theorem 4.3 (II)

depends on the following additional tasks performed by the computation paths in order to test for the equality mentioned in the definition of L_{NH}:

i. path_1 walks over the a's at the speed of one tape square per step until reading the first b. After that point, path_1 pauses for one step over each a before moving on to the next symbol.
ii. path_2 pauses for one step over each a until reading the first b. After that point, path_2 walks over each a at the speed of one square per step.

iii. On each b except the first one, path_1 and path_2 split to take the following two courses of action with equal probability:

- In the first alternative, path_1 and path_2 perform a 2-way quantum Fourier transform (QFT) [4]:
  (i) The targets of the QFT are two new computational paths, i.e., $\mathsf{path}_{\mathsf{accept}}$ and $\mathsf{path}_{\mathsf{reject}}$. Disregarding the equal-probability transitions to the twin halting states mentioned above, the QFT is realized as:

$$\mathsf{path}_1 \rightarrow \frac{1}{\sqrt{2}}\mathsf{path}_{\mathsf{accept}} + \frac{1}{\sqrt{2}}\mathsf{path}_{\mathsf{reject}} \tag{4.4}$$

$$\mathsf{path}_2 \rightarrow \frac{1}{\sqrt{2}}\mathsf{path}_{\mathsf{accept}} - \frac{1}{\sqrt{2}}\mathsf{path}_{\mathsf{reject}} \tag{4.5}$$

  (ii) $\mathsf{path}_{\mathsf{accept}}$ and $\mathsf{path}_{\mathsf{reject}}$ continue computation at the speed of path_2, walking over the b's without performing the QFT any more.
- In the second alternative, path_1 and path_2 continue computation without performing the QFT.

iv. On symbol \$, $\mathsf{path}_{\mathsf{accept}}$ enters an accepting state, $\mathsf{path}_{\mathsf{reject}}$ enters a rejecting state, path_1 and path_2 enter accepting and rejecting states with equal probability.

Suppose that the input is of the form

$$w = a^x b a^{y_1} b a^{y_2} b \cdots a^{y_t} b, \tag{4.6}$$

where $x, t, y_1, \cdots, y_t \in \mathbb{Z}^+$.

path_1 reaches the first b earlier than path_2. Once it has passed the first b, path_2 becomes faster, and may or may not catch up with path_1, depending on the number of a's in the input after the first b. The two paths can meet on the symbol following the x'th a after the first b, since at that point path_1 has paused for the same number of steps as path_2. Only if that symbol is a b, the two paths perform a QFT in the same

place and at the same time. To paraphrase, if there exists a k $(1 \leq k \leq t)$ such that $x = \sum_{i=1}^{k} y_i$, path_1 and path_2 meet over the $(k+1)^{th}$ b and perform the QFT at the same step. If there is no such k, the paths either never meet, or meet over an a without a QFT.

The $\mathsf{path}_{\mathsf{accept}}$ and $\mathsf{path}_{\mathsf{reject}}$s that are offshoots of path_1 continue their traversal of the string faster than path_1. On the other hand, the offshoots of path_2 continue their traversal at the same speed as path_2.

By definition, the twin halting states reached during the computation contribute equal amounts to the acceptance and rejection probabilities. path_1 and path_2 accept and reject equiprobably when they reach the end of the string. If path_1 and path_2 never perform the QFT at the same time and in the same position, every QFT produces two equal-probability paths which perform identical tasks, except that one accepts and the other one rejects at the end.

The overall acceptance and rejection probabilities are equal, $\frac{1}{2}$, unless a $\mathsf{path}_{\mathsf{reject}}$ with positive amplitude and a $\mathsf{path}_{\mathsf{reject}}$ with negative amplitude can meet and therefore cancel each other. In such a case, the surviving $\mathsf{path}_{\mathsf{accept}}$'s contributes the additional acceptance probability that tips the balance. As described above, such a cancellation is only possible when path_1 and path_2 perform the QFT together.

Therefore, if $w \in L_{NH}$, the overall acceptance probability is greater than $\frac{1}{2}$. If $w \notin L_{NH}$, the overall acceptance probability equals $\frac{1}{2}$. □

Corollary 4.4. *For any space bound s satisfying $s(n) = o(\log \log n)$,*

$$PrSPACE(s) \subsetneq PrQSPACE(s). \tag{4.7}$$

Open Problem 2. Is PrSPACE$(s) \subsetneq$ PrQSPACE(s), for $s \in \Omega(\log \log n) \cap o(\log n)$?

Corollary 4.5. *For any space bound s satisfying $s(n) = o(\log n)$,*

$$\textit{one-way-PrSPACE}(s) \subsetneq \textit{one-way-PrQSPACE}(s). \tag{4.8}$$

Corollary 4.6. $\textit{coC}_{=}\textit{SPACE}(1) \subsetneq \textit{coC}_{=}\textit{QSPACE}(1)$.

Proof. Since coC$_=$SPACE(1) is a proper subset of S [29], L_{NH} is not a member of coC$_=$SPACE(1). On the other hand, as shown in Theorem 4.3, L_{NH} is also a member of one-way-coC$_=$QSPACE(1). □

In the next section, we prove a fact which allows us to state a similar inclusion relationship between the classes of languages recognized by QTMs with restricted measurements and PTMs using constant space.

As noted before, Watrous proved the equality PrQSPACE(s)=PrSPACE(s) ($s \in \Omega(\log n)$) for the cases where PrQSPACE is defined in terms of Wa98-QTMs [6, 7], and Wa03-QTMs [8]. However, we do not know how to prove these results for our more general QTMs.

Open Problem 3. Is PrQSPACE(s) $\subseteq$ PrSPACE(s), for $s \in \Omega(\log n)$?

We continue with presenting some other nonstochastic languages[12] in one-way-PrQSPACE(1) by extending the 1KWQFA algorithm for L_{NH} as follows:

- On symbol ¢, the computation splits into two paths, path_1 and path_2, with equal amplitude. path_1 (resp., or path_2) immediately moves to the right and path_2 (resp., or path_1) stays c steps on ¢ before moving to the right.
- While scanning the input, path_1 (resp., path_2) stays on each symbol c_1 (resp., c_2) step(s); and, just before moving to the right, it produces a subpath, say $\mathsf{subpath}_1$ (resp., $\mathsf{subpath}_2$), with some amplitude when reading $\sigma_1 \in \Sigma$ (resp., $\sigma_2 \in \Sigma$).

[12]Note that, their complementary languages are also nonstochastic.

- $\mathsf{subpath}_1$ (resp., $\mathsf{subpath}_2$) travels the remaining part of the input by staying c_1' (resp., c_2') step(s) on each symbol.
- When path_1 and path_2 read \$, the input is equiprobably accepted and rejected by each of them.
- The subpaths, on the other hand, make the following 2-QFT

$$\begin{array}{rcl} \mathsf{subpath}_1 : |S_1\rangle & \rightarrow & \frac{1}{\sqrt{2}}|A\rangle + \frac{1}{\sqrt{2}}|R\rangle \\ \mathsf{subpath}_2 : |S_2\rangle & \rightarrow & \frac{1}{\sqrt{2}}|A\rangle - \frac{1}{\sqrt{2}}|R\rangle \end{array}, \tag{4.9}$$

  where $|S_1\rangle$ (resp., $|S_2\rangle$) is the configuration that $\mathsf{subpath}_1$ (resp., $\mathsf{subpath}_2$) is in before reading \$; $|A\rangle$ (resp., $|R\rangle$) is a specified accepting (resp., rejecting) configuration.

After reading \$, the overall accepting probability exceeds $\frac{1}{2}$ only if any pair of $\mathsf{subpath}_1$ and $\mathsf{subpath}_2$ reach to \$ at the same time. Otherwise, the overall accepting and rejecting probabilities become equal.

Let $L \subseteq \Sigma$ be the language recognized by the above 1KWQFA algorithm with one-sided cutpoint $\frac{1}{2}$. One can easily verifies that $w \in \Sigma^*$ is a member of L if and only if there are two indexes, say i and j $(1 \leq i, j \leq |w|)$, satisfying $w_i = \sigma_1$ and $w_j = \sigma_2$ and

$$c_1 i + c_1'(|w| - i) = c_2 j + c_2'(|w| - j) + c', \tag{4.10}$$

where $c' = -c$ if path_1 reads ¢ at least twice and $c' = c$ otherwise. By simplifying the equation, we obtain

$$(c_1 - c_1')i = (c_2 - c_2')j + (c_2' - c_1')|w| + c', \tag{4.11}$$

or

$$d_1 i = d_2 j + d_3 |w| + d, \tag{4.12}$$

where $d_1 = c_1 - c_1'$, $d_2 = c_2 - c_2'$, $d_3 = c_2' - c_1'$, and $d = c'$. Note that, c_1, c_1', c_2, and c_2' are nonnegative integers, but d_1, d_2, and d_3 can take negative values. By setting d, d_1', d_2', d_2, d_1, σ_1, and σ_2 appropriately, many different languages in one-way-PrQSPACE(1) are obtained. The languages given in Figure 4.3 are examples of such languages. (Except the special ones, they are nonstochatic). Another (nonstochastic) example is $L_{center} = \{w \in \{a,b\}^* \mid |w| = 2k-1, w_k = b, k > 0\}$ due to the fact that it can be characterized by the equation $2i = |w| + 1$ and $\sigma_1 = \sigma_2 = b$.

In [41], a joint work with R. Freivalds, we presented a new family of languages:

$$L_{d_1,\sigma_1,d_2,\sigma_2,d,\Sigma} = \{w \in \Sigma^* \mid \exists i,j\,(w_i = \sigma_1, w_j = \sigma_2, d_1 i = d_2(|w| + 1 - j) + d)\}, \quad (4.13)$$

where $d, d_1, d_2 \in \mathbb{Z}$, $\sigma_1, \sigma_2 \in \Sigma$. We can classify those languages as follows:

- They are in REG, if
  - i. Σ is unary,
  - ii. d_1 and d_2 have different signs, or
  - iii. $\gcd(d_1, d_2)$ does not divide d.
- They are in S, if they are members of $\{L_{1,a,1,b,d,\{a,b\}} \mid d \in \mathbb{Z}\}$.
- They are not in uS, otherwise.

One of the simplest languages that are not in uS is $L_{1,b,1,b,0,\{a,b\}}$, that is,

$$L_{say} = \{w \mid \exists u_1, u_2, v_1, v_2 \in \{a,b\}^*, w = u_1 b u_2 = v_1 b v_2, |u_1| = |v_2|\}, \quad (4.14)$$

where the name comes from the surname of A. C. Cem Say, who considered this language for the first time.

Figure 4.3. A new family of nonstochastic languages

Theorem 4.7. *The nonstochastic language*[13]

$$L_{div} = \{a^n b a^{kn} \mid k, n \in \mathbb{Z}^+\} \quad (4.15)$$

can be recognized by a 2QFA with unbounded error.

[13]See page 88 on [42].

Proof. We give a sketch of the algorithm:

i. If the input string is not of the form a^+ba^+, it is accepted with probability $\frac{1}{2}$.

ii. Otherwise, the computation splits into two paths, say path_1 and path_2, on the symbol b.

iii. In an infinite loop, path_1 travels to the left end-marker and comes back to the b with the speed of one symbol per step. On the b, it

- performs the following transition with probability $\frac{1}{2}$

$$\mathsf{path}_1 \rightarrow \frac{1}{\sqrt{2}}(Accept) + \frac{1}{\sqrt{2}}(Reject), \tag{4.16}$$

- and goes on with probability $\frac{1}{2}$.

iv. path_2 travels to the right end-marker and comes back to the b with the speed of one symbol per step, on which it performs the following transition exactly:

$$\mathsf{path}_2 \rightarrow \frac{1}{\sqrt{2}}(Accept) - \frac{1}{\sqrt{2}}(Reject). \tag{4.17}$$

The two paths meet only if the input string is a member of L_{div}, in which case the acceptance probability would exceed $\frac{1}{2}$ due to interference. Otherwise, the acceptance probability becomes equal to $\frac{1}{2}$. □

Open Problem 4. Is L_{div} in one-way-PrQSPACE(1)?

Open Problem 5. Is one-way-PrQSPACE(1) $\subsetneq$ PrQSPACE(1)?

4.1.2. Languages Recognized by Kondacs-Watrous Quantum Finite Automata

In this section, we settle an open problem of Brodsky and Pippenger [9], giving a complete characterization of the class of languages recognized with unbounded error by RT-KWQFAs. It turns out that these restricted RT-QFAs, which are known to be inferior to RT-PFAs in the bounded error case, are equivalent to them in the unbounded error setting.

Lemma 4.8. *Any language recognized with cutpoint (or nonstrict cutpoint) $\frac{1}{2}$ by a*

Let S be a finite set and $\{A_s \mid s \in S\}$ be a set of $m \times m$-dimensional matrices such that the norm of each column belonging to $A_{s \in S}$ does not exceed 1. We present a method in order to find a set of $m \times m$-dimensional matrices, $\{B_s \mid s \in S\}$, with a generic constant l such that the columns of the matrix

$$\frac{1}{l}\left(\frac{A_s}{B_s}\right) \tag{4.18}$$

form an orthonormal set for each $s \in S$. The details of the method is given below.

- The entries of $B_{s \in S}$ are set to 0.
- l is set to $2m + 1$.
- For each $s \in S$, by executing the following loop, the entries of B_s are updated to make the length of each column of $\left(\frac{A_s}{B_s}\right)$ equal to l, and also to make the columns of $\left(\frac{A_s}{B_s}\right)$ pairwise orthogonal, where l must have been set to a value so that the loop works properly[a] .

  i. for $i = 1$ to $n + 2$

  ii. set l_i to the current length of the i^{th} column

  iii. set $b_{i,i}$ to $\sqrt{l^2 - l_i^2}$

  iv. for $j = i + 1$ to $n + 2$

  v. set $b_{i,j}$ to some value so that i^{th} and j^{th} columns can become orthogonal

[a] The unique constraint for the loop to work properly is that the value of l_i, calculated at the (ii)nd step, must be at most l. By setting l to $2m + 1$, the following bounds can be easily verified for each iteration of the loop:
(ii) $l_i < 2$ at the (ii)nd step;
(ii) $2m < |b_{i,i}| < 2m + 1$ at the (iii)rd step;
(ii) $|b_{j,i}| < \frac{1}{m}$ at the (v)th step.

Figure 4.4. General template to build a unitary matrix (I)

RT-PFA with n internal states can be recognized with cutpoint (or nonstrict cutpoint) $\frac{1}{2}$ by a RT-KWQFA with $O(n)$ internal states.

Proof. Let L be a language recognized by a RT-PFA with n internal states

$$\mathcal{P} = (Q, \Sigma, \{A_{\sigma \in \tilde{\Sigma}}\}, q_1, Q_a) \tag{4.19}$$

with (nonstrict) cutpoint $\frac{1}{2}$. We construct a RT-KWQFA

$$\mathcal{M} = (R, \Sigma, \{U_{\sigma\in\tilde{\Sigma}}\}, r_1, R_a, R_r) \tag{4.20}$$

with $(3n + 6)$ internal states recognizing L with (nonstrict) cutpoint $\frac{1}{2}$. The idea is to "embed" the (not necessarily unitary) matrices A_σ of the RT-PFA within the larger unitary matrices U_σ of the RT-KWQFA.

We define Q', v'_0, and $\{A'_{\sigma\in\tilde{\Sigma}}\}$ as follows:

i. $Q' = Q \cup \{q_{n+1}, q_{n+2}\}$;
ii. $v'_0 = (1, 0, \ldots, 0)^T$ is an $(n + 2)$-dimensional column vector;
iii. Each A'_σ is a $(n + 2) \times (n + 2)$-dimensional matrix:

$$A'_{\sigma\in\Sigma\cup\{¢\}} = \left(\begin{array}{c|c} A_\sigma & 0_{n\times 2} \\ \hline 0_{2\times n} & I_{2\times 2} \end{array}\right), A'_\$ = \left(\begin{array}{c|c} 0_{n\times n} & 0_{2\times n} \\ \hline T_{2\times n} & I_{2\times 2} \end{array}\right) \left(\begin{array}{c|c} A_\$ & 0_{n\times 2} \\ \hline 0_{2\times n} & I_{2\times 2} \end{array}\right), \tag{4.21}$$

where $T(1, i) = 1$ and $T(2, i) = 0$ when $q_i \in Q_a$, and $T(1, i) = 0$ and $T(2, i) = 1$ when $q_i \notin Q_a$ for $1 \leq i \leq n$.

For a given input $w \in \Sigma^*$,

$$v'_{|\tilde{w}|} = A'_\$ A'_{w_{|w|}} \cdots A'_{w_1} A'_¢ v_0. \tag{4.22}$$

It can easily be verified that

$$v'_{|\tilde{w}|} = (0_{1\times n} \mid f_\mathcal{P}(w), 1 - f_\mathcal{P}(w))^T. \tag{4.23}$$

For each $\sigma \in \tilde{\Sigma}$, we obtain constant l and the set $\{B_{\sigma\in\tilde{\Sigma}}\}$ for the set $\{A'_{\sigma\in\tilde{\Sigma}}\}$

according to template described in Figure 4.4 such that the columns of the matrix

$$\frac{1}{l}\left(\begin{array}{c} A'_\sigma \\ \hline B_\sigma \end{array}\right) \tag{4.24}$$

form an orthonormal set. Then, we obtain $U_{\sigma\in\tilde{\Sigma}}$ as

$$U_\sigma = \left(\begin{array}{c|c} \begin{array}{c} A''_\sigma \\ \hline B'_\sigma \\ \hline B''_\sigma \end{array} & D_\sigma \end{array}\right), \tag{4.25}$$

where $A''_\sigma = \frac{1}{l}A'_\sigma$, $B'_\sigma = B''_\sigma = \frac{1}{\sqrt{2}l}B_\sigma$, and the entries of D_σ are selected to make U_σ a unitary matrix.

The state set $R = R_n \cup R_a \cup R_r$ is specified as:

i. $r_{n+1} \in R_a$ corresponds to state q_{n+1};
ii. $r_{n+2} \in R_r$ corresponds to state q_{n+2};
iii. $\{r_1, \ldots, r_n\} \in R_n$ correspond to the states of Q, where r_1 is the start state;
iv. All the states defined for the rows of B'_σ and B''_σ are respectively accepting and rejecting states.

$\mathcal{M}$ simulates the computation of $\mathcal{P}$ for the input string w by multiplying the amplitude of each non-halting state with $\frac{1}{l}$ in each step. Hence, the top $n+2$ entries of the state vector of $\mathcal{M}$ equal

$$\left(\frac{1}{l}\right)^{|\tilde{w}|}(0_{1\times n}, f_{\mathcal{P}}(w), 1 - f_{\mathcal{P}}(w))^{\mathrm{T}} \tag{4.26}$$

just before the last measurement on the right end-marker. Note that, the halting states, except q_{n+1} and q_{n+2}, come in accept/reject pairs, so that transitions to them during the computation add equal amounts to the overall acceptance and rejection probabilities, and therefore not affect the decision on the membership of the input in

L. We conclude that

$$f_{\mathcal{M}}(w) > \frac{1}{2} \text{ if and only if } f_{\mathcal{P}}(w) > \frac{1}{2}, \tag{4.27}$$

and

$$f_{\mathcal{M}}(w) \geq \frac{1}{2} \text{ if and only if } f_{\mathcal{P}}(w) \geq \frac{1}{2}. \tag{4.28}$$

□

Theorem 4.9. *The class of languages recognized by RT-KWQFAs with unbounded error is uS (uQAL).*

Proof. Follows from Lemma 4.8, Lemma 3.2 and [26]. □

Corollary 4.10. *UMM = QAL ∩ coQAL = S ∩ coS.*

Proof. It is obvious that UMM ⊆ QAL ∩ coQAL. Let $L \in$ QAL ∩ coQAL. Then, there exist two RT-KWQFAs $\mathcal{M}_1$ and $\mathcal{M}_2$ such that for all $w \in L$, $f_{\mathcal{M}_1}(w) > \frac{1}{2}$ and $f_{\mathcal{M}_2}(w) \geq \frac{1}{2}$ and for all $w \notin L$, $f_{\mathcal{M}_1}(w) \leq \frac{1}{2}$ and $f_{\mathcal{M}_2}(w) < \frac{1}{2}$. Let $\mathcal{M}_3$ be a RT-KWQFA running $\mathcal{M}_1$ and $\mathcal{M}_2$ with equal probability. Thus, we obtain that for all $w \in L$, $f_{\mathcal{M}_3}(w) > \frac{1}{2}$, and for all $w \notin L$, $f_{\mathcal{M}_3}(w) < \frac{1}{2}$. Therefore, $L \in$ UMM. □

Considering this result together with Theorem 4.3, we conclude that, unlike classical deterministic and probabilistic finite automata, allowing the tape head to "stay put" for some steps during its left-to-right traversal of the input increases the language recognition power of quantum finite automata in the unbounded error case.

Since unbounded-error RT-PFAs and 2PFAs are equivalent in computational power [28], we are now able to state the following corollary to Theorem 4.3:

Corollary 4.11. *The class of languages recognized with unbounded error by constant-space PTMs is a proper subclass of the respective class for QTMs with restricted measurements.*

Also note that, since the algorithm described in the proof of Theorem 4.3 is presented for a 1KWQFA, Corollary 4.6 is still valid when coC$_=$QSPACE(1) is defined for QTMs with restricted measurements.

4.2. Nondeterministic Quantum Computation

We begin with some basic facts.

4.2.1. Basic Facts

For a fixed Σ, the pair $(\mathcal{A}, \lambda)$ is said to be *equivalent under cutpoint separation to* the pair $(\mathcal{A}', \lambda')$, denoted as $(\mathcal{A}, \lambda) \equiv (\mathcal{A}', \lambda')$, if the equalities

$$\{w \in \Sigma^* \mid f_{\mathcal{A}}(w) < \lambda\} = \{w \in \Sigma^* \mid f_{\mathcal{A}'}(w) < \lambda'\}, \tag{4.29}$$

$$\{w \in \Sigma^* \mid f_{\mathcal{A}}(w) = \lambda\} = \{w \in \Sigma^* \mid f_{\mathcal{A}'}(w) = \lambda'\}, \text{ and} \tag{4.30}$$

$$\{w \in \Sigma^* \mid f_{\mathcal{A}}(w) > \lambda\} = \{w \in \Sigma^* \mid f_{\mathcal{A}'}(w) > \lambda'\} \tag{4.31}$$

hold, where $\mathcal{A}$, $\mathcal{A}'$ are machines and $\lambda, \lambda' \in \mathbb{R}$ are cutpoints.

Fact 4.12. *[26] Let $\mathcal{G}_1$ be a GFA and $\lambda_1 \in \mathbb{R}$ be a cutpoint. For any cutpoint $\lambda_2 \in \mathbb{R}$, there exists a GFA $\mathcal{G}_2$ such that $(\mathcal{G}_1, \lambda_1) \equiv (\mathcal{G}_2, \lambda_2)$.*

Fact 4.13. *[29] Let $\mathcal{P}_1$ be a RT-PFA and $\lambda_1 \in [0, 1)$ be a cutpoint. For any cutpoint $\lambda_2 \in (0, 1)$, there exists a RT-PFA $\mathcal{P}_2$ such that $(\mathcal{P}_1, \lambda_1) \equiv (\mathcal{P}_2, \lambda_2)$.*

Fact 4.14. *(see also Lemma 4.8) [17, 23] For any RT-PFA $\mathcal{P}$, there exists a RT-KWQFA $\mathcal{M}$ such that $(\mathcal{P}, \frac{1}{2}) \equiv (\mathcal{M}, \frac{1}{2})$.*

Fact 4.15. *(see also Lemma 3.2) [23, 38] For any RT-KWQFA $\mathcal{M}$ and cutpoint $\lambda \in [0, 1)$, there exists a GFA $\mathcal{G}$ such that $(\mathcal{M}, \lambda) \equiv (\mathcal{G}, \lambda)$.*

Fact 4.16. *[26] For any GFA $\mathcal{G}$ and cutpoint $\lambda_1 \in \mathbb{R}$, there exist a RT-PFA $\mathcal{P}$ and a cutpoint $\lambda_2 \in (0,1)$ such that $(\mathcal{G}, \lambda_1) \equiv (\mathcal{P}, \lambda_2)$.*

Fact 4.17. *[43] MCL $\subsetneq S^{>}$.*

Fact 4.18. *Since MCQFA is a special case of RT-KWQFA, MCL $\subseteq$ QAL and NMCL $\subseteq$ NQAL.*

Fact 4.19. *[29] REG is a proper subset of both $S^{\neq}$ and $S^{=}$.*

Fact 4.20. *[29] $S^{\neq} \subsetneq S$ and $S \backslash S^{=} \neq \emptyset$.*

Fact 4.21. *[9, 44] REG is a proper subset of NQAL.*

4.2.2. Languages Recognized with One-Sided Error

RT-PFAs (and 1PFAs) can recognize all and only the regular languages with cutpoint 0 [45]. RT-KWQFAs (and so 1QFAs) can do more than that, as be characterized in this section.

We start the presentation of our main result by stating a fact which is useful in several proofs in this section.

Lemma 4.22. *For any language L, $L \in S^{\neq}$ if and only if there exists a GFA that recognizes L with one-sided cutpoint 0.*

Proof. The forward direction is proven on page 171 of [29]. In the reverse direction, if a GFA recognizes L with one-sided cutpoint 0, then $L \in \mathrm{S}^{\neq}$ by Fact 4.16. □

Lemma 4.23. *$S^{\neq} \subseteq NQAL$.*

Proof. If $L \in S^{\neq}$, then there exists an n-state RT-PFA $\mathcal{P} = (Q, \Sigma, \{A_{\sigma \in \tilde{\Sigma}}\}, q_1, Q_a)$ such that $w \in L \leftrightarrow f_{\mathcal{P}}(w) \neq \frac{1}{2}$. We define Q', v'_0, and $\{A'_{\sigma \in \Gamma}\}$ as follows:

i. $Q' = Q \cup \{q_{n+1}, q_{n+2}, q_{n+3}\}$;

ii. $v'_0 = (1, 0, \ldots, 0)$ is a $(n+3) \times 1$-dimensional column vector;

iii. Each A'_σ is a $(n+3) \times (n+3)$-dimensional matrix:

$$A'_{¢} = \left(\begin{array}{c|ccc} & 1 & \cdots & 1 \\ \frac{1}{2}A_{¢}[c_1] & 0 & \cdots & 0 \\ & \vdots & \ddots & \vdots \\ & 0 & \cdots & 0 \\ \hline 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \frac{1}{2} & 0 & \cdots & 0 \end{array}\right), \quad A'_{\sigma\in\Sigma} = \left(\begin{array}{c|ccc} A_\sigma & \multicolumn{3}{c}{0_{n\times 3}} \\ \hline & 1 & 0 & 0 \\ 0_{3\times n} & 0 & 1 & 0 \\ & 0 & 0 & 1 \end{array}\right),$$

$$A'_{\$} = \left(\begin{array}{ccc|ccc} \multicolumn{3}{c|}{0_{n\times n}} & \multicolumn{3}{c}{0_{n\times 3}} \\ \hline t_{1,1} & \cdots & t_{n,1} & 1 & 0 & -\frac{1}{2} \\ t_{1,2} & \cdots & t_{n,2} & 0 & 1 & \frac{1}{2} \\ 0 & \cdots & 0 & 0 & 0 & 0 \end{array}\right) \left(\begin{array}{c|ccc} A_{\$} & \multicolumn{3}{c}{0_{n\times 3}} \\ \hline & 1 & 0 & 0 \\ 0_{3\times n} & 0 & 1 & 0 \\ & 0 & 0 & 1 \end{array}\right),$$

where $A_{¢}[c_1]$ is the first column of $A_{¢}$; $t_{i,1} = 1$ and $t_{i,2} = 0$ when $q_i \in Q_a$, and $t_{i,1} = 0$ and $t_{i,2} = 1$ when $q_i \notin Q_a$ for $1 \leq i \leq n$.

For a given input $w \in \Sigma^*$, let $v'_{|\tilde{w}|} = A'_{\$} A'_{w_{|w|}} \cdots A'_{w_{|1|}} A'_{¢} v'_0$. It is easily verified that this computation "imitates" the processing of w by $\mathcal{P}$; the first n entries of the manipulated vector v' contain exactly the state vector of $\mathcal{P}$ (multiplied by $\frac{1}{2}$) in the corresponding steps of its execution. The last matrix multiplication results in

$$v'_{|\tilde{w}|} = \left(0_{n\times 1}, \frac{2f_{\mathcal{P}}(w) - 1}{4}, \frac{3 - 2f_{\mathcal{P}}(w)}{4}, 0\right). \tag{4.32}$$

The $(n+1)^{st}$ entry of $v'_{|\tilde{w}|}$ equals 0 if and only if $w \notin L$.

Using a modified version of the RT-PFA simulation method described in Section

4.1.2, we can construct a RT-KWQFA $\mathcal{M} = (R, \Sigma, \{U_{\sigma\in\Gamma}\}, r_1, R_a, R_r)$ recognizing L with cutpoint 0. For each $\sigma \in \tilde{\Sigma}$, U_σ is built as follows:

$$U_\sigma = \left(\begin{array}{c|c} \frac{1}{l}A'_\sigma & \\ \hline \frac{1}{l}B_\sigma & D_\sigma \end{array} \right), \tag{4.33}$$

where the constant l and the set $\{B_\sigma \mid \sigma \in \tilde{\Sigma}\}$ are obtained from $\{A'_\sigma \mid \sigma \in \tilde{\Sigma}\}$ according to the template described in Figure 4.4.

The state set $R = R_n \cup R_a \cup R_r$ is specified as:

i. $r_{n+1} \in R_a$ corresponds to state q_{n+1};
ii. $r_{n+2} \in R_r$ corresponds to state q_{n+2};
iii. $\{r_1, \ldots, r_n, r_{n+3}\} \in R_n$ correspond to the remaining states of Q', where r_1 is the start state;
iv. All the new states that are defined during the construction of $\{U_{\sigma\in\Gamma}\}$ are rejecting ones.

$\mathcal{M}$ simulates the computation of $\mathcal{P}$ for a given input string $w \in \Sigma^*$ by representing the probability of each state q_j by the amplitude of the corresponding state r_j. The transitions from the $2n + 6$ states added during the construction of $U_{\sigma\in\Gamma}$ for ensuring unitarity do not interfere with this simulation, since the computation halts immediately on the "branches" where these states are entered. Therefore, the top $n + 3$ entries of the state vector of $\mathcal{M}$ equal

$$\left(\frac{1}{l}\right)^{|\tilde{w}|} \left(0_{n\times 1}, \frac{2f_{\mathcal{P}}(w) - 1}{4}, \frac{3 - 2f_{\mathcal{P}}(w)}{4}, 0\right)^{\mathrm{T}} \tag{4.34}$$

just before the last measurement on \$. Since the amplitude of the only accepting state is nonzero if and only if $w \in L$, L is recognized by $\mathcal{M}$ with cutpoint 0. □

Lemma 4.24. *$NQAL \subseteq S^{\neq}$.*

Proof. By Fact 4.15, there exists a GFA with one-sided cutpoint 0 for any member L of NQAL. By Lemma 4.22, L is an exclusive stochastic language. □

Theorem 4.25. *The class of the languages recognized by RT-KWQFA with one-sided unbounded error is precisely equal to* $S^{\neq}$.

Corollary 4.26. $S^{\neq} = NQAL$.

Corollary 4.27. $S^{=} = coNQAL$.

By Fact 4.20, there exist languages that RT-KWQFAs can recognize with two-sided (unbounded), but not one-sided error. The class of these languages is precisely $\mathrm{uS} \setminus (\mathrm{S}^{\neq} \cup \mathrm{S}^{=})$. Note that the above results also establish that the class of languages recognized by RT-NQFAs is not closed under complementation (Fact 4.46).

Open Problem 6. Does $\mathrm{S}^{\neq} \cap \mathrm{S}^{=}$ contain a nonregular language?

Open Problem 7. Is $\mathrm{S}^{\neq}$ countable or uncountable?

Watrous [7] has shown that $\mathrm{NQSPACE}_{\mathbb{Q}}(s) = \mathrm{coC}_{=}\mathrm{SPACE}_{\mathbb{Q}}(s)$ for $s = \Omega(\log(n))$. Due to the fact [28] that $\mathrm{S}^{\neq} = \mathrm{coC}_{=}\mathrm{SPACE}(1)$, we have proven that co-$\mathrm{C}_{=}$SPACE(1) $\subseteq$ NQSPACE(1), and whether the inclusion is strict or not depends on whether a two-way (or one-way) head would increase the computational power of a NQFA[15] .

Open Problem 8. Can NQFAs with a two-way (or one-way) tape head recognize more languages than the realtime model discussed here?

For sublogarithmic space, we can conclude the following corollary, firstly stated in [46], by combining some previously known facts:

Corollary 4.28. *NSPACE(s)* $\subsetneq$ *NQSPACE(s) for* $s = o(\log(n))$.

Proof. (Sketch.) QTMs can simulate PTMs easily for any common space bound. $L_{neq} = \{w \in \{a,b\}^* \mid |w|_a \neq |w|_b\} \in \mathrm{S}^{\neq}$ [43]. It is easily seen that L_{neq} is a non-

[15]For any 2PFA $\mathcal{M}$ and cutpoint $\lambda_1 \in [0,1)$, there exist a one-way PFA $\mathcal{P}$ and a cutpoint $\lambda_2 \in [0,1)$ such that $(\mathcal{M}, \lambda_1) \equiv (\mathcal{P}, \lambda_2)$ [28], whereas one-way QFAs are more powerful than RT-QFAs in the general unbounded error setting.

regular deterministic context-free language (DCFL). It is known that no nonregular DCFL is in NSPACE(s) for $s = o(\log(n))$ [47]. □

For space bounds $s \in \Omega(\log(n))$, all we know in this regard is the trivial fact that NSPACE(s) $\subseteq$ NQSPACE(s).

Open Problem 9. Is NSPACE(s) $\subsetneq$ NQSPACE(s), for $s \in \Omega(\log n)$?

4.2.3. Space Efficiency of Nondeterministic Quantum Finite Automata

It is well known [48, 49] that some infinite families of languages can be recognized with one-sided bounded error by just tuning the transition amplitudes of a RT-QFA with a constant number of states, whereas the sizes of the corresponding RT-PFAs grow without bound. After a simple example, we argue that this advantage is also valid in the unbounded error case.

For $m \in \mathbb{Z}^+$, $L_m \subseteq \{a\}^*$ is defined as

$$L_m = \{a^i \mid i \mod (m) \neq 0\}. \tag{4.35}$$

Theorem 4.29. *For any $m > 1$, L_m can be recognized by a 2-state MCQFA*[16] *with cutpoint* 0.

Proof. $\mathcal{M}$ begins the computation at state q_0, and each transition with the symbol a corresponds to a rotation[17] by angle $\frac{\pi}{m}$ in the $|q_0\rangle$-$|q_1\rangle$ plane, where q_1 is the accepting state. □

For any positive n, it is known [45] that every n-state RT-PFA with cutpoint 0 has an equivalent nondeterministic finite automaton with the same number of states.

[16] There is an equivalent 4-state RT-KWQFA.

[17] For details of a similar construction for a nonregular language, see [43].

Therefore, only finitely many distinct languages can be recognized with one-sided unbounded error by RT-PFAs with at most n states.

Combining this with the fact that any n-state RT-PFA with cutpoint 0 can be simulated by a RT-KWQFA with $2n + 6$ states (see Section 4.1.2), the superiority of RT-KWQFAs (and so RT-QFAs) over RT-PFAs in this regard is established.

4.2.4. Languages Recognized with Two-Sided Error

To gain a better understanding of the classes of languages recognizable by positive one-sided, negative one-sided, and necessarily two-sided error by QFAs, we examine some examples from each of those families. Bertoni and Carpentieri [43] showed that L_{neq} is in NMCL, and that its complement, say, L_{eq}, is not in MCL. Now that we have Theorem 4.25, we can use the well-known results [29, 45] from the RT-PFA literature that state that $L_{eq} \in \mathrm{S}^{=}$, $L_{neq} \in \mathrm{S}^{\neq}$, but not vice versa, to conclude that stronger RT-QFA variants also can not recognize L_{eq} with positive one-sided error, and neither can they recognize L_{neq} with negative one-sided error. Similarly, Lāce *et al.* [11] proved recently that the complement of the palindrome language $L_{pal} = \{w \in \{a, b\}^* \mid w = w^r\}$ is in NQAL. We can show the corresponding result for L_{pal} using the following fact:

Fact 4.30. *[50] Let $L \in S^{=}$. Then there exists a natural number*[18] *$n \geq 1$ such that for any strings $u, v, y \in \Sigma^*$,*

$$\text{if } uv, uyv, \ldots, uy^{n-1}v \in L, \text{ then } uy^*v \subseteq L. \tag{4.36}$$

Theorem 4.31. *$L_{pal} \notin S^{\neq}$.*

Proof. Suppose that $L_{pal} \in \mathrm{S}^{\neq}$. Then $\overline{L_{pal}} \in \mathrm{S}^{=}$. Let $u = a^n b$, $y = a$, and $v = \varepsilon$.

$$a^n b, a^n ba, \ldots, a^n ba^{n-1} \in \overline{L_{pal}} \tag{4.37}$$

[18] This number can be chosen as the number of states of a PFA $\mathcal{P}$ such that $L = \mathbb{L}(\mathcal{P}, = \lambda)$ for some $\lambda \in [0, 1]$.

imply that $a^n b a^n \in \overline{L_{pal}}$ by Fact 4.30. Since this string is actually a member of L_{pal}, we have a contradiction. □

We now exhibit some languages which can only be recognized by two-sided error by a QFA.

Theorem 4.32. $L = \{aw_1 \cup bw_2 \mid w_1 \in L_{eq}, w_2 \in L_{neq}\} \in S \backslash (S^{=} \cup S^{\neq})$.

Proof. Suppose that $L \in \mathrm{S}^{\neq}$, then there exists a GFA

$$\mathcal{G} = (S, \Sigma, \{A_{\sigma \in \{a,b\}}\}, v_0, f) \tag{4.38}$$

recognizing L with one-sided cutpoint 0. The GFA

$$\mathcal{G}' = (S, \Sigma, \{A_{\sigma \in \{a,b\}}\}, A_a v_0, f) \tag{4.39}$$

recognizes L_{eq} with one-sided cutpoint 0, meaning that $L_{eq} \in \mathrm{S}^{\neq}$. This contradicts the well-known fact mentioned in the first paragraph of this section. Suppose now that $L \in \mathrm{S}^{=}$, then

$$\overline{L} = \{\varepsilon \cup aw_2 \cup bw_1 \mid w_1 \in L_{eq}, w_2 \in L_{neq}\} \tag{4.40}$$

is in $\mathrm{S}^{\neq}$, which also results in a contradiction for the same reason. Since both L_{eq} and its complement are stochastic, it is not difficult to show that L is stochastic. □

Lemma 4.33. $L_{lt} = \{w \in \{a,b\}^* \mid |w|_a < |w|_b\} \notin (S^{=} \cup S^{\neq})$.

Proof. Suppose that $L_{lt} \in \mathrm{S}^{=}$. Let $u = \varepsilon$, $y = a$, and $v = b^n$.

$$b^n, ab^n, \ldots, a^{n-1}b^n \in L_{lt} \tag{4.41}$$

imply that $a^n b^n \in L_{lt}$ by Fact 4.30. Since this string is actually a member of $\overline{L_{lt}}$, we have a contradiction.

Similarly, suppose that $L_{lt} \in \mathrm{S}^{\neq}$, or $\overline{L_{lt}} \in \mathrm{S}^{=}$. Let $u = a^n$, $y = b$, and $v = b$.

$$a^n b, a^n b^2, \ldots, a^n b^n \in \overline{L_{lt}} \tag{4.42}$$

imply that $a^n b^{n+1} \in \overline{L_{lt}}$ by Fact 4.30. Since this string is actually a member of L_{lt}, we have a contradiction. □

Corollary 4.34. $L_{lt} \in S \backslash (S^{=} \cup S^{\neq})$.

Proof. This follows from Lemma 4.33 and the fact that $L_{lt} \in \mathrm{S}$ [28, 51]. □

Theorem 4.35. $L_{eq \cdot b} = L_{eq} \cdot b^{+} \in S \backslash (S^{=} \cup S^{\neq})$.

Proof. The proof of $L_{eq \cdot b} \notin (\mathrm{S}^{=} \cup \mathrm{S}^{\neq})$ uses the setup presented in Lemma 4.33, i.e.,

i. select $u = \varepsilon$, $y = a$, and $v = b^n$ to contradict with $L_{eq \cdot b} \in \mathrm{S}^{=}$,
ii. select $u = a^n$, $y = b$, and $v = b$ to contradict with $L_{eq \cdot b} \in \mathrm{S}^{\neq}$.

Any string w is a member of $L_{eq \cdot b}$ if and only if it has the following three properties:

- w ends with b.
- $w \in L_{lt}$.
- Let u be the longest prefix of w ending with a ($u = \varepsilon$ if $w \in \{b^*\}$). Then, $u \in \overline{L_{lt}}$.

Since these properties can be checked easily by a 2PFA with bounded error, $L_{eq \cdot b} \in \mathrm{S}$ [28, 51]. □

Now, we show the stochasticity of an important family of languages.

The *word problem* for a group is the problem of deciding whether or not a product of group elements is equal to the identity element [52]. Let $\mathcal{G}^k = (G, \circ)$ be a finitely generated free group with a basis

$$\Sigma = \{\sigma_1, \ldots, \sigma_k, \sigma_1^{-1}, \ldots, \sigma_k^{-1}\}, \tag{4.43}$$

where $k \in \mathbb{Z}^+$ is the rank of $\mathcal{G}^k$. $L_{wp}(\mathcal{G}^k) \subseteq \Sigma^*$ is the language defined as

$$L_{wp}(\mathcal{G}^k) = \{w \mid w_i \in \Sigma, 1 \leq i \leq |w|, w_1 \circ \cdots \circ w_{|w|} = \imath\}, \tag{4.44}$$

where $\imath \in G$ is the identity element of $\mathcal{G}^k$.

Fact 4.36. *(Page 1 of [53]) Let $\mathcal{G}_1^{k_1}$ and $\mathcal{G}_2^{k_2}$ be finitely generated free groups. Then $\mathcal{G}_1^{k_1}$ and $\mathcal{G}_2^{k_2}$ are isomorphic if and only if $k_1 = k_2$.*

Corollary 4.37. *$L_{wp}(\mathcal{G}_1^{k_1})$ and $L_{wp}(\mathcal{G}_2^{k_2})$ are isomorphic if and only if $k_1 = k_2$, where $\mathcal{G}_1^{k_1}$ and $\mathcal{G}_2^{k_2}$ are finitely generated free groups.*

As a generic name, $L_{wp}(k)$ can be used instead of $L_{wp}(\mathcal{G}^k)$ due to Corollary 4.37, where $k \in \mathbb{Z}^+$.

Fact 4.38. *[54] $L_{wp}(1) \in S$.*

Fact 4.39. *[9] $L_{wp}(k) \in$ coNMCL, the class of languages whose complements are in NMCL, for any $k \in \mathbb{Z}^+$.*

Corollary 4.40. *$L_{wp}(k) \in S^{=}$ for any $k \in \mathbb{Z}^+$.*

We now provide a proof of the following theorem.

Theorem 4.41. *$L_{wp}(k) \in S$ for any $k \geq 2$.*

In fact, Theorem 4.41 was stated as a corollary on page 1463 of [9], but the purported proof there was based on the claim that coNMCL$\subseteq$MCL$\subseteq$ S. It is however known [43], as we mentioned above, that a member of coNMCL (L_{eq}) lies outside MCL. Furthermore, the same demonstration can be easily extended to $L_{wp}(k)$, where $k \in \mathbb{Z}^+$.

Corollary 4.42. *$L_{wp}(k) \notin MCL$ for any $k \in \mathbb{Z}^+$.*

Since it is still an open problem whether $\mathrm{S}^{=} \subseteq \mathrm{S}$ or not, we cannot use Corollary 4.40 directly to prove Theorem 4.41. Instead, we focus on a subclass of $\mathrm{S}^{=}$ that is known to be a subset of S, $\mathrm{S}^{=}_{\mathbb{Q}}$ [55].

Fact 4.43. *[55] $S^{=}_{\mathbb{Q}} \subsetneq S$.*

$\mathrm{SO}_3(\mathbb{Q})$ is the group of rotations on $\mathbb{R}^3$ that are 3×3 dimensional orthogonal matrices having only rational entries with determinant $+1$.

Fact 4.44. *[56, 57] For any $k \geq 2$, $SO_3(\mathbb{Q})$ contains a free subgroup with rank k, namely $\mathcal{S}^k$.*

Proof of Theorem 4.41. For $L_{wp}(k)$, we define a rational GFA

$$\mathcal{G}_k = (\{s_1, s_2, s_3\}, \Sigma, \{A_{\sigma \in \Sigma}\}, v_0, f), \tag{4.45}$$

where

i. $\Sigma = \{R_1, \ldots, R_k, R_1^{-1}, \ldots, R_k^{-1}\}$ is a basis of $\mathcal{S}^k$;
ii. $A_\sigma = \sigma$ for each $\sigma \in \Sigma$;
iii. $v_0 = (1 \;\; 0 \;\; 0)^{\mathrm{T}}$;
iv. $f = (1 \;\; 0 \;\; 0)$.

It is obvious that $w \in L_{wp}(k)$ if and only if $A_w \cdots A_1 = I_{3\times 3}$ if and only if $f_{\mathcal{G}_k}(w) = fA_w \cdots A_1 v_0 = 1$, where $w \in \Sigma^*$. Thus, $L_{wp}(k) \in \mathrm{S}^{=}_{\mathbb{Q}}$ by selecting the cutpoint as 1. We can conclude with Fact 4.43. □

4.2.5. Closure Properties

The previously discovered closure properties of S, $\mathrm{S}^{\neq}$ and $\mathrm{S}^{=}$ are listed below.

Fact 4.45.

i. *S is not closed under union and intersection [58–60].*
ii. *S is closed under union and intersection with a regular language [61, 62].*
iii. *S is closed under reversal [26].*
iv. *S is not closed under concatenation, Kleene closure, and homomorphism [63, 64].*
v. *S is closed under complementation over unary alphabets [59].*

Fact 4.46.

i. *Both $S^{\neq}$ and $S^{=}$ are closed under union and intersection [29].*
ii. *Neither $S^{\neq}$ nor $S^{=}$ is closed under complementation [50].*
iii. *S is closed under intersection with a member of $S^{\neq}$ [29].*

In [46], we proved several new nontrivial closure properties of the "one-sided" classes $\mathrm{S}^{\neq}$ and $\mathrm{S}^{=}$. We refer the reader to [46] for the proofs of the following theorems.

Theorem 4.47. *$S^{\neq}$ is closed under concatenation.*

Theorem 4.48. *$S^{=}$ is not closed under concatenation.*

Theorem 4.49. *$S^{\neq}$ is closed under Kleene closure.*

Theorem 4.50. *$S^{=}$ is not closed under Kleene closure.*

Lemma 4.51. *Let $h : \Sigma \to \Sigma \setminus \{\kappa\}$ be a homomorphism such that*

$$h(\sigma) = \begin{cases} \sigma, & \sigma \neq \kappa \\ \varepsilon, & \sigma = \kappa \end{cases}, \tag{4.46}$$

where κ is a specific symbol in Σ. If $L \subseteq \Sigma^*$ is in $S^{\neq}$, then so is $h(L)$.

Lemma 4.52. *Let $h : \Sigma \to \Upsilon^*$ be a homomorphism such that $|h(\sigma)| > 0$ for all $\sigma \in \Sigma$. If $L \subseteq \Sigma^*$ is in $S^{\neq}$, then so is $h(L)$.*

Theorem 4.53. *$S^{\neq}$ is closed under homomorphism.*

Theorem 4.54. *$S^{\neq}$ and $S^{=}$ are closed under inverse homomorphism.*

Theorem 4.55. *$S^{\neq}$ and $S^{=}$ are closed under reversal.*

Theorem 4.56. *$S^{\neq}$ and $S^{=}$ are closed under word quotient.*

Theorem 4.57. *$S^{\neq}$ and $S^{=}$ are not closed under difference.*

Theorem 4.58. *$S^{\neq}$ and $S^{=}$ are closed under difference with a regular language.*

For completeness, we list below the following easy facts about MCL and NMCL:

i. NMCL is closed under both union and intersection.
ii. Neither MCL nor NMCL is closed under complementation [43].
iii. Both MCL and NMCL are closed under inverse homomorphism [37].
iv. Both MCL and NMCL are closed under word quotient [9].

Some related open problems are as follows:

Open Problem 10. Is MCL closed under union? Intersection?

Open Problem 11. Do NMCL and MCL coincide?

Open Problem 12. Is S closed under complementation? (page 158 of [29])

Open Problem 13. Is $S^{=}$ a subset of S? (page 173 of [29])

5. CONSTANT-SPACE BOUNDED-ERROR COMPUTATION

In this chapter, we focus on bounded-error computation in constant space. The results presented in Sections 5.1.3, 5.1.4, 5.2, and 5.3 are obtained by our new two-way PFAs and QFAs (defined in Section 5.1.1). In the second part of the chapter (Section 5.4), we present our results related with realtime PFAs and QFAs with postselection.

5.1. Probabilistic and Quantum Automata with Resetting

In this section, we keep the original definitions of the quantum machines having capability of resetting as stated in [65] (the underlying model is RT-KWQFA) since the computational power of them does not increase (as shown in Theorem 5.10) when the underlying model is selected as RT-QFA.

5.1.1. Definitions

A *two-way quantum finite automaton with reset* (2QFA$^{\circlearrowleft}$) is a 7-tuple

$$\mathcal{M} = (Q, \Sigma, \delta, q_1, Q_a, Q_r, Q_{reset} = \cup_{q \in Q_n} Q_q^{\circlearrowleft}). \tag{5.1}$$

In contrast to the previous models,

i. $Q_n = Q \setminus (Q_a \cup Q_r \cup Q_{reset})$ is the set of nonhalting and nonresetting states;
ii. Q_{reset} is the union of disjoint reset sets, i.e., each $Q_{q \in Q_n}^{\circlearrowleft}$ contains reset states that cause the computation to restart with state q.

We assume that the states in Q_n have smaller indices than other members of Q; $q_i \in Q_n$ for $1 \leq i < |Q_n|$.

Apart from the left reset capability, 2QFA$^{\circlearrowleft}$s are identical to 2KWQFAs. (We refer the reader to Section 3.2.5 and [4] for detailed coverage of the technical properties of 2KWQFAs.) In each step of its execution, a 2QFA$^{\circlearrowleft}$ undergoes two linear operations: The first one is a unitary transformation of the current superposition according to δ, and the second one is the measurement, done on the configuration set, $\mathcal{C}^w$, with set of projectors, $P_{\tau\in\Delta}$, for a given input string $w \in \Sigma^*$, where

- $\Delta = \{a, n, r\} \cup \{\text{reset-}i \mid 1 \leq i \leq |Q_n|\}$,
- $\mathcal{C}^w_{\tau\in\{a,n,r\}} = \{c \mid c \in Q_\tau \times \{1, \ldots, |\tilde{w}|\}\}$,
- $\mathcal{C}^w_{\text{reset-}i} = \{c \mid c \in Q^{\circlearrowleft}_{q_i\in Q_n} \times \{1, \ldots, |\tilde{w}|\}\}$ for $1 \leq i \leq |Q_n|$, and
- $P_{\tau\in\Delta} = \sum_{c\in\mathcal{C}^w_\tau} |c\rangle\langle c|$.

Thus, the outcome of the measurement is one of "accept", "reject", "continue without resetting", or "reset with state q", for any $q \in Q_n$. Note that, if "reset with state q" is measured, the tape head is reset to point to the left end-marker, and the machine continuous from the superposition $|q, 1\rangle$ in the next step.

A *two-way quantum finite automaton with restart* (2QFA$^{\circlearrowleft}$) is a restricted 2QFA$^{\circlearrowleft}$ in which the "reset moves" can target only the original start state of the machine, that is, in terms of Equation 5.1, all the $Q^{\circlearrowleft}_q$ of a 2QFA$^{\circlearrowleft}$ are empty, with the exception of $Q^{\circlearrowleft}_{q_1}$, represented as $Q^{\circlearrowleft}$.

Other variants of two-way automata with reset that are examined in this thesis are

i. A *realtime (Kondacs-Watrous) quantum finite automaton with reset* (RT-QFA$^{\circlearrowleft}$) is a restricted 2QFA$^{\circlearrowleft}$ which uses neither "move one square to the left" nor "stay put" transitions, and whose tape head is therefore classical,
ii. A *realtime (Kondacs-Watrous) quantum finite automaton with restart* (RT-QFA$^{\circlearrowleft}$) is a RT-QFA$^{\circlearrowleft}$ where the reset moves can target only the original start state, and,
iii. A *realtime probabilistic finite automaton with restart* (RT-PFA$^{\circlearrowleft}$) is a RT-PFA which has been enhanced with the capability of resetting the tape head to the

left end-marker and swapping to the original start state.

5.1.2. Basic Facts

We start by stating some basic facts concerning automata with restart, which are used in later sections.

A segment of computation which begins with a (re)start, and ends with a halting or restarting configuration is called a *round*. Clearly, every automaton with restart which makes nontrivial use of its restarting capability runs for infinitely many rounds on some input strings. Throughout this chapter, we make the assumption that our two-way automata do not contain infinite loops within a round, that is, the computation restarts or halts with probability 1 in a finite number steps for each round.

Everywhere in this section, $\mathcal{R}$ stands for a finite state automaton with restart, and $w \in \Sigma^*$ represents an input string using the alphabet Σ. $p^a_{\mathcal{R}}(w)$ (resp., $p^r_{\mathcal{R}}(w)$) denote the probability that $\mathcal{R}$ accepts (resp., rejects) w, in the first round. Moreover, $p^h_{\mathcal{R}}(w) = p^a_{\mathcal{R}}(w) + p^r_{\mathcal{R}}(w)$.

Lemma 5.1.

$$f^a_{\mathcal{R}}(w) = \frac{1}{1 + \dfrac{p^r_{\mathcal{R}}(w)}{p^a_{\mathcal{R}}(w)}}; \quad f^r_{\mathcal{R}}(w) = \frac{1}{1 + \dfrac{p^a_{\mathcal{R}}(w)}{p^r_{\mathcal{R}}(w)}}. \tag{5.2}$$

Proof.

$$\begin{aligned} f^a_{\mathcal{R}}(w) &= \sum_{i=0}^{\infty} \left(1 - p^a_{\mathcal{R}}(w) - p^r_{\mathcal{R}}(w)\right)^i p^a_{\mathcal{R}}(w) \\ &= p^a_{\mathcal{R}}(w) \left(\frac{1}{1 - (1 - p^a_{\mathcal{R}}(w) - p^r_{\mathcal{R}}(w))} \right) \\ &= \frac{p^a_{\mathcal{R}}(w)}{p^a_{\mathcal{R}}(w) + p^r_{\mathcal{R}}(w)} \\ &= \frac{1}{1 + \frac{p^r_{\mathcal{R}}(w)}{p^a_{\mathcal{R}}(w)}}. \end{aligned}$$

$f^r_{\mathcal{R}}(w)$ is calculated in the same way. □

Lemma 5.2. *The language $L \subseteq \Sigma^*$ is recognized by $\mathcal{R}$ with error bound $\epsilon > 0$ if and only if $\frac{p^r_{\mathcal{R}}(w)}{p^a_{\mathcal{R}}(w)} \leq \frac{\epsilon}{1-\epsilon}$ when $w \in L$, and $\frac{p^a_{\mathcal{R}}(w)}{p^r_{\mathcal{R}}(w)} \leq \frac{\epsilon}{1-\epsilon}$ when $w \notin L$. Furthermore, if $\frac{p^r_{\mathcal{R}}(w)}{p^a_{\mathcal{R}}(w)}$ (resp., $\frac{p^a_{\mathcal{R}}(w)}{p^r_{\mathcal{R}}(w)}$) is at most ϵ, then $f^a_{\mathcal{R}}(w)$ (resp, $f^r_{\mathcal{R}}(w)$) is at least $1-\epsilon$.*

Proof. This follows from Lemma 5.1, since, for all $p \geq 0$, $\epsilon \in [0, \frac{1}{2})$,

$$\frac{1}{1+p} \geq 1-\epsilon \Leftrightarrow p \leq \frac{\epsilon}{1-\epsilon}, \text{ and} \tag{5.3}$$

$$p \leq \epsilon \Rightarrow \frac{1}{1+p} \geq 1-\epsilon. \tag{5.4}$$

□

Lemma 5.3. *Let $p = p^h_{\mathcal{R}}(w)$, and let $s(w)$ be the maximum number of steps in any branch of a round of $\mathcal{R}$ on w. The worst-case expected runtime of $\mathcal{R}$ on w is*

$$\frac{1}{p}(s(w)). \tag{5.5}$$

Proof. The worst-case expected running time of $\mathcal{R}$ on w is

$$\sum_{i=0}^{\infty}(i+1)(1-p)^i(p)(s(w)) = (p)(s(w))\frac{1}{p^2} = \frac{1}{p}(s(w)). \tag{5.6}$$

□

Lemma 5.4. *Any one-way automaton with restart with expected runtime t can be simulated by a corresponding two-way automaton without restart in expected time no more than $2t$.*

Proof. The program of the two-way machine ($\mathcal{R}_2$) is identical to that of the one-way machine with restart ($\mathcal{R}_1$), except for the fact that each restart move of $\mathcal{R}_1$ is imitated

by $\mathcal{R}_2$ by moving the head one square per step all the way to the left end-marker. This causes the runtimes of the i nonhalting rounds in the summation in Equation (5.6) in Lemma 5.3 to increase by a factor of 2. □

We now give a quick review of the technique of probability amplification. Suppose that we are given a machine (with or without reset) $\mathcal{A}$, which recognizes a language L with error bounded by ϵ, and we wish to construct another machine which recognizes L with a much smaller, but still positive, probability of error, say, ϵ'. It is well known[19] that one can achieve this by running $\mathcal{A}$ $O(\log(\frac{1}{\epsilon'}))$ times on the same input, and then giving the majority answer as our verdict about the membership of the input string in L.

Suppose that the original machine $\mathcal{A}$ needs to be run $2k+1$ times for the overall procedure to work with the desired correctness probability. Two counters can be used to count the acceptance and rejection responses, and the overall computation accepts (resp., rejects) when the number of recorded acceptances (resp., rejections) reaches $k+1$. To implement these counters in the finite automaton setting, we need to "connect" $(k+1)^2$ copies of $\mathcal{A}$, $\{\mathcal{A}_{i,j} \mid 0 \leq i, j \leq k\}$, where the subscripts indicate the values of the two counters, i.e., the states of $\mathcal{A}_{i,j}$ encode the information that $\mathcal{A}$ has accepted i times and rejected j times in its previous runs. The new machine $\mathcal{M}$ is constructed from the $\mathcal{A}_{i,j}$'s as follows:

- The start state of $\mathcal{M}$ is the start state of $\mathcal{A}_{0,0}$;
- Upon reaching any accept state of $\mathcal{A}_{i,j}$ $(0 \leq i, j < k)$, $\mathcal{M}$ moves the head back to the left end-marker and then switches to the start state of $\mathcal{A}_{i+1,j}$;
- Upon reaching any reject states of $\mathcal{A}_{i,j}$ $(0 \leq i, j < k)$, $\mathcal{M}$ moves the head back to the left end-marker and then switches to the start state of $\mathcal{A}_{i,j+1}$;
- The accept states of $\mathcal{M}$ are the accept states of $\mathcal{A}_{k,j}$ $(0 \leq j < k)$;
- The reject states of $\mathcal{M}$ are the reject states of $\mathcal{A}_{i,k}$ $(0 \leq i < k)$.

Lemma 5.5. *If language $L \subseteq \Sigma^*$ is recognized by $\mathcal{R}$ with a fixed error bound $\epsilon > 0$,*

[19] See, for instance, pages 369-370 of [34].

then for any positive error bound $\epsilon' < \epsilon$, there exists a finite automaton with reset, $\mathcal{R}'$, recognizing L. Moreover, if $\mathcal{R}$ has n states and its (expected) runtime is $O(s(|w|))$, then $\mathcal{R}'$ has $O(\log^2(\frac{1}{\epsilon'})n)$ states, and its (expected) runtime is $O(\log(\frac{1}{\epsilon'})s(|w|))$, where w is the input string.

Proof. Follows easily from the above description. □

Finally, we note the following relationship between the computational powers of the 2CQFA and the RT-QFA$^{\circlearrowleft}$.

Lemma 5.6. *For any RT-QFA$^{\circlearrowleft}$ $\mathcal{M}_1$ with n states and expected runtime $t(|w|)$, there exists a 2CQFA $\mathcal{M}_2$ with n states and expected runtime $O(t(|w|))$, such that $\mathcal{M}_2$ accepts every input string w with the same probability that $\mathcal{M}_1$ accepts w.*

5.1.3. Computational Powers of Realtime Probabilistic Finite Automata with Restart

It is interesting to examine the power of the restart move in classical computation. Any RT-PFA$^{\circlearrowleft}$ which runs in expected t steps can be simulated by a 2PFA which runs in expected $2t$ steps (see Lemma 5.4). We ask in this section whether the restart move can substitute the "left" and "stationary" moves of a 2PFA without loss of computational power. Since every polynomial-time 2PFA recognizes a regular language, which can of course be recognized by using only "right" moves, we focus on the best-known example of a nonregular language that can be recognized by an exponential-time 2PFA.

Theorem 5.7. *There exists a natural number k, such that for any $\epsilon > 0$, there exists a k-state RT-PFA$^{\circlearrowleft}$ $\mathcal{P}_\epsilon$ recognizing language L_{eq} with error bound ϵ and expected runtime $O((\frac{2}{\epsilon^2})^{|w|}|w|)$, where w is the input string.*

Proof. We construct the RT-PFA$^{\circlearrowleft}$ $\mathcal{P}_\epsilon$, shortly $\mathcal{P}$, as follows: Let $x = \frac{\epsilon^2}{2}$. The computation splits into three paths called path_1, path_2, and path_3 with equal probabilities on symbol ¢. All three paths, while performing their main tasks, parallelly check whether

the input is of the form a^*b^*, if not, all paths simply reject. The main tasks of the paths are as follows:

- path_1 moves on with probability x and restarts with probability $1-x$ when reading symbols a and b. After reading the right end-marker \$, it accepts with probability with 1.
- path_2 moves on with probability x^2 and restarts with probability $1-x^2$ when reading symbol a. On b's, it continuous with the "syntax" check. After reading the \$, it rejects with probability $\frac{\epsilon}{2}$ and restarts with probability $1-\frac{\epsilon}{2}$.
- path_3 is similar to path_2, except that the transitions of symbols a and b are interchanged.

If the input is of the form $a^m b^n$, then the accept and reject probabilities of the first round are calculated as

$$p^a_{\mathcal{P}}(w) = \frac{1}{3}x^{m+n}, \quad \text{and} \quad p^r_{\mathcal{P}}(w) = \frac{\epsilon}{6}\left(x^{2m}+x^{2n}\right). \tag{5.7}$$

If $m = n$, then

$$\frac{p^r_{\mathcal{P}}(w)}{p^a_{\mathcal{P}}(w)} = \epsilon. \tag{5.8}$$

If $m \neq n$ (assume without loss of generality that $m = n + d$ for some $d \in \mathbb{Z}^+$) , then

$$\frac{p^a_{\mathcal{P}}(w)}{p^r_{\mathcal{P}}(w)} = \frac{2}{\epsilon}\frac{x^{2n+d}}{x^{2n+2d}+x^{2n}} = \frac{2}{\epsilon}\frac{x^d}{x^{2d}+1} < \frac{2}{\epsilon}x^d \leq \frac{2}{\epsilon}x \tag{5.9}$$

By replacing $x = \dfrac{\epsilon^2}{2}$, we can get

$$\frac{p^a_{\mathcal{P}}(w)}{p^r_{\mathcal{P}}(w)} < \epsilon. \tag{5.10}$$

By using Lemma 5.2, we can conclude that $\mathcal{P}$ recognizes L_{eq} with error bound ϵ.

Since $p^h_{\mathcal{P}}(w)$ is always greater than $\frac{1}{3}x^{|w|}$, the expected runtime of the algorithm is $O((\frac{2}{\epsilon^2})^{|w|}|w|)$, where w is the input string. □

5.1.4. Computational Powers of Realtime Quantum Finite Automata with Restart

In this section, we focus on the RT-QFA$^{\circlearrowleft}$, which turns out to be the simplest and most restricted known model of quantum computation that is strictly superior in terms of bounded-error language recognition to its classical counterpart.

Our first result shows that RT-QFA$^{\circlearrowleft}$s can simulate any RT-PFA$^{\circlearrowleft}$ with small state cost, albeit with great slowdown. Note that no such relation is known between the 2KWQFA and its classical counterpart, the 2PFA, in the bounded error case.

Theorem 5.8. *Any language $L \subseteq \Sigma^*$ recognized by an n-state RT-PFA$^{\circlearrowleft}$ with error bound ϵ can be recognized by a $(2n+4)$-state RT-QFA$^{\circlearrowleft}$ with the same error bound. Moreover, if the expected runtime of the RT-PFA$^{\circlearrowleft}$ is $O(s(|w|))$, then the expected runtime of the RT-QFA$^{\circlearrowleft}$ is $O(l^{2|w|}s^2(|w|))$ for a constant $l > 1$ depending on n, where w is the input string.*

Proof. Let $\mathcal{P}$ be an n-state RT-PFA$^{\circlearrowleft}$ recognizing L with error bound ϵ. We construct a $(2n+4)$-state RT-QFA$^{\circlearrowleft}$ $\mathcal{M}$ recognizing the same language with error bound $\epsilon' \leq \epsilon$.

By adding two more states, q_a and q_r, to $\mathcal{P}$, we obtain a new RT-PFA$^{\circlearrowleft}$, $\mathcal{P}'$, where the halting of the computation in each round is postponed to the last symbol, \$, on which the overall accepting and rejecting probabilities are summed up into q_a and q_r, respectively. Therefore, for any given input string $w \in \Sigma^*$, the value of q_a and q_r are $p^a_{\mathcal{P}}(w)$ and $p^r_{\mathcal{P}}(w)$, respectively, at the end of the first round.

By using the method described in Section 4.1.2, each stochastic matrix can be

converted to a unitary one with twice the size, i.e. each transition matrix of $\mathcal{P}'$ can be converted to a $(2n+4)\times(2n+4)$-dimensional unitary matrix. These are the transition matrices of $\mathcal{M}$. The state set of $\mathcal{M}$ can be specified as follows:

i. The initial state of $\mathcal{M}$ is the state corresponding to the initial state of $\mathcal{P}$;
ii. The states corresponding to q_a and q_r are the accepting and rejecting states, q'_a and q'_r, respectively;
iii. the states corresponding to the non-halting and non-restarting states of $\mathcal{P}'$ are non-halting and non-restarting states, respectively; and,
iv. all remaining states are restarting states.

When $\mathcal{M}$ runs on input string w, the amplitudes of q'_a and q'_r, the only halting states of $\mathcal{M}$, at the end of the first round are $\left(\frac{1}{l}\right)^{|\tilde{w}|} p^a_{\mathcal{P}}(w)$ and $\left(\frac{1}{l}\right)^{|\tilde{w}|} p^r_{\mathcal{P}}(w)$, respectively, where l is set to $2n+5$ with respect to the template described in Figure 4.4. Therefore, when $w \in L$,

$$\frac{p^r_{\mathcal{M}}(w)}{p^a_{\mathcal{M}}(w)} = \frac{(p^r_{\mathcal{P}}(w))^2}{(p^a_{\mathcal{P}}(w))^2} \leq \frac{\epsilon^2}{(1-\epsilon)^2}, \tag{5.11}$$

and similarly, when $w \notin L$,

$$\frac{p^a_{\mathcal{M}}(w)}{p^r_{\mathcal{M}}(w)} = \frac{(p^a_{\mathcal{P}}(w))^2}{(p^r_{\mathcal{P}}(w))^2} \leq \frac{\epsilon^2}{(1-\epsilon)^2}. \tag{5.12}$$

By solving the equation

$$\frac{\epsilon'}{1-\epsilon'} = \frac{\epsilon^2}{(1-\epsilon)^2}, \tag{5.13}$$

we obtain

$$\epsilon' = \frac{\epsilon^2}{1-2\epsilon+2\epsilon^2} \leq \epsilon. \tag{5.14}$$

The expected runtime of $\mathcal{P}$ is

$$\frac{|\tilde{w}|}{p^a_{\mathcal{P}}(w) + p^r_{\mathcal{P}}(w)} \in O(s(|w|)), \tag{5.15}$$

and so the expected runtime of $\mathcal{M}$ is

$$(l)^{2|\tilde{w}|} \frac{|\tilde{w}|}{(p^a_{\mathcal{P}}(w))^2 + (p^r_{\mathcal{P}}(w))^2} < 3\,(l)^{2|\tilde{w}|} \left(\frac{|\tilde{w}|}{p^a_{\mathcal{P}}(w) + p^r_{\mathcal{P}}(w)} \right)^2 \in O(l^{2|w|} s^2(|w|)). \tag{5.16}$$

□

Corollary 5.9. *RT-QFA$^{\circlearrowleft}$s can recognize all regular languages with zero error.*

If the underlying QFA model of realtime QFA with restart is chosen as RT-QFA instead of RT-KWQFA, we obtain the general RT-QFA$^{\circlearrowleft}$, denoted shortly as RT-GQFA$^{\circlearrowleft}$. Thus, the accepting, rejecting, and restarting parts can easily be postponed to the end of the computation, that is, only one observation is implemented after the whole input is read. A *general* realtime quantum finite automaton with restart is a 6-tuple

$$\mathcal{M} = (Q, \Sigma, \{\mathcal{E}_{\sigma \in \tilde{\Sigma}}\}, q_1, Q_a, Q_r), \tag{5.17}$$

where all specifications are the same as RT-QFA (see Section 3.2.4) except:

- Q_r is the set of rejecting states;
- $Q^{\circlearrowleft} = Q \setminus (Q_a \cup Q_r)$ is the set of restarting states;
- $\Delta = \{a, r, \circlearrowleft\}$ with the following specifications:
  - i. "a": the computation is halted and the input is accepted,
  - ii. "r": the computation is halted and the input is rejected, and
  - iii. "$\circlearrowleft$": the computation is restarted.

  The corresponding projectors, P_a, P_r, and $P^{\circlearrowleft}$, are defined in a standard way, based on the related set of states, Q_a, Q_r, and $Q^{\circlearrowleft}$, respectively.

Theorem 5.10. *Any language $L \subseteq \Sigma^*$ recognized by an n-state RT-GQFA$^{\circlearrowleft}$ with error*

bound ϵ can be recognized by a $O(n)$-state RT-QFA$^{\circlearrowleft}$ with the same error bound. Moreover, if the expected runtime of the RT-GQFA$^{\circlearrowleft}$ is $O(s(|w|))$, then the expected runtime of the RT-QFA$^{\circlearrowleft}$ is $O(l^{2|w|}s^2(|w|))$ for a constant $l > 1$, where w is the input string.

Proof. We use almost the same idea presented in the proof of Theorem 5.8 after linearizing the computation of the given RT-GQFA$^{\circlearrowleft}$. Let $\mathcal{G} = (Q, \Sigma, \{\mathcal{E}_{\sigma\in\tilde{\Sigma}}\}, q_1, Q_a, Q_r)$ be an n-state RT-GQFA$^{\circlearrowleft}$ recognizing L with error bound ϵ. We construct a $3n^2+6$-state RT-QFA$^{\circlearrowleft}$ $\mathcal{M}$ recognizing the same language with error bound $\epsilon' \leq \epsilon$.

In order to linearize $\mathcal{G}$, we use the technique described in Lemma 3.2 and so we obtain $n^2 \times n^2$-dimensional matrices for each $\sigma \in \tilde{\Sigma}$, i.e. $A_\sigma = \sum_{i=1}^{|\mathcal{E}_\sigma|} E_{\sigma,i} \otimes E^*_{\sigma,i}$. By adding two more states, q_{n^2+1} and q_{n^2+2}, the overall accepting and rejecting probabilities are respectively summed up on them, i.e.

$$A'_{\sigma\in\Sigma\cup\{¢\}} = \left(\begin{array}{c|c} A_\sigma & 0_{n\times 2} \\ \hline 0_{2\times n} & I_{2\times 2} \end{array}\right), A'_\$ = \left(\begin{array}{c|c} 0_{n\times n} & 0_{2\times n} \\ \hline T_{2\times n} & I_{2\times 2} \end{array}\right)\left(\begin{array}{c|c} A_\$ & 0_{n\times 2} \\ \hline 0_{2\times n} & I_{2\times 2} \end{array}\right), \tag{5.18}$$

where all the entries of T are zeros except that $T[1,(i-1)n^2+i] = 1$ when $q_i \in Q_a$ and $T[2,(i-1)n^2+i] = 1$ when $q_i \in Q_r$. Let $v_0 = (1,0,\ldots,0)$ be a $(n^2+2)\times 1$-dimensional column vector. It can be easily verified that, for any $w \in \Sigma^*$,

$$v'_{|\tilde{w}|} = A'_\$ A'_{w_{|w|}} \cdots A'_{w_1} A'_¢ v_0 = (0_{n^2\times 1}, p^a_{\mathcal{G}}(w), p^r_{\mathcal{G}}(w)) \tag{5.19}$$

Based on the template given on Figure 5.1, we obtain constant l and the sets $B_{\sigma\in\tilde{\Sigma}}$ and $C_{\sigma\in\tilde{\Sigma}}$ such that the columns of the following matrix form an orthonormal set,

$$\frac{1}{l}\left(\begin{array}{c} A'_\sigma \\ \hline B_\sigma \\ \hline C_\sigma \end{array}\right). \tag{5.20}$$

The remaining part is as in the proof of Theorem 5.8. □

Let S be a finite set and $\{A_s \mid s \in S\}$ be a set of $m \times m$-dimensional matrices. We present a method in order to find two sets of $m \times m$-dimensional matrices, $\{B_s \mid s \in S\}$ and $\{C_s \mid s \in S\}$, with a generic constant l such that the columns of the following matrix form an orthonormal set

$$\frac{1}{l}\begin{pmatrix} A_s \\ \hline B_s \\ \hline C_s \end{pmatrix}, \tag{5.21}$$

for each $s \in S$. The details of the method is given below.

i. The entries of $B_{s\in S}$ and $C_{s\in S}$ are set to 0.

ii. For each $s \in S$, the entries of B_s are updated to make the columns of $\begin{pmatrix} A_s \\ \hline B_s \end{pmatrix}$ pairwise orthogonal. Specifically,

for $i = 1, \ldots, m-1$

set $b_{i,i} = 1$

for $j = i+1, \ldots, m$

set $b_{i,j}$ to some value so that the i^{th} and j^{th} columns become orthogonal

set l_s to the maximum of the lengths (norms) of the columns of $\begin{pmatrix} A_s \\ \hline B_s \end{pmatrix}$

iii. $l = \max(\{l_s \mid s \in S\})$.

iv. For each $s \in S$, the diagonal entries of C_s are updated to make the length of each column of $\begin{pmatrix} A_s \\ \hline B_s \\ \hline C_s \end{pmatrix}$ equal to l.

Figure 5.1. General template to build a unitary matrix (II)

For an automaton $\mathcal{M}$ recognizing a language L, we define the *gap function*, $g_{\mathcal{M}} : N \to [0,1]$, such that $g_{\mathcal{M}}(n)$ is the difference between the minimum acceptance probability of a member of L with length at most n and the maximum acceptance probability of a non-member of L with length at most n[20] .

Lemma 5.11. *If a language L is recognized by a RT-KWQFA $\mathcal{M}$ with positive (negative) one-sided unbounded error such that $g_{\mathcal{M}}(n) \geq c^{-n}$ for some $c > 1$, then for all $\epsilon \in (0, \frac{1}{2})$, L is recognized by some RT-QFA$^{\circlearrowleft}$ having three more states than $\mathcal{M}$ with positive (negative) one-sided error ϵ in expected time $O(\frac{1}{\epsilon}c^{|w|}|w|)$, where w is an input*

[20]The definition of $g_{\mathcal{M}}$ is due to Bertoni and Carpentieri [66], who call it the "error function."

string.

Proof. We consider the case of positive one-sided error. The adaptation to the other case is trivial. $\mathcal{M}$ is converted into a RT-QFA$^{\circlearrowleft}$ $\mathcal{M}'_\epsilon$, shortly $\mathcal{M}'$, as follows. $\mathcal{M}'$ starts by branching to two equiprobable paths, $\mathsf{path_1}$ and $\mathsf{path_2}$, at the beginning of the computation. $\mathsf{path_1}$ imitates the computation of $\mathcal{M}$, except that all reject states that appear in its subpaths are replaced by restart states. Regardless of the form of the input, $\mathsf{path_2}$ moves right with amplitude $\frac{1}{\sqrt{c}}$, (and so restarts the computation with the remaining probability,) on every input symbol. When it arrives at the right end-marker, $\mathsf{path_2}$ rejects with amplitude $\sqrt{\epsilon}$, and restarts the computation with amplitude $\sqrt{1-\epsilon}$.

When $w \notin L$,

$$p^a_{\mathcal{M}'}(w) = 0, \text{ and } p^r_{\mathcal{M}'}(w) = \frac{\epsilon}{2c^{|w|}}, \tag{5.22}$$

and so the input is rejected with probability 1. When $w \in L$,

$$p^a_{\mathcal{M}'}(w) \geq \frac{1}{2c^{|w|}}, \text{ and } p^r_{\mathcal{M}'}(w) = \frac{\epsilon}{2c^{|w|}}, \tag{5.23}$$

and so the input is accepted with error bound $\epsilon > 0$ due to Lemma 5.2, since

$$\frac{p^r_{\mathcal{M}'}(w)}{p^a_{\mathcal{M}'}(w)} \leq \epsilon. \tag{5.24}$$

Since $p^h_{\mathcal{M}'}(w)$ is always greater than $\frac{\epsilon}{2c^{|w|}}$, the expected runtime of $\mathcal{M}'_\epsilon$ is $O(\frac{1}{\epsilon}c^{|w|}|w|)$. □

Lemma 5.12. *If a language L is recognized by a RT-KWQFA $\mathcal{M}$ with positive (negative) one-sided bounded error such that $g_{\mathcal{M}}(n) \geq c^{-1}$ for some $c > 1$, then for all $\epsilon \in (0, \frac{1}{2})$, L is recognized by some RT-QFA$^{\circlearrowleft}$ having three more states than $\mathcal{M}$ with positive (negative) one-sided error ϵ in expected time $O(\frac{1}{\epsilon}c|w|)$, where w is an input string.*

Proof. The construction is almost identical to that in Lemma 5.11, except that path_2 rejects with amplitude $\sqrt{\epsilon}$, and restarts the computation with amplitude $\sqrt{1-\epsilon}$ immediately on the left end-marker, thereby causing every input to be rejected with the constant probability $\frac{\epsilon}{2c}$. Hence, the expected runtime of the RT-QFA$^{\circlearrowleft}$s turns out to be $O(\frac{1}{\epsilon}c|w|)$. □

Lemma 5.11 is a useful step towards an eventual characterization of the class of languages that are recognized with one-sided bounded error by RT-QFA$^{\circlearrowleft}$s, since full classical characterizations are known (see Section 4.2.2) for the classes of languages recognized by one-sided unbounded error by several RT-QFA models, including the RT-KWQFA.

Theorem 5.13. *For every language $L \in S^{=}_{\mathbb{Q}}$, there exists a number n such that for all error bounds $\epsilon > 0$, there exist n-state RT-QFA$^{\circlearrowleft}$s that recognize L and $\overline{L}$ with one-sided error bounded by ϵ.*

Proof. For a language L in $\mathrm{S}^{=}_{\mathbb{Q}}$, let $\mathcal{P}$ be the rational RT-PFA (i.e. each transition probability must be a rational number) associated by L. Turakainen [55] showed that there exists a constant $b > 1$ such that for any string $w \notin L$, the probability that $\mathcal{P}$ accepts w cannot be in the interval $(\frac{1}{2} - b^{-|w|}, \frac{1}{2} + b^{-|w|})$. By using the method described in Section 4.2.2, we can convert $\mathcal{P}$ to a RT-KWQFA $\mathcal{M}$ recognizing $\overline{L}$ with one-sided unbounded error, so that $\mathcal{M}$ accepts any $w \in \overline{L}$ with probability greater than $c^{-|w|}$, for a constant $c > b$. We can conclude with Lemma 5.11. □

$\mathrm{S}^{=}_{\mathbb{Q}}$ contains many well-known languages, such as L_{eq}, L_{pal}, $L_{twin} = \{wcw \mid w \in \{a,b\}^*\}$, $L_{mult} = \{x\#y\#z \mid x, y, z \text{ are natural numbers in binary notation and } x \times y = z\}$, $L_{square} = \{a^n b^{n^2} \mid n > 0\}$, $L_{power} = \{a^n b^{2^n} \mid n > 0\}$, the word problem for finitely generated free groups, and all *polynomial languages*, [67] defined as

$$\{a_1^{n_1} \cdots a_k^{n_k} b_1^{p_1(n_1,\ldots,n_k)} \cdots b_r^{p_r(n_1,\ldots,n_k)} \mid p_i(n_1,\ldots,n_k) \geq 0\}, \tag{5.25}$$

where $a_1, \ldots, a_k, b_1, \ldots, b_r$ are distinct symbols, and each p_i is a polynomial with integer coefficients. Note that Theorem 5.13 and Lemma 5.6 answer a question posed by Ambainis and Watrous [5] about whether L_{square} and L_{power} can be recognized with bounded error by 2CQFAs affirmatively.

Corollary 5.14. *The class of languages recognized by RT-QFA$^{\circlearrowleft}$s with bounded error properly contains the class of languages recognized by RT-PFA$^{\circlearrowleft}$s.*

Proof. This follows from Theorems 5.8 and 5.13, Lemma 5.4, and the fact [35,68] that L_{pal} cannot be recognized with bounded error by 2PFAs. □

Since RT-QFAs [17, 36, 69] are known to be equivalent in language recognition power to RT-PFAs, one has to consider a two-way model to demonstrate the superiority of quantum computers over classical ones. The 2CQFA is known [5] to be superior to its classical counterpart, the 2PFA, also by virtue of L_{pal}. Recall that, by Lemma 5.6, 2CQFAs can simulate RT-QFA$^{\circlearrowleft}$s easily, and we do not know of a simulation in the other direction.

5.2. Succinctness of Two-Way Models

In this section, we demonstrate several infinite families of regular languages which can be recognized with some fixed probability greater than $\frac{1}{2}$ by just tuning the transition amplitudes of a RT-QFA$^{\circlearrowleft}$ with a constant number of states, whereas the sizes of the corresponding RT-QFAs, RT-PFAs, and 2NFAs grow without bound. One of our constructions can be adapted easily to show that RT-PFA$^{\circlearrowleft}$s, (and, equivalently, 2PFAs), also possess the same advantage over those machines.

Definition 5.15. *For an alphabet Σ containing symbols a and b, and $m \in \mathbb{Z}^+$, the family of languages A_m is defined as*

$$A_m = \{ua \mid u \in \Sigma^*, |u| \leq m\}. \tag{5.26}$$

Note that Ambainis et al. [70] report that any Nayak realtime quantum finite automaton[21] that recognizes A_m with some fixed probability greater than $\frac{1}{2}$ has $2^{\Omega(m)}$ states.

Theorem 5.16. *A_m is recognized by a 6-state RT-QFA$^{\circlearrowleft}$ $\mathcal{M}_{m,\epsilon}$ for any error bound $\epsilon > 0$. Moreover, the expected runtime of $\mathcal{M}_{m,\epsilon}$ on input w is $O(\left(\frac{1}{\epsilon}\right)^{2m}|w|)$.*

Proof. Let $\mathcal{M}_{m,\epsilon}$ (shortly $\mathcal{M}$) $= \{Q, \Sigma, \delta, q_0, Q_a, Q_r, Q^{\circlearrowleft}\}$ be a RT-QFA$^{\circlearrowleft}$ with $Q_n = \{q_0, q_1\}$, $Q_a = \{A\}$, $Q_r = \{R\}$, $Q^{\circlearrowleft} = \{I_1, I_2\}$. $\mathcal{M}$ contains the transitions

$$\begin{aligned}
U_{¢}|q_0\rangle &= \epsilon|q_1\rangle + \epsilon^{\frac{2m+5}{2}}|R\rangle + \sqrt{1-\epsilon^2-\epsilon^{2m+5}}|I_1\rangle \\
U_a|q_0\rangle &= \epsilon|q_0\rangle + \sqrt{\frac{1}{2}-\epsilon^2}|I_1\rangle + \frac{1}{\sqrt{2}}|I_2\rangle \\
U_a|q_1\rangle &= \epsilon|q_0\rangle + \sqrt{\frac{1}{2}-\epsilon^2}|I_1\rangle - \frac{1}{\sqrt{2}}|I_2\rangle \\
U_{\Sigma\setminus\{a\}}|q_0\rangle &= \epsilon|q_1\rangle + \sqrt{\frac{1}{2}-\epsilon^2}|I_1\rangle + \frac{1}{\sqrt{2}}|I_2\rangle \\
U_{\Sigma\setminus\{a\}}|q_1\rangle &= \epsilon|q_1\rangle + \sqrt{\frac{1}{2}-\epsilon^2}|I_1\rangle - \frac{1}{\sqrt{2}}|I_2\rangle \\
U_{\$}|q_0\rangle &= |A\rangle \\
U_{\$}|q_1\rangle &= |R\rangle
\end{aligned}$$

and the transitions not mentioned above can be completed easily, by extending each U_σ to be unitary.

On the left end-marker, $\mathcal{M}$ rejects with probability ϵ^{2m+5}, goes on to scan the input string with amplitude ϵ, and restarts immediately with the remaining probability. States q_0 and q_1 implement the check for the regular expression $\Sigma^* a$, but the machine restarts with probability $1-\epsilon^2$ on all input symbols during this check.

If $w = u\sigma'$ for $u \in \Sigma^*$, and $\sigma' \neq a$, the input is rejected with probability 1, since

[21] This is a realtime QFA model of intermediate power, subsuming the RT-KWQFA, but strictly weaker than RT-QFA in bounded error.

$p^a_{\mathcal{M}'}(w) = 0$.

If $w = ua$ for $u \in \Sigma^*$,

$$p^a_{\mathcal{M}}(w) = \epsilon^{2|w|+2}, \quad p^r_{\mathcal{M}}(w) = \epsilon^{2m+5}. \tag{5.27}$$

Hence, if $w \in A_m$,

$$p^a_{\mathcal{M}}(w) \geq \epsilon^{2m+4}, \tag{5.28}$$

and if $w \notin A_m$,

$$p^a_{\mathcal{M}}(w) \leq \epsilon^{2m+6}. \tag{5.29}$$

In both cases, the corresponding ratio $\frac{p^r_{\mathcal{M}}(w)}{p^a_{\mathcal{M}}(w)}$ or $\frac{p^a_{\mathcal{M}}(w)}{p^r_{\mathcal{M}}(w)}$ is not greater than ϵ. Thus, by Lemma 5.2, we conclude that $\mathcal{M}$ recognizes A_m with error bounded by ϵ. Since $p^h_{\mathcal{M}}(w)$ is always greater than ϵ^{2m+5}, the expected runtime of $\mathcal{M}$ is $O(\left(\frac{1}{\epsilon}\right)^{2m}|w|)$. □

By a theorem of Rabin [15], for any fixed error bound, if a language L is recognized with bounded error by a RT-PFA with n states, then there exists a RT-DFA that recognizes L with $2^{O(n)}$ states. Parallelly, Freivalds et al. [71] note that one-way quantum finite automata with mixed states are no more than superexponentially more concise than RT-DFAs. These facts can be used to conclude that a collection of RT-PFAs (or RT-QFAs) with a fixed common number of states that recognize an infinite family of languages with a fixed common error bound less than $\frac{1}{2}$, *à la* the two-way quantum automata of Theorem 5.16, cannot exist, since that would imply the existence of a similar family of RT-DFAs of fixed size. By the same reasoning, the existence of such families of 2NFAs can also be overruled.

The reader should note that there exists a bounded-error RT-PFA$^{\circlearrowleft}$ (and there-

fore, a 2PFA[22] ,) for A_m, which one can obtain simply by replacing each transition amplitude of 1QFA$^{\circlearrowleft}$ $\mathcal{M}_{m,\epsilon}$ defined in Theorem 5.16 by the square of its modulus. This establishes the fact that 2PFAs also possess the succinctness advantage discussed above over RT-PFAs, RT-QFAs and RT-NFAs.

We proceed to present two more examples.

Definition 5.17. *For $m \in \mathbb{Z}^+$, the language family $B_m \subseteq \{a\}^*$ is defined as*

$$B_m = \{a^i \mid i \mod (m) \equiv 0\}. \tag{5.30}$$

Theorem 5.18. *For any error bound $\epsilon > 0$, there exists a 7-state RT-QFA$^{\circlearrowleft}$ $\mathcal{M}_{m,\epsilon}$ which accepts any $w \in B_m$ with certainty, and rejects any $w \notin B_m$ with probability at least $1 - \epsilon$. Moreover, the expected runtime of $\mathcal{M}_{m,\epsilon}$ on w is $O\left(\frac{1}{\epsilon}\sin^{-2}(\frac{\pi}{m})|w|\right)$.*

Proof. We construct a 4-state RT-KWQFA recognizing $\overline{B_m}$ with positive one-sided bounded error, as described in [48]. Let $\mathcal{M}_m = (Q, \Sigma, \delta, q_0, Q_a, Q_r)$ be RT-KWQFA with $Q_n = \{q_0, q_1\}$, $Q_a = \{A\}$, and $Q_r = \{R\}$. $\mathcal{M}_m$ contains the transitions

$$\begin{aligned}
U_{¢}|q_0\rangle &= |q_0\rangle \\
U_a|q_0\rangle &= \cos(\frac{\pi}{m})|q_0\rangle + \sin(\frac{\pi}{m})|q_1\rangle \\
U_a|q_1\rangle &= -\sin(\frac{\pi}{m})|q_0\rangle + \cos(\frac{\pi}{m})|q_1\rangle \\
U_{\$}|q_0\rangle &= |R\rangle \\
U_{\$}|q_1\rangle &= |A\rangle
\end{aligned}$$

and the transition amplitudes not listed above are filled in to satisfy unitarity. $\mathcal{M}_m$ begins computation at the $|q_0\rangle$-axis, and performs a rotation by angle $\frac{\pi}{m}$ in the $|q_0\rangle$-$|q_1\rangle$ plane for each a it reads. Therefore, the value of the gap function, $g_{\mathcal{M}_m}$, is not less than $\sin^2(\frac{\pi}{m})$ for $|w| > 0$. By Lemma 5.12, there exists a 7-state RT-QFA$^{\circlearrowleft}$ $\mathcal{M}_{m,\epsilon}$ recognizing $\overline{B_m}$ with positive one-sided bounded error and whose expected runtime is

[22] See Section 5.1.3 for an examination of the relationship between the computational powers of the RT-PFA$^{\circlearrowleft}$ and the 2PFA.

$O\left(\frac{1}{\epsilon}\sin^{-2}(\frac{\pi}{m})|w|\right)$. By swapping the accepting and rejecting states of $\mathcal{M}_{m,\epsilon}$, we can get the desired machine. □

Definition 5.19. *For an alphabet Σ, and $m \in \mathbb{Z}^+$, the language family C_m is defined as*

$$C_m = \{w \in \Sigma^* \mid |w| = m\}. \tag{5.31}$$

Theorem 5.20. *For any error bound $\epsilon > 0$, there exists a 7-state RT-QFA$^{\circlearrowleft}$ $\mathcal{M}_{m,\epsilon}$ which accepts any $w \in C_m$ with certainty, and rejects any $w \notin C_m$ with probability at least $1 - \epsilon$. Moreover, the expected runtime of $\mathcal{M}_{m,\epsilon}$ on w is $O(\frac{1}{\epsilon}2^m|w|)$.*

Proof. We construct a 4-state RT-KWQFA recognizing $\overline{C_m}$ with positive one-sided unbounded error. Let $\mathcal{M}_m = (Q, \Sigma, \delta, q_0, Q_a, Q_r)$ be RT-KWQFA with $Q_n = \{q_0, q_1\}$, $Q_a = \{A\}$, and $Q_r = \{R\}$. $\mathcal{M}_m$ contains the transitions

$$\begin{aligned}
U_{¢}|q_0\rangle &= \frac{1}{\sqrt{2}}|q_0\rangle + \left(\frac{1}{\sqrt{2}}\right)^{m+1}|q_1\rangle + \sqrt{\frac{1}{2} - \left(\frac{1}{2}\right)^{m+1}}|R\rangle \\
U_{\sigma\in\Sigma}|q_0\rangle &= \frac{1}{\sqrt{2}}|q_0\rangle + \frac{1}{\sqrt{2}}|R\rangle \\
U_{\sigma\in\Sigma}|q_1\rangle &= |q_1\rangle \\
U_{\$}|q_0\rangle &= \frac{1}{\sqrt{2}}|A\rangle + \frac{1}{\sqrt{2}}|R\rangle \\
U_{\$}|q_1\rangle &= -\frac{1}{\sqrt{2}}|A\rangle + \frac{1}{\sqrt{2}}|R\rangle
\end{aligned}$$

with the amplitudes of the transitions not mentioned above filled in to ensure unitarity.

$\mathcal{M}_m$ encodes the length of the input string in the amplitude of state q_0, which equals $\left(\frac{1}{\sqrt{2}}\right)^{|w|+1}$ just before the processing of the right end-marker. The desired length m is "hardwired" into the amplitudes of q_1. For a given input string $w \in \Sigma^*$, if $w \in C_m$, then the amplitudes of states q_0 and q_1 are equal, and the QFT [4] performed on the right end-marker sets the amplitude of A to 0. Therefore, w is rejected with certainty.

If $w \in \overline{C_m}$, then the accepting probability is equal to

$$\left(\left(\frac{1}{\sqrt{2}}\right)^{|w|+2} - \left(\frac{1}{\sqrt{2}}\right)^{m+2}\right)^2 \tag{5.32}$$

and it is minimized when $|w| = m+1$, which gives us the inequality

$$g_{\mathcal{M}_m}(w) > \left(\frac{1}{2}\right)^{m+6}. \tag{5.33}$$

By Lemma 5.12, there exists a 7-state RT-QFA$^{\circlearrowleft}$ $\mathcal{M}_{m,\epsilon}$ recognizing $\overline{C_m}$ with positive one-sided bounded error and whose expected runtime is $O\left(\frac{1}{\epsilon}2^m|w|\right)$. By swapping the accepting and rejecting states of $\mathcal{M}_{m,\epsilon}$, we can get the desired machine. □

Note that, unlike what we had with Theorem 5.16, the QFAs of Theorems 5.18 and 5.20 cannot be converted so easily to 2PFAs. In fact, we can prove that there exist no 2PFA families of fixed size which recognize B_m and C_m with fixed one-sided error less than $\frac{1}{2}$, like those QFAs: Assume that such a 2PFA family exists. Switch the accept and reject states to obtain a family for the complements of the languages. The 2PFAs thus obtained operate with cutpoint 0. Obtain an equivalent 2NFA with the same number of states by converting all transitions with nonzero weight to nondeterministic transitions. But there are only finitely many 2NFAs of this size, meaning that they cannot recognize our infinite family of languages.

5.3. Probability Amplification

Many automaton descriptions in this thesis, and elsewhere in the theory of probabilistic and quantum automata, describe not a single algorithm, but a general template which one can use for building a machine M_ϵ that operates with a desired error bound ϵ. The dependences of the runtime and number of states of M_ϵ on $\frac{1}{\epsilon}$ are measures of the complexity of the probability amplification process involved in the construction method used. Viewed as such, the constructions described in the theorems in Section 5.2 are maximally efficient in terms of the state cost, with no dependence on the er-

ror bound. In this section, we present improvements over previous results about the efficiency of probability amplification in 2QFAs.

5.3.1. Improved Algorithms for UPAL Language

In classical computation, one only needs to sequence $O(\log(\frac{1}{\epsilon}))$ identical copies of a given probabilistic automaton with one sided error $p < 1$ to run on the same input in order to obtain a machine with error bound ϵ. Yakaryılmaz and Say [72] noted that this method of probability amplification does not yield efficient results for 2KWQFAs; the number of machine copies required to reduce the error to ϵ can be as high as $(\frac{1}{\epsilon})^2$. The most succinct 2KWQFAs for L_{upal}, $\{a^n b^n \mid n > 0\}$, produced by alternative methods developed in [72] have $O(\log^2(\frac{1}{\epsilon}) \log\log(\frac{1}{\epsilon}))$ states, and runtime linear in the size of the input w. In Section 5.3.2, we present a construction which yields (exponential time) RT-QFA$^{\circlearrowleft}$s that recognize L_{upal} within any desired error bound ϵ, with no dependence of the state set size on ϵ. Ambainis and Watrous [5] present a method which can be used to build 2QCFAs that recognize L_{upal} also with constant state set size, where the "tuning" of the automaton for a particular error bound is achieved by setting some transition amplitudes appropriately, and the expected runtime of those machines is $O(|w|^4)$. We now show that the 2QFA$^{\circlearrowleft}$ formalism allows more efficient probability amplification.

Theorem 5.21. *There exists a constant n, such that, for any $\epsilon > 0$, an n-state 2QFA$^{\circlearrowleft}$ which recognizes L_{upal} with one-sided error bound ϵ within $O(\frac{1}{\epsilon}|w|)$ expected runtime can be constructed, where w is the input string.*

Proof. We start with Kondacs and Watrous' original 2KWQFA [4] M_N, which recognizes L_{upal} with one-sided error $\frac{1}{N}$, for any integer $N > 1$. After a deterministic test for membership of a^*b^*, M_N branches to N computational paths, each of which perform a QFT at the end of the computation. Set $N = 2$. M_2 accepts all members of L_{eq} with probability 1. Non-members of L_{eq} are rejected with probability at least $\frac{1}{2}$. We convert

M_2 to a 2QFA$^{\circlearrowleft}$ $\mathcal{M}'_\epsilon$ by changing the target states of the QFT as follows:

$$\mathsf{path}_1 \rightarrow \frac{1}{\sqrt{2}}|\text{Reject}\rangle + \sqrt{\frac{\epsilon}{2}}|\text{Accept}\rangle + \sqrt{\frac{1-\epsilon}{2}}|\text{Restart}\rangle \tag{5.34}$$

$$\mathsf{path}_2 \rightarrow -\frac{1}{\sqrt{2}}|\text{Reject}\rangle + \sqrt{\frac{\epsilon}{2}}|\text{Accept}\rangle + \sqrt{\frac{1-\epsilon}{2}}|\text{Restart}\rangle \tag{5.35}$$

where the amplitude of each path is $\frac{1}{\sqrt{2}}$. For a given input $w \in \Sigma^*$,

i. if w is not of the form a^*b^*, then $p^r_{\mathcal{M}'}(w) = 1$;
ii. if w is of the form a^*b^* and $w \notin L$, then $p^r_{\mathcal{M}'}(w) = \frac{1}{2}$, and $p^a_{\mathcal{M}'}(w) = \frac{\epsilon}{2}$;
iii. if $w \in L$, then $p^r_{\mathcal{M}'}(w) = 0$ and $p^a_{\mathcal{M}'}(w) = \epsilon$.

It is easily seen that the error is one-sided. Since $\frac{p^a_{\mathcal{M}'}(w)}{p^r_{\mathcal{M}'}(w)} = \epsilon$, we can conclude with Lemma 5.2. Moreover, the minimum halting probability occurs in the third case above, and so the expected runtime of $\mathcal{M}'_\epsilon$ is $O(\frac{1}{\epsilon}|w|)$. □

Theorem 5.22. *For any $\epsilon \in (0, \frac{1}{2})$, there exists a 2QFA$^{\curvearrowleft}$ with $O(\log(\frac{1}{\epsilon}))$ states that recognizes L_{upal} with one-sided error bound ϵ in $O(\log(\frac{1}{\epsilon})|w|)$ steps, where w is the input string.*

Proof. Let M_2 be the 2KWQFA recognizing L_{upal} with one-sided error bound $\frac{1}{2}$ mentioned in the proof of Theorem 5.21. Then, a 2QFA$^{\curvearrowleft}$ that is constructed by sequentially connecting $O(\log(\frac{1}{\epsilon}))$ copies of M_2, so that the input is accepted only if it is accepted by all the copies, and rejected otherwise, can recognize L_{upal} with one-sided error bound ϵ. □

5.3.2. A Realtime Quantum Finite Automata with Restart Algorithm for UPAL Language

Theorem 5.23. *For any $\epsilon > 0$, there exists a 15-state RT-QFA$^{\circlearrowleft}$ $\mathcal{M}_\epsilon$, which accepts any $w \in L_{upal}$ with certainty, and rejects any $w \notin L_{upal}$ with probability at least $1-\epsilon$. Moreover, the expected runtime of $\mathcal{M}_\epsilon$ on w is $O(\frac{1}{\epsilon}(2\sqrt{2})^{|w|}|w|)$.*

Proof. We construct a 12-state RT-KWQFA recognizing $\overline{L_{eq}}$ with positive one-sided unbounded error. Let $\mathcal{M} = (Q, \Sigma, \delta, q_0, Q_a, Q_r)$ be RT-KWQFA with $Q_a = \{A_1, A_2, A_3\}$, $Q_{rej} = \{R_1, R_2, R_3\}$, and $Q_n = \{p_0, p_1, p_2, q_0, q_1, q_2\}$. The transition function of $\mathcal{M}$ is shown in Figure 5.2. As before, we assume that the transitions not specified in the figure are filled in to ensure that the U_σ are unitary.

Paths	$U_{¢}, U_a$	U_b	$U_{\$}$
	$U_{¢}\|q_0\rangle = \frac{1}{\sqrt{2}}\|p_0\rangle + \frac{1}{\sqrt{2}}\|q_0\rangle$		
path_1	$U_a\|p_0\rangle = \frac{1}{2}\|p_1\rangle + \frac{1}{2}\|R_1\rangle + \frac{1}{\sqrt{2}}\|R_2\rangle$	$U_b\|p_0\rangle = \|A_1\rangle$	$U_{\$}\|p_0\rangle = \|R_1\rangle$
	$U_a\|p_1\rangle = \frac{1}{2}\|p_1\rangle + \frac{1}{2}\|R_1\rangle - \frac{1}{\sqrt{2}}\|R_2\rangle$	$U_b\|p_1\rangle = \frac{1}{\sqrt{2}}\|p_2\rangle + \frac{1}{\sqrt{2}}\|R_1\rangle$	$U_{\$}\|p_1\rangle = \|A_1\rangle$
	$U_a\|p_2\rangle = \|A_1\rangle$	$U_b\|p_2\rangle = \frac{1}{\sqrt{2}}\|p_2\rangle - \frac{1}{\sqrt{2}}\|R_1\rangle$	$U_{\$}\|p_2\rangle = \frac{1}{\sqrt{2}}\|R_2\rangle + \frac{1}{\sqrt{2}}\|A_2\rangle$
path_2	$U_a\|q_0\rangle = \frac{1}{\sqrt{2}}\|q_1\rangle + \frac{1}{\sqrt{2}}\|R_3\rangle$	$U_b\|q_0\rangle = \|A_2\rangle$	$U_{\$}\|q_0\rangle = \|R_3\rangle$
	$U_a\|q_1\rangle = \frac{1}{\sqrt{2}}\|q_1\rangle - \frac{1}{\sqrt{2}}\|R_3\rangle$	$U_b\|q_1\rangle = \frac{1}{2}\|q_2\rangle + \frac{1}{2}\|R_2\rangle + \frac{1}{\sqrt{2}}\|R_3\rangle$	$U_{\$}\|q_1\rangle = \|A_3\rangle$
	$U_a\|q_2\rangle = \|A_2\rangle$	$U_b\|q_2\rangle = \frac{1}{2}\|q_2\rangle + \frac{1}{2}\|R_2\rangle - \frac{1}{\sqrt{2}}\|R_3\rangle$	$U_{\$}\|q_2\rangle = \frac{1}{\sqrt{2}}\|R_2\rangle - \frac{1}{\sqrt{2}}\|A_2\rangle$

Figure 5.2. Specification of the transition function of the RT-KWQFA presented in the proof of Theorem 5.23

As seen in the figure, $\mathcal{M}$ branches to two paths on the left end-marker. Both paths reject immediately if the input $w \in \{a,b\}^*$ is the empty string, and accept with nonzero probability, say α, if it is of the form $(\{a,b\}^* \setminus a^*b^*) \cup a^+ \cup b^+$. Otherwise, $w = a^m b^n$ $(m, n > 0)$, and the amplitudes of the paths just before the transition associated with the right end-marker in the first round are as follows:

- State p_2 has amplitude $\frac{1}{\sqrt{2}}(\frac{1}{2})^m(\frac{1}{\sqrt{2}})^n$,
- state q_2 has amplitude $\frac{1}{\sqrt{2}}(\frac{1}{\sqrt{2}})^m(\frac{1}{2})^n$.

If $m = n$, then the accepting probability is zero. If $m \neq n$ (assume without loss of

generality that $m = n + d$ for some $d \in \mathbb{Z}^+$), then the accepting probability is equal to

$$\left(\frac{1}{2}\right)^{m+n+1}\left(\left(\frac{1}{\sqrt{2}}\right)^{m}-\left(\frac{1}{\sqrt{2}}\right)^{n}\right)^{2}=\underbrace{\left(\frac{1}{2}\right)^{m+2n+1}}_{>\left(\frac{1}{2}\right)^{\frac{3|w|}{2}+1}}\underbrace{\left(1-\left(\frac{1}{\sqrt{2}}\right)^{d-2}+\left(\frac{1}{2}\right)^{d}\right)}_{>\frac{1}{16}} \tag{5.36}$$

Since α is always greater than this value,

$$g_{\mathcal{M}}(|w|) > \left(\frac{1}{2}\right)^{\frac{3|w|}{2}+5}, \tag{5.37}$$

for $|w| > 0$. By Lemma 5.11, there exists a 15-state RT-QFA$^{\circlearrowleft}$ $\mathcal{M}_\epsilon$ recognizing $\overline{L_{upal}}$ with positive one-sided bounded error and whose expected runtime is $O(\frac{1}{\epsilon}(2\sqrt{2})^{|w|}|w|)$. By swapping accepting and rejecting states of $\mathcal{M}_m$, we can get the desired machine. □

5.3.3. An Improved Algorithm for PAL Language

Ambainis and Watrous [5] present a 2CQFA construction which decides L_{pal} in expected time $O(\left(\frac{1}{\epsilon}\right)^{|w|}|w|)$ with error bounded by $\epsilon > 0$, where w is the input string. (Watrous [6] describes a 2KWQFA which accepts all members of the complement of L_{pal} with probability 1, and fails to halt for all palindromes; it is not known if 2KWQFAs can recognize this language by halting for all inputs.) We now present a RT-QFA$^{\circlearrowleft}$ construction, which, by Lemma 5.6, can be adapted to yield 2CQFAs with the same complexity, which reduces the dependence of the Ambainis-Watrous method on the desired error bound considerably.

Theorem 5.24. *For any $\epsilon > 0$, there exists a 15-state RT-QFA$^{\circlearrowleft}$ $\mathcal{M}_\epsilon$ which accepts any $w \in L_{pal}$ with certainty, and rejects any $w \notin L_{pal}$ with probability at least $1 - \epsilon$. Moreover, the expected runtime of $\mathcal{M}_\epsilon$ on w is $O(\frac{1}{\epsilon}3^{|w|}|w|)$.*

Proof. We first construct a modified version of the RT-KWQFA algorithm of Lāce et al. [11] for recognizing the nonpalindrome language. The idea behind the construction is that we encode both the input string and its reverse into the amplitudes of two of

the states of the machine, and then perform a substraction between these amplitudes using the QFT [11]. If the input is not a palindrome, the two amplitudes do not cancel each other completely, and the nonzero difference is transferred to an accept state. Otherwise, the accepting probability is zero.

Let $\mathcal{M} = (Q, \Sigma, \delta, q_0, Q_a, Q_r)$ be RT-KWQFA with $Q_n = \{p_1, p_2, q_0, q_1, q_2, q_3\}$, $Q_a = \{A\}$, $Q_r = \{R_i \mid 1 \leq i \leq 5\}$. The transition function of $\mathcal{M}$ is shown in Figure 5.3. As before, we assume that the transitions not specified in the figure are filled in to ensure that the U_σ are unitary.

Paths	$U_¢, U_a$	U_b
	$U_¢\|q_0\rangle = \frac{1}{\sqrt{2}}\|p_1\rangle + \frac{1}{\sqrt{2}}\|q_1\rangle$	
path_1	$U_a\|p_1\rangle = \sqrt{\frac{2}{3}}\|p_1\rangle - \frac{1}{\sqrt{3}}\|R_1\rangle$ $U_a\|p_2\rangle = \frac{1}{\sqrt{6}}\|p_1\rangle + \frac{1}{\sqrt{6}}\|p_2\rangle + \frac{1}{\sqrt{3}}\|R_1\rangle + \frac{1}{\sqrt{3}}\|R_2\rangle$	$U_b\|p_1\rangle = \frac{1}{\sqrt{6}}\|p_1\rangle + \frac{1}{\sqrt{6}}\|p_2\rangle + \frac{1}{\sqrt{3}}\|R_1\rangle + \frac{1}{\sqrt{3}}\|R_2\rangle$ $U_b\|p_2\rangle = \sqrt{\frac{2}{3}}\|p_2\rangle - \frac{1}{\sqrt{3}}\|R_1\rangle$
path_2	$U_a\|q_1\rangle = \frac{1}{\sqrt{6}}\|q_1\rangle + \frac{1}{\sqrt{6}}\|q_3\rangle - \frac{1}{\sqrt{3}}\|R_3\rangle + \frac{1}{\sqrt{3}}\|R_4\rangle$ $U_a\|q_2\rangle = \sqrt{\frac{2}{3}}\|q_2\rangle + \frac{1}{\sqrt{3}}\|R_5\rangle$ $U_a\|q_3\rangle = \sqrt{\frac{2}{3}}\|q_3\rangle + \frac{1}{\sqrt{3}}\|R_3\rangle$	$U_b\|q_1\rangle = \frac{1}{\sqrt{6}}\|q_1\rangle + \frac{1}{\sqrt{6}}\|q_2\rangle - \frac{1}{\sqrt{3}}\|R_3\rangle + \frac{1}{\sqrt{3}}\|R_4\rangle$ $U_b\|q_2\rangle = \sqrt{\frac{2}{3}}\|q_2\rangle + \frac{1}{\sqrt{3}}\|R_3\rangle$ $U_b\|q_3\rangle = \sqrt{\frac{2}{3}}\|q_3\rangle + \frac{1}{\sqrt{3}}\|R_5\rangle$
	$U_\$$	
path_1	$U_\$\|p_1\rangle = \|R_1\rangle$ $U_\$\|p_2\rangle = \frac{1}{\sqrt{2}}\|A\rangle + \frac{1}{\sqrt{2}}\|R_2\rangle$	
path_2	$U_\$\|q_1\rangle = \|R_3\rangle$ $U_\$\|q_2\rangle = -\frac{1}{\sqrt{2}}\|A\rangle + \frac{1}{\sqrt{2}}\|R_2\rangle$ $U_\$\|q_3\rangle = \|R_4\rangle$	

Figure 5.3. Specification of the transition function of the RT-KWQFA presented in the proof of Theorem 5.24

path_2 and path_1 encode the input string and its reverse [15,29] into the amplitudes of states q_2 and p_2, respectively. If the input is $w = w_1 w_2 \cdots w_l$, then the values of these amplitudes just before the transition associated with the right end-marker in the first round are as follows:

- State p_2 has amplitude $\frac{1}{\sqrt{2}} \left(\sqrt{\frac{2}{3}}\right)^{|w|} (0.w_l w_{l-1} \cdots w_1)_2$, and

- state q_2 has amplitude $\frac{1}{\sqrt{2}}\left(\sqrt{\frac{2}{3}}\right)^{|w|}(0.w_1w_2\cdots w_l)_2$.

The factor of $\sqrt{\frac{2}{3}}$ is due to the "loss" of amplitude necessitated by the fact that the originally non-unitary encoding matrices of [15, 29] have to be "embedded" in a unitary matrix [17, 23]. Note that the symbols a and b are encoded by 0 and 1, respectively.

If $w \in L_{pal}$, the acceptance probability is zero. If $w \in \overline{L_{pal}}$, the acceptance probability is minimized by strings which are almost palindromes, except for a single defect in the middle, that is, when $|w| = 2k$ for $k \in \mathbb{Z}^+$, $w_i = w_{2k-i+1}$, where $1 \leq i \leq k-1$, and $w_k \neq w_{k+1}$, so,

$$g_{\mathcal{M}}(w) \geq \frac{1}{8}\left(\frac{1}{3}\right)^{|w|}. \tag{5.38}$$

By Lemma 5.11, there exists a 15-state RT-QFA$^{\circlearrowleft}$ $\mathcal{M}_\epsilon$ recognizing $\overline{L_{pal}}$ with positive one-sided bounded error, whose expected runtime is $O(\frac{1}{\epsilon}3^{|w|}|w|)$. By swapping accepting and rejecting states of $\mathcal{M}_m$, we can get the desired machine. □

Note that the technique used in the proof above can be extended easily to handle bigger input alphabets by using the matrices defined on Page 169 of [29], and the method of simulating stochastic matrices by unitary matrices described in Sections 4.1.2 and 4.2.2.

5.4. Probabilistic and Quantum Automata with Postselection

In this section, we define the realtime finite automaton with postselection (RT-PostFA) in the spirit of Aaronson [10]. In fact, there is no difference between a standard finite automaton model and its counterpart with postselection in the processing of the input except for the final decision. Instead of accepting states, RT-PostFAs have a set of *postselection states*, denoted as Q_p, which is the union of two disjoint subsets Q_{pa} and Q_{pr}, the *accepting* and *rejecting postselection states*, and have the capability of discarding all computation branches except the ones belonging to Q_p at the end, on

which a normalization is performed, and the output is given. Therefore, the probability of being in a postselection state at the end must be nonzero.

5.4.1. Definitions

The acceptance and rejection probabilities of a machine, say $\mathcal{M}$, on a given input, say $w \in \Sigma^*$, in postselection are denoted as $p^a_{\mathcal{M}}(w)$ and $p^r_{\mathcal{M}}(w)$, respectively. Thus, by normalizing these probabilities, we obtain

$$f^a_{\mathcal{P}}(w) = \frac{p^a_{\mathcal{P}}(w)}{p^a_{\mathcal{P}}(w) + p^r_{\mathcal{P}}(w)}, \tag{5.39}$$

and

$$f^r_{\mathcal{P}}(w) = \frac{p^r_{\mathcal{P}}(w)}{p^a_{\mathcal{P}}(w) + p^r_{\mathcal{P}}(w)}. \tag{5.40}$$

A RT-PFA with postselection (RT-PostPFA) is a 5-tuple

$$\mathcal{P} = (Q, \Sigma, \{A_{\sigma \in \tilde{\Sigma}}\}, q_1, Q_p), \tag{5.41}$$

satisfying that for each input string $w \in \Sigma^*$,

$$\sum_{q_i \in Q_p} v_{|\tilde{w}|}[i] > 0. \tag{5.42}$$

The acceptance and rejection probabilities of $\mathcal{P}$ on $w \in \Sigma^*$ in postselection are defined respectively as

$$p^a_{\mathcal{P}}(w) = \sum_{q_i \in Q_{pa}} v_{|\tilde{w}|}[i] \tag{5.43}$$

and

$$p^r_{\mathcal{P}}(w) = \sum_{q_i \in Q_{pr}} v_{|\tilde{w}|}[i]. \tag{5.44}$$

We call the class of languages recognized by RT-PostPFAs with bounded error *PostS* (post-stochastic languages).

A RT-QFA with postselection (RT-PostQFA) is a 5-tuple

$$\mathcal{M} = (Q, \Sigma, \{\mathcal{E}_{\sigma \in \tilde{\Sigma}}\}, q_1, Q_p), \tag{5.45}$$

satisfying that for each input string $w \in \Sigma^*$,

$$tr(P_p \rho_{|\tilde{w}|}) > 0, \tag{5.46}$$

where P_p is the projector defined as

$$P_p = \sum_{q \in Q_p} |q\rangle\langle q|. \tag{5.47}$$

Additionally we define projectors P_{pa} and P_{pr} as follows:

$$P_{pa} = \sum_{q \in Q_{pa}} |q\rangle\langle q| \tag{5.48}$$

and

$$P_{pr} = \sum_{q \in Q_{pr}} |q\rangle\langle q|. \tag{5.49}$$

The acceptance and rejection probabilities of $\mathcal{M}$ on $w \in \Sigma^*$ in postselection are defined

as

$$p^a_{\mathcal{M}}(w) = tr(P_{pa}\rho_{|\tilde{w}|}) \tag{5.50}$$

and

$$p^r_{\mathcal{M}}(w) = tr(P_{pr}\rho_{|\tilde{w}|}). \tag{5.51}$$

We call the class of languages recognized by RT-PostQFAs with bounded error *PostQAL* (post-quantum automaton languages).

The error bound of a given postselection machine can be improved by performing a tensor product of the machine with itself as many times as required. Specifically, if we combine k copies of a machine with postselection state set $Q_{pa} \cup Q_{pr}$, the new accepting and rejecting postselection state sets can be chosen as

$$Q'_{pa} = \underbrace{Q_{pa} \times \cdots \times Q_{pa}}_{k \text{ times}} \tag{5.52}$$

and

$$Q'_{pr} = \underbrace{Q_{pr} \times \cdots \times Q_{pr}}_{k \text{ times}}, \tag{5.53}$$

respectively.

Lemma 5.25. *If L is recognized by RT-PostQFA (resp., RT-PostPFA) $\mathcal{M}$ with error bound $\epsilon \in (0, \frac{1}{2})$, then there exists a RT-PostQFA (resp., RT-PostPFA), say $\mathcal{M}'$, recognizing L with error bound ϵ^2.*

Proof. We give a proof for RT-PostQFAs and it can easily be extended for RT-PostPFAs. M' can be obtained by tensoring k copies of $\mathcal{M}$, where the new accepting (resp., rejecting) postselection states, Q'_{pa} (resp., Q'_{pr}), are $\otimes_{i=1}^{k} Q_{pa}$ (resp., $\otimes_{i=1}^{k} Q_{pa}$), where Q_{pa}

(resp., Q_{pr}) are accepting postselection states of $\mathcal{M}$.

Let $\rho_{\tilde{w}}$ and $\rho'_{\tilde{w}}$ be the respectively density matrices of $\mathcal{M}$ and $\mathcal{M}'$ after reading $\tilde{w}$ for a given input string $w \in \Sigma^*$. By definition, we have

$$p^a_{\mathcal{M}}(w) = \sum_{q_i \in Q_{pa}} \rho_{\tilde{w}}[i,i], \quad p^a_{\mathcal{M}'}(w) = \sum_{q_{i'} \in Q'_{pa}} \rho_{\tilde{w}}[i',i'] \tag{5.54}$$

and

$$p^r_{\mathcal{M}}(w) = \sum_{q_i \in Q_{pr}} \rho_{\tilde{w}}[i,i], \quad p^r_{\mathcal{M}'}(w) = \sum_{q_{i'} \in Q'_{pr}} \rho_{\tilde{w}}[i',i']. \tag{5.55}$$

By using the equality $\rho'_{\tilde{w}} = \otimes^k_{i=1}\rho_{\tilde{w}}$, the following can be obtained after a straightforward calculation:

$$p^a_{\mathcal{M}'}(w) = \left(p^a_{\mathcal{M}}(w)\right)^k \tag{5.56}$$

and

$$p^r_{\mathcal{M}'}(w) = \left(p^r_{\mathcal{M}}(w)\right)^k. \tag{5.57}$$

We examine the case of $w \in L$ (the case $w \notin L$ is symmetric). Since L is recognized by $\mathcal{M}$ with error bound ϵ, we have

$$\frac{p^r_{\mathcal{M}}(w)}{p^a_{\mathcal{M}}(w)} \leq \frac{\epsilon}{1-\epsilon}. \tag{5.58}$$

If L is recognized by $\mathcal{M}'$ with error bound ϵ^2, we must have

$$\frac{p^r_{\mathcal{M}'}(w)}{p^a_{\mathcal{M}'}(w)} \leq \frac{\epsilon^2}{1-\epsilon^2}. \tag{5.59}$$

Thus, any k satisfying the following inequality provides the desired machine $\mathcal{M}'$:

$$\left(\frac{\epsilon}{1-\epsilon}\right)^k \leq \frac{\epsilon^2}{1-\epsilon^2} \tag{5.60}$$

due to the fact that

$$\frac{p^r_{\mathcal{M}'}(w)}{p^a_{\mathcal{M}'}(w)} = \left(\frac{p^r_{\mathcal{M}}(w)}{p^a_{\mathcal{M}}(w)}\right)^k. \tag{5.61}$$

By solving Equation 5.61, we can get

$$k = 1 + \left\lceil \frac{\log\left(\frac{1}{\epsilon}+1\right)}{\log\left(\frac{1}{\epsilon}-1\right)} \right\rceil. \tag{5.62}$$

Therefore, for any $0 < \epsilon < \frac{1}{2}$, we can find a value for k. □

Corollary 5.26. *If L is recognized by RT-PostQFA (resp., RT-PostPFA) $\mathcal{M}$ with error bound $0 < \epsilon < \frac{1}{2}$, then there exists a RT-PostQFA (resp., RT-PostPFA), say $\mathcal{M}'$, recognizing L with error bound $\epsilon' < \epsilon$ such that ϵ' can be arbitrarily close to 0.*

5.4.2. Characterization of Realtime Postselection Automata

Theorem 5.27. *The classes of languages recognized by RT-PFA$^{\circlearrowleft}$ and RT-QFA$^{\circlearrowleft}$ with bounded error are identical to PostS and PostQAL, respectively.*

Proof. As shown in Theorem 5.10, the computational power of RT-GQFA$^{\circlearrowleft}$ and RT-QFA$^{\circlearrowleft}$ are identical. Therefore, we assume RT-GQFA$^{\circlearrowleft}$ as the restart machine in the remaining part of the proof. For a given RT-PostFA, we obtain a machine with restart by converting Q_{pa} and Q_{pr} to respectively the accepting and rejecting states, and restarting computation at the end of the input in the cases where the original machine halts in a state not in Q_p. For a given machine with restart, (we assume the computation is restarted and halted only at the end of the input,) we obtain a RT-PostFA by taking the accepting and rejecting states of the original machine as the members

of Q_{pa} and Q_{pr}, respectively, and converting the remaining states to nonpostselection states. □

Corollary 5.28. *PostQAL and PostS are subsets of the class of the languages recognized by 2QFAs and 2PFAs, respectively, with bounded error.*

Corollary 5.29. *L_{pal} is a member of PostQAL but not PostS.*

Proof. $L_{pal} \in$ PostQAL since there is a RT-QFA$^{\circlearrowleft}$ algorithm for L_{pal} (See 5.1.4 and [65]). However, L_{pal} cannot be recognized with bounded error even by 2PFAs [68]. □

Theorem 5.30. *PostQAL and PostS are closed under complementation, union, and intersection.*

Proof. If a language is recognized by a RT-PostFA with bounded error, by swapping the accepting and rejecting postselection states, we obtain a new RT-PostFA recognizing the complement of the language with bounded error. Therefore, both classes are closed under complementation.

Let L_1 and L_2 be members of PostQAL (resp., PostS). Then, there are two RT-PostQFAs (resp., RT-PostPFAs) $\mathcal{P}_1$ and $\mathcal{P}_2$ recognizing L_1 and L_2 with error bound $\epsilon < \frac{3}{4}$, respectively. Moreover, let Q_{pa_1} and Q_{pr_1} (resp., Q_{pa_2} and Q_{pr_2}) represent the sets of the accepting and rejecting postselection states of $\mathcal{P}_1$ (resp., $\mathcal{P}_2$), respectively, and let $Q_{p_1} = Q_{pa_1} \cup Q_{pr_1}$ and $Q_{p_2} = Q_{pa_2} \cup Q_{pr_2}$. By tensoring $\mathcal{P}_1$ and $\mathcal{P}_2$, we obtain two new machines, say $\mathcal{M}_1$ and $\mathcal{M}_2$, such that

- the sets of the accepting and rejecting postselection states of $\mathcal{M}_1$ is

$$Q_{p_1} \otimes Q_{p_2} \setminus Q_{pr_1} \otimes Q_{pr_2} \tag{5.63}$$

and

$$Q_{pr_1} \otimes Q_{pr_2}, \tag{5.64}$$

respectively, and

- the sets of the accepting and rejecting postselection states of $\mathcal{M}_2$ is

$$Q_{pa_1} \otimes Q_{pa_2}, \tag{5.65}$$

and

$$Q_{p_1} \otimes Q_{p_2} \setminus Q_{pa_1} \otimes Q_{pa_2}, \tag{5.66}$$

respectively.

Thus, the following inequalities can be verified for a given input string $w \in \Sigma^*$:

- if $w \in L_1 \cup L_2$, $f^a_{\mathcal{M}_1}(w) \geq \frac{15}{16}$;
- if $w \notin L_1 \cup L_2$, $f^a_{\mathcal{M}_1}(w) \leq \frac{7}{16}$;
- if $w \in L_1 \cap L_2$, $f^a_{\mathcal{M}_2}(w) \geq \frac{9}{16}$;
- if $w \notin L_1 \cap L_2$, $f^a_{\mathcal{M}_2}(w) \leq \frac{1}{16}$.

We can conclude that both classes are closed under union and intersection. □

Theorem 5.31. *PostQAL and PostS are subsets of S (QAL).*

Proof. A given RT-PostFA can be converted to its corresponding standard model (without postselection) as follows:

- All nonpostselection states of the RT-PostFA are made to transition to accepting states with probability $\frac{1}{2}$ at the end of the computation.
- All members of Q_{pa} are accepting states in the new machine.

Therefore, for the members, the overall accepting probability of the new machine exceeds $\frac{1}{2}$, and for nonmembers, it can be at most $\frac{1}{2}$. □

By using the fact that S is not closed under union and intersection [58–60], Corollary 5.29, and Theorems 5.30 and 5.31, we obtain the following corollary.

Corollary 5.32. *$PostS \subsetneq PostQAL \subsetneq S\ (QAL)$.*

5.4.3. Latvian Postselection Automata

In [11, 12, 73], a somewhat different RT-PostQFA model, that violates the assumption that the postselection is done on a set of computation branches having nonzero probability, is presented. Although the motivation for this feature is not clear in [11, 12, 73], such a violation seems reasonable due to some fundamental reasons related to the capabilities of finite automata. For example, when we are given "more" resources, we can create some computational paths with sufficiently small probabilities as a part of the postselection set such that they do not affect the overall computation but can help to accept or to reject the input as desired whenever there is zero probability of observing the other postselection states. However, we do not know how to implement such a solution for quantum or probabilistic automata[23] .

We call the machines defined in [11, 12] Latvian RT-PostFAs (RT-LPostFAs). These have an additional component $\tau \in \{A, R\}$ such that whenever the postselection probability is zero for a given input string $w \in \Sigma^*$,

- w is accepted if $\tau = A$,
- w is rejected if $\tau = R$.

The bounded-error classes corresponding to the RT-LPostPFA and RT-LPostQFA models are called LPostS and LPostQAL, respectively.

Theorem 5.33. *$LPostS = PostS$.*

[23] As a similar issue, we do not know how to increase or decrease an acceptance probability that is exactly equal to the cutpoint, for a given quantum or probabilistic automaton (in most cases). A related open problem is whether $coS = S$ or not [29, 46] even when we restrict ourselves to computable transition probabilities [74].

Proof. We need to show that LPostS $\subseteq$ PostS. Let L be in LPostS and $\mathcal{P}$ with $\tau \in \{A, R\}$ be the RT-LPostPFA recognizing L with error bound $\epsilon < \frac{1}{2}$. Suppose that L' is the language such that for each member of L', the probability of postselection assigned by $\mathcal{P}$ is zero. By designating all postselection states as accepting states and removing the probability values of transitions, we obtain a RT-NFA which recognizes $\overline{L'}$. Thus, there exists a RT-DFA, say $\mathcal{D}$, recognizing L'.

By combining (tensoring) $\mathcal{P}$ and $\mathcal{D}$, a RT-PostPFA, say $\mathcal{P}'$, can be obtained such that if the input string is a member of L', the decision is given deterministically with respect to τ, and if it is not a member of L' (the probability of the postselection is nonzero), the decision is given by standard postselection procedure. Therefore, L is recognized by $\mathcal{P}'$ with the same error bound and so L is in PostS, too. □

However, we cannot use the same idea in the quantum case due to the fact that the class of the languages recognized by NQFAs, is a proper superclass of the regular languages (see Section 4.2).

Theorem 5.34. *NQAL $\cup$ coNQAL $\subseteq$ LPostQAL.*

Proof. For $L \in$ NQAL, take the accepting states of the NQFA recognizing L as postselection accepting states with $\tau = R$. (There are no postselection rejecting states.) For $L \in$ coNQAL, take the accepting states of the NQFA recognizing $\overline{L}$ as postselection rejecting states with $\tau = A$. (There are no postselection accepting states.) □

Interestingly, all languages in NQAL $\cup$ coNQAL are recognized with zero error by RT-LPostQFAs[24] .

Theorem 5.35. *LPostQAL is closed under complementation.*

Proof. If a language is recognized by a RT-LPostQFA with bounded error, by swapping the accepting and rejecting postselection states and by setting τ to $\{A, R\} \backslash \tau$, we obtain

[24] Lāce et al. [11] describe a zero error machine for L_{pal}.

a new RT-LPostQFA recognizing the complement of the language with bounded error. Therefore, LPostQAL is closed under complementation. □

Theorem 5.36. *LPostQAL* $\subseteq$ *uQAL (uS).*

Proof. The proof is similar to the proof of Theorem 5.31 with the exception that

- if $\tau = A$, we have recognition with nonstrict cutpoint;
- if $\tau = R$, we have recognition with strict cutpoint.

□

6. WRITE-ONLY MEMORY

In this chapter, we examine realtime quantum finite automaton (RT-QFA) models augmented with a "write-only memory" (WOM) under several types of restrictions related to WOM access.

6.1. Definitions

A WOM is a two-way write-only work tape having alphabet Γ containing # and ε, where ε means that the square under the tape head is not changed. If we restrict the tape head movement of WOM to $\triangleright$, i.e. one-way, we obtain a "push-only stack" (POS). For POSs, we assume that, if ε is written, the work tape head does not move and if a symbol different than ε is written, the work tape head automatically moves one square to the right. A special case of POS is obtained by restricting Γ with ε and a *counting* symbol (different than #), this is called an "increment-only counter" (IOC). As a further restriction, one symbol (except ε) is required to be written on the work tape at every step of the computation. We call this type of memory as "trash tape" (TT).

In this chapter, we examine the power of a RT-QFA augmented by WOM, POS, or IOC, namely RT-QFA-WOM, RT-QFA-POS (0-rev-RT-QPDA), or RT-QFA-IOC (0-rev-RT-Q1CA), respectively. Note that, RT-QFA-WOMs, RT-QFA-POSs, and RT-QFA-IOCs are special cases of realtime quantum Turing machines, realtime quantum pushdown automata, and realtime quantum one-counter automata.

Formally, a RT-QFA-WOM $\mathcal{M}$ is a 7-tuple

$$(Q, \Sigma, \Gamma, \Omega, \delta, q_1, Q_a). \tag{6.1}$$

When in state $q \in Q$ and reading symbol $\sigma \in \tilde{\Sigma}$ on the input tape, $\mathcal{M}$ changes its state to $q' \in Q$, writes $\gamma \in \Gamma$ and $\omega \in \Omega$ on the WOM tape and the finite register,

respectively, and then updates the position of the WOM tape head with respect to $d_w \in <>$ with transition amplitude $\delta(q, \sigma, q', \gamma, d_w, \omega) = \alpha$, where $\alpha \in \mathbb{C}$ and $|\alpha| \leq 1$.

In order to represent all transitions from the case where $\mathcal{M}$ is in state $q \in Q$ and reading symbol $\sigma \in \tilde{\Sigma}$ together, we use the notation

$$\delta(q, \sigma) = \sum_{(q', \gamma, d_w, \omega) \in Q \times \Gamma \times <> \times \Omega} \delta(q, \sigma, q', \gamma, d_w, \omega)(q', \gamma, d_w, \omega), \tag{6.2}$$

where

$$\sum_{(q', \gamma, d_w, \omega) \in Q \times \Gamma \times <> \times \Omega} |\delta(q, \sigma, q', \gamma, d_w, \omega)|^2 = 1. \tag{6.3}$$

A configuration of a RT-QFA-WOM is the collection of

- the internal state of the machine,
- the position of the input tape head,
- the contents of the WOM tape, and the position of the WOM tape head.

The formal definition of the RT-QFA-POS is similar to that of the RT-QFA-WOM, except that the movement of the WOM tape head is restricted to $\triangleright$, and so the position of that head does not need to be a part of a configuration. On the other hand, the definition of the RT-QFA-IOC can be simplified by removing the Γ component from (6.1):

A RT-QFA-IOC $\mathcal{M}$ is a 6-tuple

$$(Q, \Sigma, \Omega, \delta, q_1, Q_a). \tag{6.4}$$

When in state $q \in Q$, and reading symbol $\sigma \in \tilde{\Sigma}$ on the input tape, $\mathcal{M}$ changes its state to $q' \in Q$, writes ω in the register, and updates the value of its counter by

$c \in \triangle = \{0, +1\}$ with transition amplitude $\delta(q, \sigma, q', c, \omega) = \alpha$, where $\alpha \in \mathbb{C}$ and $|\alpha| \leq 1$.

In order to show all transitions from the case where $\mathcal{M}$ is in state $q \in Q$ and reads symbol $\sigma \in \tilde{\Sigma}$ together, we use the notation

$$\delta(q, \sigma) = \sum_{(q', c, \omega) \in Q \times \triangle \times \Omega} \delta(q, \sigma, q', c, \omega)(q', c, \omega), \tag{6.5}$$

where

$$\sum_{(q', c, \omega) \in Q \times \triangle \times \Omega} |\delta(q, \sigma, q', c, \omega)|^2 = 1. \tag{6.6}$$

A configuration of a RT-QFA-IOC is the collection of

- the internal state of the machine,
- the position of the input tape head, and
- the value of the counter.

RT-QFA-IOC(m) is a RT-QFA-IOC with capability of incrementing its counter by a value from the set $\{0, \ldots, m\}$, where $m > 1$.

6.2. Basic Facts

We present some facts that are useful in the next parts in this section.

Lemma 6.1. *The computational power of any realtime classical finite automaton is unchanged when the model is augmented with a WOM.*

Proof. For a given machine $\mathcal{M}$ and an input string w, consider the tree $\mathcal{T}$ of states, where the root is the initial state, each subsequent level corresponds to the processing of the next input symbol, and the children of each node N are the states that have nonzero-probability transitions from S with the input symbol corresponding to that level. Each

such edge in the tree is labeled with the corresponding transition probability. The probability of node N is the product of the probabilities on the path to N from the root. The acceptance probability is the sum of the probabilities of the accept states at the last level.

Now consider attaching a WOM to $\mathcal{M}$, and augmenting its program so that every transition now also specifies the action to be taken on the WOM. Several new transitions of this new machine may correspond to a single transition of $\mathcal{M}$, since, for example, a transition with probability p can be divided into two transitions with probability $\frac{p}{2}$, whose effects on the internal state are identical, but which write different symbols on the WOM. It is clear that many different programs can be obtained by augmenting $\mathcal{M}$ in this manner with different WOM actions. Visualize the configuration tree $\mathcal{T}_{new}$ of any one of these new machines on input w. There exists a homomorphism h from $\mathcal{T}_{new}$ to $\mathcal{T}$, where h maps nodes in $\mathcal{T}_{new}$ to nodes on the same level in $\mathcal{T}$, the configurations in $h^{-1}(N)$ all have N as their states, and the total probability of the members of $h^{-1}(N)$ equals the probability of N in $\mathcal{T}$, for any N. We conclude that all the machines with WOM accept w with exactly the same probability as w, so the WOM does not make any difference. □

Fact 6.2. *[75] For every k, if L is recognized by a deterministic RT-kBCA (RT-DkBCA), then for every $\epsilon \in (0, \frac{1}{2})$, then there exists a probabilistic RT-1BCA (RT-P1BCA) recognizing L with negative one-sided error bound ϵ.*

We can generalize this result to probabilistic RT-kBCAs (RT-PkBCAs), where $k > 1$.

Lemma 6.3. *Let $\mathcal{P}$ be a given RT-PkBCA and $\epsilon \in (0, \frac{1}{2})$ be a given error bound. Then, there exists a RT-P1BCA(R) $\mathcal{P}'$ such that for all $w \in \Sigma^*$,*

$$f_{\mathcal{P}}(w) \leq f_{\mathcal{P}'}(w) \leq f_{\mathcal{P}}(w) + \epsilon(1 - f_{\mathcal{P}}(w)), \tag{6.7}$$

where $R = 2^{\lceil \frac{k}{\epsilon} \rceil}$.

Proof. Based on the probabilistic method described in Figure 6.1, we can obtain $\mathcal{P}'$ by making the following modifications on $\mathcal{P}$:

In this figure, we review a method presented by Freivalds in [75]: Given a machine with $k > 1$ counters, say $C_1, \ldots, C_k$, whose values can be updated using the increment set $\{-1, 0, 1\}$, we can build a machine with a single counter, say C, whose value can be updated using the increment set $\{-R, \ldots, R\}$ $(R = 2^{\lceil \frac{k}{\epsilon} \rceil})$, such that all updates on $C_1, \ldots, C_k$ can be simulated on C in the sense that (i) if all values of $C_1, \ldots, C_k$ are zeros, then the value of C is zero; and (ii) if the value of at least one of $C_1, \ldots, C_k$ is nonzero, then the value of C is nonzero with probability $1 - \epsilon$, where $\epsilon \in (0, \frac{1}{2})$. The probabilistic method for this simulation is as follows:

- Choose a number r equiprobably from the set $\{1, \ldots, R\}$.
- The value of C is increased (resp., decreased) by r^i if the value of C_i is increased (resp., decreased) by 1.

Figure 6.1. Probabilistic zero-checking of multiple counters by one counter

i. At the beginning of the computation, $\mathcal{P}'$ equiprobably chooses a number r from the set $\{1, \ldots, R\}$.
ii. For each transition of $\mathcal{P}$, in which the values of counters are updated by $(c_1, \ldots, c_k) \in \{-1, 0, 1\}^k$, i.e., the value of the i^{th} counter is updated by c_i $(1 \leq i \leq k)$, $\mathcal{P}$ makes the same transition by updating its counter values by $\sum_{i=1}^{k} r^i c_i$.

Hence, (i) for each accepting path of $\mathcal{P}$, the input is accepted by $\mathcal{P}'$, too; (ii) for each rejecting path of $\mathcal{P}$, the input may be accepted by $\mathcal{P}'$ with a probability at most ϵ. By combining these cases, we obtain the following inequality for any input string $w \in \Sigma^*$:

$$f_{\mathcal{P}}(w) \leq f_{\mathcal{P}'}(w) \leq f_{\mathcal{P}}(w) + \epsilon(1 - f_{\mathcal{P}}(w)) \tag{6.8}$$

□

Theorem 6.4. *If L is recognized by a RT-PkBCA with error bound $\epsilon \in (0, \frac{1}{2})$, then L*

is recognized by a RT-P1BCA with error bound ϵ' *(*$0 < \epsilon < \epsilon' < \frac{1}{2}$*). Moreover,* ϵ' *can be tuned to be arbitrarily close to* ϵ*.*

Proof. Let $\mathcal{P}$ be a RT-PkBCA recognizing L with error bound ϵ. By using the previous lemma (Lemma 6.3), for any $\epsilon'' \in (0, \frac{1}{2})$, we can construct a RT-P1BCA(R), say $\mathcal{P}''$, from $\mathcal{P}$, where $R = 2^{\lceil \frac{k}{\epsilon''} \rceil}$. Hence, depending on the value of ϵ, we can select ϵ'' to be sufficiently small such that L is recognized by $\mathcal{P}''$ with error bound $\epsilon' = \epsilon + \epsilon''(1-\epsilon) < \frac{1}{2}$. Since for each RT-P1BCA(m), there is an equivalent RT-P1BCA for any $m > 1$, L is also recognized by a RT-P1BCA with bounded error ϵ', which can be tuned to be arbitrarily close to ϵ. □

Corollary 6.5. *If* L *is recognized by a RT-PkBCA with negative one-sided error bound* $\epsilon \in (0, 1)$*, then* L *is recognized by a RT-P1BCA with negative one-sided error bound* ϵ' *(*$0 < \epsilon < \epsilon'$*). Moreover,* ϵ' *can be tuned to be arbitrarily close to* ϵ*.*

Lemma 6.6. *For a given RT-QFA-IOC(m)* $\mathcal{M}$*, these exists a RT-QFA-IOC* $\mathcal{M}'$ *such that*

$$f_{\mathcal{M}}(w) = f_{\mathcal{M}'}(w), \tag{6.9}$$

for all $w \in \Sigma^*$*, where* $m > 2$*.*

Proof. It can be easily followed from Lemma 3.4 by considering only one counter[25] and then restricting its update values with $\{0, \ldots, m\}$. □

6.3. Realtime Quantum Finite Automata with Incremental-Only Counters

We examine the capabilities of RT-QFA-IOCs in both the bounded and unbounded error settings, and show that they can simulate a family of conventional counter machines, which are themselves superior to RT-QFAs, in both these cases.

[25] In fact, the proof is independent from the number of the counters.

6.3.1. Bounded Error

The main theorem to be proven in this subsection is

Theorem 6.7. *The class of languages recognized with bounded error by RT-QFA-IOCs contains all languages recognized with bounded error by conventional realtime quantum automata with one blind counter (RT-Q1BCAs).*

Before presenting our proof of Theorem 6.7, let us demonstrate the underlying idea by showing how RT-QFA-IOCs can simulate a simpler family of machines, namely, deterministic automata with one blind counter.

Lemma 6.8. *If a language L is recognized by a RT-D1BCA, then L can also be recognized by a RT-QFA-IOC with negative one-sided error bound $\frac{1}{m}$, for any desired value of m.*

Proof. We build a RT-QFA-IOC(m) that recognizes L, which is sufficient by Lemma 6.6.

Throughout this proof, the symbol "i" is reserved for the imaginary number $\sqrt{-1}$. Let the given RT-D1BCA be $\mathcal{D} = (Q, \Sigma, \delta, q_1, Q_a)$, where $Q = \{q_1, \ldots, q_n\}$. We build $\mathcal{M} = (Q', \Sigma, \Omega, \delta', q_{1,1}, Q'_a)$, where

- $Q' = \{q_{j,1}, \ldots, q_{j,n} \mid 1 \leq j \leq m\}$,
- $Q'_a = \{q_{m,i} \mid q_i \in Q_a\}$, and
- $\Omega = \{\omega_1, \ldots, \omega_n\}$.

$\mathcal{M}$ splits the computation into m paths, i.e. path_j $(1 \leq j \leq m)$, with equal amplitude on the left end-marker ¢. That is,

$$\delta'(q_{1,1}, ¢) = \underbrace{\frac{1}{\sqrt{m}}(q_{1,t}, 0, \omega_1)}_{\mathsf{path}_1} + \cdots + \underbrace{\frac{1}{\sqrt{m}}(q_{m,t}, 0, \omega_1)}_{\mathsf{path}_m}, \tag{6.10}$$

whenever $\delta(q_1, ¢, q_t, 0) = 1$, where $1 \leq t \leq n$. Until reading the right end-marker \$, path_j proceeds in the following way: For each $\sigma \in \Sigma$ and $s \in \{1, \ldots, n\}$,

$$\mathsf{path}_j : \delta'(q_{j,s}, \sigma) \quad = \quad (q_{j,t}, c_j, \omega_s) \tag{6.11}$$

whenever $\delta(q_s, \sigma, q_t, c) = 1$, where $1 \leq t \leq n$, and

- $c_j = j$ if $c = 1$,
- $c_j = m - j + 1$ if $c = -1$, and
- $c_j = 0$, otherwise.

To paraphrase, each path separately simulates[26] the computation of $\mathcal{D}$ on the input string, going through states that correspond to the states of $\mathcal{D}$, and incrementing their counters whenever $\mathcal{D}$ changes its counter, as follows:

- path_j increments the counter by j whenever $\mathcal{D}$ increments the counter by 1,
- path_j increments the counter by $m - j + 1$ whenever $\mathcal{D}$ decrements the counter by 1, and
- path_j does not make any incrementation, otherwise.

On symbol \$, the following transitions are executed (note that the counter updates in this last step are also made according to the setup described above):
If $q_t \in Q_a$,

$$\mathsf{path}_j : \delta'(q_{j,s}, \$) = \frac{1}{\sqrt{m}} \sum_{l=1}^{m} e^{\frac{2\pi i}{m} jl} (q_{l,t}, c_j, \omega_s) \tag{6.12}$$

and if $q_t \notin Q_a$,

$$\mathsf{path}_j : \delta'(q_{j,s}, \$) = (q_{j,t}, c_j, \omega_s), \tag{6.13}$$

[26]Note that each transition of $\mathcal{M}$ in Equation 6.11 writes a symbol determined by the source state of the corresponding transition of $\mathcal{D}$ to the register. This ensures the orthonormality condition for quantum machines described earlier.

whenever $\delta(q_s, \$, q_t, c) = 1$, where $1 \leq t \leq n$.

The essential idea behind this setup, where different paths increment their counters with different values to represent increments and decrements performed by $\mathcal{D}$ is that the increment values used by $\mathcal{M}$ have been selected carefully to ensure that the counter has the same value in all of $\mathcal{M}$'s paths at any time if $\mathcal{D}$'s counter is zero at that time. Furthermore, all of $\mathcal{M}$'s paths are guaranteed to have different counter values if $\mathcal{D}$'s counter is nonzero[27] .

Let $N > 1$ be a integer. The N-way quantum Fourier transform (QFT) is the transformation

$$\delta(d_j) \rightarrow \alpha \sum_{l=1}^{N} e^{\frac{2\pi i}{N} jl}(r_l), \qquad 1 \leq j \leq N, \tag{6.14}$$

from the *domain* states $d_1, \ldots, d_N$ to the *range* states $r_1, \ldots, r_N$. r_N is the *distinguished* range element. α is a real number such that $\alpha^2 N \leq 1$. The QFT can be used to check whether separate computational paths of a quantum program that are in superposition have converged to the same configuration at a particular step. Assume that the program has previously split to N paths, each of which have the same amplitude, and whose state components are the d_j. In all the uses of the QFT in our algorithms, one of the following conditions is satisfied:

i. The WOM component of the configuration is different in each of the N paths: In this case, the QFT further divides each path to N subpaths, that differs from each other by the internal state component. No interference takes place.

ii. Each path has the same WOM content at the moment of the QFT: In this case, the paths that have $r_1, \ldots, r_{N-1}$ as their state components destructively interfere with each other [72], and $\alpha^2 N$ of the probability of the N incoming paths are accumulated on a single resulting path with that WOM content, and r_N as its state component.

Figure 6.2. The description of N-way quantum Fourier transform used by the machines having a WOM

For a given input string $w \in \Sigma^*$,

i. if $\mathcal{D}$ ends up in a state not in Q_a (and so $w \notin L$), then $\mathcal{M}$ rejects the input in each of its m paths, and the overall rejection probability is 1;

[27] This idea has been adapted from an algorithm by Kondacs and Watrous for a different type of quantum automaton, whose analysis can be found in [4].

ii. if $\mathcal{D}$ ends up in a state in Q_a, all paths make an m-way QFT (see Figure 6.2) whose distinguished target is an accepting state:
   - if the counter of $\mathcal{D}$ is zero (and so $w \in L$), all paths have the same counter value, that is, they interfere with each other, and so $\mathcal{M}$ accepts with probability 1;
   - if the counter of $\mathcal{D}$ is not zero (and so $w \notin L$), there is no interference, and each path ends by accepting w with probability $\frac{1}{m^2}$, leading to a total acceptance probability of $\frac{1}{m}$, and a rejection probability of $1 - \frac{1}{m}$.

□

Proof of Theorem 6.7. Given a RT-Q1BCA that recognizes a language L $\mathcal{M}$ with error bound $\epsilon < \frac{1}{2}$, we build a RT-QFA-IOC(m) $\mathcal{M}'$, using essentially the same construction as in Lemma 6.8: $\mathcal{M}'$ simulates m copies of $\mathcal{M}$, and these copies use the set of increment sizes described in the proof of Lemma 6.8 to mimic the updates to $\mathcal{M}$'s counter. Unlike the deterministic machine of that lemma, $\mathcal{M}$ can fork to multiple computational paths, which is handled by modifying the transformation of Equation 6.11 as

$$\mathsf{path}_j : \delta'(q_{j,s}, \sigma, q_{j,t}, c_j, \omega) = \alpha \tag{6.15}$$

whenever $\delta(q_s, \sigma, q_t, c, \omega) = \alpha$, where $1 \leq t \leq n$, and $\omega \in \Omega$, and that of Equation 6.12 as

$$\mathsf{path}_j : \delta'(q_{j,s}, \$, q_{l,t}, c_j, \omega) = \frac{\alpha}{\sqrt{m}} e^{\frac{2\pi i}{m} jl}, \text{ for } l \in \{1, \ldots, m\} \tag{6.16}$$

whenever $\delta(q_s, \$, q_t, c, \omega) = \alpha$, where $1 \leq t \leq n$ and $\omega \in \Omega$; causing the corresponding paths of the m copies of $\mathcal{M}$ to undergo the m-way QFTs associated by each accept state as described above at the end of the input.

We therefore have that the paths of $\mathcal{M}$ that end in non-accept states do the same thing with the same total probability in $\mathcal{M}'$. The paths of $\mathcal{M}$ that end in accept

states with the counter containing zero make $\mathcal{M}'$ accept also with their original total probability, thanks to the QFT. The only mismatch between the machines is in the remaining case of the paths of $\mathcal{M}$ that end in accept states with a nonzero counter value. As explained in the proof of Lemma 6.8, each such path contributes $\frac{1}{m}$ of its probability to acceptance, and the rest to rejection.

For any given input string $w \in \Sigma^*$:

- If $w \in L$, we have $f^a_{\mathcal{M}}(w) \geq 1-\epsilon$ and $f^r_{\mathcal{M}}(w) \leq \epsilon$, then

$$f^a_{\mathcal{M}'}(w) = f^a_{\mathcal{M}}(w) + \frac{1}{m} f^r_{\mathcal{M}}(w) \geq 1-\epsilon. \tag{6.17}$$

- If $w \notin L$, we have $f^a_{\mathcal{M}}(w) \leq \epsilon$ and $f^r_{\mathcal{M}}(w) \geq 1-\epsilon$, then

$$f^a_{\mathcal{M}'}(w) = f^a_{\mathcal{M}}(w) + \frac{1}{m} f^r_{\mathcal{M}}(w) \leq \epsilon + \frac{1}{m}(1-\epsilon). \tag{6.18}$$

Therefore, by setting m to a value greater than $\frac{2-2\epsilon}{1-2\epsilon}$, L is recognized by $\mathcal{M}'$ with error bound $\epsilon' = \epsilon + \frac{1}{m}(1-\epsilon) < \frac{1}{2}$. Moreover, by setting m to sufficiently large values, ϵ' can be tuned to be arbitrarily close to ϵ. □

Corollary 6.9. *If L is recognized by a RT-Q1BCA (or a RT-P1BCA) $\mathcal{P}$ with negative one-sided error bound $\epsilon < 1$, then L is recognized by a RT-QFA-IOC $\mathcal{M}$ with negative one-sided error bound ϵ', i.e. $\epsilon < \epsilon' < 1$. Moreover, ϵ' can be tuned to be arbitrarily close to ϵ.*

For a given nonnegative integer k, L_{eq-k} is the language defined over the alphabet $\{a_1, \ldots, a_k, b_1, \ldots, b_k\}$ as the set of all strings containing equal numbers of a_i's and b_i's, for each $i \in \{1, \ldots, k\}$.

Fact 6.10. *[75] For any nonnegative k, L_{eq-k} can be recognized by a RT-P1BCA with negative one-sided bounded error ϵ, where $\epsilon < \frac{1}{2}$.*

Corollary 6.11. *RT-QFA-IOCs can recognize some non-context-free languages with bounded error.*

We have therefore established that realtime quantum finite automata equipped with a WOM tape are more powerful than plain RT-QFAs, even when the WOM in question is restricted to be just a counter.

L_{eq-1}'s complement, which can of course be recognized with positive one-sided bounded error by a RT-QFA-IOC by the results above, is a deterministic context-free language (DCFL). Using the fact [47] that no nonregular DCFL can be recognized by a nondeterministic TM using $o(\log(n))$ space, together with Lemma 3.1, we are able to conclude the following.

Corollary 6.12. *QTM-WOMs are strictly superior to PTM-WOMs for any space bound $o(\log(n))$ in terms of language recognition with positive one-sided bounded error.*

6.3.2. Unbounded Error

The simulation method introduced in Lemma 6.8 turns out to be useful in the analysis of the power of increment-only counter machines in the unbounded error mode as well:

Theorem 6.13. *Any language recognized by a nondeterministic realtime automaton with one blind counter (RT-NQ1BCA) is recognized by a RT-QFA-IOC with cutpoint $\frac{1}{2}$.*

Proof. Given a RT-NQ1BCA $\mathcal{N}$, we note that it is just a RT-Q1BCA recognizing a language L with positive one-sided unbounded error [46], and we can simulate it using the technique described in the proof of Theorem 6.7. We set m, the number of copies of the RT-Q1BCA to be parallelly simulated, to 2. We obtain a RT-QFA-IOC(2) $\mathcal{M}$ such that

i. paths of $\mathcal{N}$ that end in an accepting state with the counter equaling zero lead $\mathcal{M}$ to accept with the same total probability;
ii. paths of $\mathcal{N}$ that end in an accepting state with a nonzero counter value contribute half of their probability to $\mathcal{M}$'s acceptance probability, with the other

half contributing to rejection; and

iii. paths of $\mathcal{N}$ that end in a reject state cause $\mathcal{M}$ to reject with the same total probability.

Finally, we modify the transitions on the right end-marker that enter the reject states mentioned in the third case above, so that they are replaced by equiprobable transitions to an (accept,reject) pair of states. The resulting machine recognizes L with "one-sided" cutpoint $\frac{1}{2}$, that is, the overall acceptance probability exceeds $\frac{1}{2}$ for the members of the language, and equals $\frac{1}{2}$ for the nonmembers. □

We now present a simulation of a classical model with non-blind counter.

Theorem 6.14. *If L is recognized by a realtime deterministic one-reversal one-counter automaton (1-rev-RT-D1CA), then it is recognized by a RT-QFA-IOC with cutpoint $\frac{1}{2}$.*

Proof. We assume that the 1-rev-RT-D1CA $\mathcal{D} = (Q, \Sigma, \delta, q_1, Q_a)$ recognizing L is in the following canonical form:

- the counter value of $\mathcal{D}$ never becomes nonnegative;
- the transition on ¢ does not make any change ($\delta(q_1, ¢, 0, q_1) = 1$, and $D_c(q_1) = 0$);
- Q is the union of two disjoint subsets Q_1 and Q_2, i.e.
  i. until the first decrement, the status of the counter is never checked – this part is implemented by the members of Q_1,
  ii. during the first decrement, the internal state of $\mathcal{D}$ switches to one of the members of Q_2, and
  iii. the computation after the first decrement is implemented by the members of Q_2;
- once the counter value is detected as zero, the status of the counter is not checked again.

We construct a RT-QFA-IOC $\mathcal{M} = (Q', \Sigma, \Omega, \delta', q_1, Q'_a)$, to recognize L with cutpoint $\frac{1}{2}$, where

- $Q' = \{q_1\} \cup \{q_{j,i} \mid j \in \{1, \ldots, 4\}, i \in \{1, \ldots, |Q|\}\}$,
- $Q'_a = \{q_{j,i} \mid j \in \{1, 2, 3\}, q_i \in Q_a\} \cup \{q_{4,i} \mid q_i \in Q_r\}$, and
- $\Omega = \{\omega_i \cup \omega'_i \mid i \in \{1, \ldots, |Q|\}\}$.

and the details of δ' are given in Figures 6.3 and 6.4.

In the following, "$*$" means that the corresponding transition does not depend on the status of the counter.

(i) On symbol ¢:

$$\delta'(q_1, ¢) = \underbrace{\frac{1}{\sqrt{2}}(q_{1,1}, 0, \omega_1)}_{\mathsf{path}_1} + \underbrace{\frac{1}{\sqrt{2}}(q_{2,1}, 0, \omega_1)}_{\mathsf{path}_2} \tag{6.19}$$

(ii) On symbol $\sigma \in \Sigma$: for each $q_i \in Q_1$, if $\delta(q_i, \sigma, *, q_j) = 1$ and $q_j \in Q_1$, then

$$\begin{aligned} \mathsf{path}_1 : \delta'(q_{1,i}, \sigma) &= \underbrace{(q_{1,j}, D_c(q_j), \omega_i)}_{\mathsf{path}_1} \\ \mathsf{path}_2 : \delta'(q_{2,i}, \sigma) &= \underbrace{(q_{2,j}, 0, \omega_i)}_{\mathsf{path}_2} \end{aligned} . \tag{6.20}$$

(iii) On symbol \$: for each $q_i \in Q_1$, if $\delta(q_i, \$, *, q_j) = 1$ and $q_j \in Q_1$, then

$$\begin{aligned} \mathsf{path}_1 : \delta'(q_{1,i}, \sigma) &= \underbrace{(q_{1,j}, 0, \omega_i)}_{\mathsf{path}_1} \\ \mathsf{path}_2 : \delta'(q_{2,i}, \sigma) &= \underbrace{(q_{2,j}, 0, \omega_i)}_{\mathsf{path}_2} \end{aligned} . \tag{6.21}$$

Figure 6.3. The details of the transition function of the RT-QFA-IOC presented in the proof of Theorem 6.14 (I)

$\mathcal{M}$ starts by branching to two paths, path_1 and path_2, with equal amplitude. These paths simulate $\mathcal{D}$ in parallel according to the specifications in Figure 6.3 until $\mathcal{D}$ decrements its counter for the first time. From that step on, path_1 and path_2 split further to create new offshoots (called path_3 and path_4,) on every symbol until the end of the computation, as seen in Figure 6.4. Throughout the computation, path_1 (resp., path_2) increments its counter whenever $\mathcal{D}$ is supposed to increment (resp., decrement) its counter. Since $\mathcal{M}$'s counter is write-only, it has no way of determining which transition $\mathcal{D}$ makes depending on its counter sign. This problem is solved by assigning different paths of $\mathcal{M}$ to these branchings of $\mathcal{D}$: path_1 and path_2 (the "pre-zero paths")

(iv) On symbol $\sigma \in \Sigma$: for each $q_i \in Q$, if $\delta(q_i, \sigma, 1, q_j) = 1$ and $q_j \in Q_2$, then

$$\begin{aligned} \mathsf{path}_1 : \delta'(q_{1,i}, \sigma) &= \underbrace{\frac{1}{\sqrt{3}}(q_{1,j}, 0, \omega_i)}_{\mathsf{path}_1} + \underbrace{\frac{1}{\sqrt{3}}(q_{3,j}, 0, \omega_i')}_{\mathsf{path}_3} + \underbrace{\frac{1}{\sqrt{3}}(q_{4,j}, 0, \omega_i')}_{\mathsf{path}_4} \\ \mathsf{path}_2 : \delta'(q_{2,i}, \sigma) &= \underbrace{\frac{1}{\sqrt{3}}(q_{2,j}, c_2, \omega_i)}_{\mathsf{path}_2} + \underbrace{\frac{1}{\sqrt{3}}(q_{3,j}, c_2, \omega_i')}_{\mathsf{path}_3} - \underbrace{\frac{1}{\sqrt{3}}(q_{4,j}, c_2, \omega_i')}_{\mathsf{path}_4} \end{aligned}, \tag{6.22}$$

and if $\delta(q_i, \sigma, 0, q_j) = 1$, then

$$\begin{aligned} \mathsf{path}_3 : \delta'(q_{3,i}, \sigma) &= \underbrace{(q_{3,j}, 0, \omega_i)}_{\mathsf{path}_3} \\ \mathsf{path}_4 : \delta'(q_{4,i}, \sigma) &= \underbrace{(q_{4,j}, 0, \omega_i)}_{\mathsf{path}_4} \end{aligned}, \tag{6.23}$$

where $c_2 = 1$ only if $D_c(q_j) = -1$.

(iii) On symbol \$: for each $q_i \in Q$, if $\delta(q_i, \$, 1, q_j) = 1$ and $q_j \in Q_2$, then

$$\begin{aligned} \mathsf{path}_1 : \delta'(q_{1,i}, \sigma) &= \underbrace{\frac{1}{\sqrt{3}}(q_{1,j}, 0, \omega_i)}_{\mathsf{path}_1} + \underbrace{\frac{1}{\sqrt{3}}(q_{3,j}, 0, \omega_i)}_{\mathsf{path}_3} + \underbrace{\frac{1}{\sqrt{3}}(q_{4,j}, 0, \omega_i)}_{\mathsf{path}_4} \\ \mathsf{path}_2 : \delta'(q_{2,i}, \sigma) &= \underbrace{\frac{1}{\sqrt{3}}(q_{2,j}, c_2, \omega_i)}_{\mathsf{path}_2} + \underbrace{\frac{1}{\sqrt{3}}(q_{3,j}, c_2, \omega_i)}_{\mathsf{path}_3} - \underbrace{\frac{1}{\sqrt{3}}(q_{4,j}, c_2, \omega_i)}_{\mathsf{path}_4} \end{aligned}, \tag{6.24}$$

and if $\delta(q_i, \$, 0, q_j) = 1$, then

$$\begin{aligned} \mathsf{path}_3 : \delta'(q_{3,i}, \sigma) &= \underbrace{(q_{3,j}, 0, \omega_i)}_{\mathsf{path}_3} \\ \mathsf{path}_4 : \delta'(q_{4,i}, \sigma) &= \underbrace{(q_{4,j}, 0, \omega_i)}_{\mathsf{path}_4} \end{aligned}, \tag{6.25}$$

where $c_2 = 1$ only if $D_c(q_j) = -1$.

Figure 6.4. The details of the transition function of the RT-QFA-IOC presented in the proof of Theorem 6.14 (II)

always assume that $\mathcal{D}$'s counter has not returned to zero yet by being decremented, whereas path_3s and path_4s (the "post-zero paths") carry out their simulations by assuming otherwise. Except for path_4s, all paths imitate $\mathcal{D}$'s decision at the end of the computation. path_4s, on the other hand, accept if and only if their simulation of $\mathcal{D}$ rejects the input.

If $\mathcal{D}$ never decrements its counter, $\mathcal{M}$ ends up with the same decision as $\mathcal{D}$ with probability 1. We now focus on the other cases. As seen in Figure 6.4, the pre-zero paths lose some of their amplitude on each symbol in this stage by performing a QFT to a new pair of post-zero paths. The outcome of this transformation depends on the status of $\mathcal{D}$'s counter at this point in the simulation by the pre-zero paths:

- If $\mathcal{D}$'s counter has not yet returned to zero, then path_2's counter has a smaller value than path_1's counter, and so they cannot interfere via the QFT. The newly created post-zero paths contribute equal amounts to the acceptance and rejection probabilities at the end of the computation.
- If path_1 and path_2 have the same counter value as a result of this transition, this indicates that $\mathcal{D}$ has performed exactly as many decrements as its previous increments, and its counter is therefore zero. The paths interfere, the target path_4's cancel each other, and path_3 survives after the QFT with a probability that is twice that of the total probability of the ongoing pre-zero paths.

As a result, it is guaranteed that the path that is carrying out the correct simulation of $\mathcal{D}$ dominates $\mathcal{M}$'s decision at the end of the computation: If $\mathcal{D}$'s counter ever returns to zero, the path_3 that is created at the moment of that last decrement has sufficient probability to tip the accept/reject balance. If $\mathcal{D}$'s counter never returns to zero, then the common decision by the pre-zero paths on the right end-marker determines whether the overall acceptance or the rejection probability is greater than $\frac{1}{2}$. □

L_{NH} is recognizable by both 1-rev-RT-D1CAs and RT-N1BCAs[28] (and so RT-NQ1BCAs). It is known (see Section 4.1) that neither a RT-QFA nor a $o(\log(\log(n)))$-space PTM can recognize L_{NH} with unbounded error. We therefore have the following corollary.

Corollary 6.15. *QTM-WOMs are strictly superior to PTM-WOMs for any space bound* $o(\log(\log(n)))$ *in terms of language recognition with unbounded error.*

[28]RT-N1BCAs can also recognize L_{center}, and the languages presented in Figure 4.3, none of which can be recognized by RT-QFAs with unbounded error.

6.4. Realtime Quantum Finite Automata with Push-Only Stacks

We conjecture that allowing more than one nonblank/nonempty symbol in the WOM tape alphabet of a QFA increases its computational power. We consider, in particular, the language $L_{twin} = \{wcw \mid w \in \{a,b\}^*\}$:

Theorem 6.16. *There exists a RT-QFA-POS that recognizes the language L_{twin} with negative one-sided error bound $\frac{1}{2}$.*

Proof. We construct a RT-QFA-POS $\mathcal{M} = (Q, \Sigma, \Gamma, \Omega, \delta, q_1, Q_a)$, where $Q = \{q_1, q_2, q_3, p_1, p_2, p_3\}$, $Q_a = \{q_2\}$, $\Omega = \{\omega_1, \omega_2\}$, and $\Gamma = \{\#, a, b, \varepsilon\}$. The transition details are shown in Figure 6.5.

i. The computation splits into two paths, path_1 and path_2, with equal amplitude at the beginning.
ii. path_1 (resp., path_2) scans the input, and copies w_1 (resp., w_2) to the POS if the input is of the form w_1cw_2, where $w_1, w_2 \in \{a,b\}^*$.
    - If the input is not of the form w_1cw_2, both paths reject.
    - Otherwise, path_1 and path_2 perform a QFT at the end of the computation, where the distinguished range element is an accept state.

The configurations at the ends of path_1 and path_2 interfere with each other, i.e., the machine accepts with probability 1, if and only if the input is of the form wcw, $w \in \{a,b\}^*$. Otherwise, each of path_1 and path_2 contributes at most $\frac{1}{4}$ to the overall acceptance probability, and the machine accepts with probability at most $\frac{1}{2}$. □

Lemma 6.17. *No PTM (or PTM-WOM) using $o(\log(n))$ space can recognize L_{twin} with bounded error.*

Proof. Any PTM using $o(\log(n))$ space to recognize L_{twin} with bounded error can be used to construct a PTM recognizing the palindrome language L_{pal} with bounded error using the same amount of space. (One would only need to modify the L_{twin} machine to

On symbol ¢:

$$\delta(q_1, ¢) = \underbrace{\frac{1}{\sqrt{2}}(q_1, \varepsilon, \omega_1)}_{\mathsf{path}_1} + \underbrace{\frac{1}{\sqrt{2}}(p_1, \varepsilon, \omega_1)}_{\mathsf{path}_2} \tag{6.26}$$

On symbols from Σ:

$$\mathsf{path}_1 : \begin{cases} \delta(q_1, a) &= (q_1, a, \omega_1) \\ \delta(q_2, a) &= (q_2, \varepsilon, \omega_1) \\ \delta(q_1, b) &= (q_1, b, \omega_1) \\ \delta(q_2, b) &= (q_2, \varepsilon, \omega_1) \\ \delta(q_1, c) &= (q_2, \varepsilon, \omega_1) \\ \delta(q_2, c) &= (q_3, \varepsilon, \omega_1) \\ \delta(q_3, a) &= (q_3, \varepsilon, \omega_2) \\ \delta(q_3, b) &= (q_3, \varepsilon, \omega_2) \\ \delta(q_3, c) &= (q_3, \varepsilon, \omega_2) \end{cases} \tag{6.27}$$

$$\mathsf{path}_2 : \begin{cases} \delta(p_1, a) &= (p_1, \varepsilon, \omega_1) \\ \delta(p_2, a) &= (p_2, a, \omega_1) \\ \delta(p_1, b) &= (p_1, \varepsilon, \omega_1) \\ \delta(p_2, b) &= (p_2, b, \omega_1) \\ \delta(p_1, c) &= (p_2, \varepsilon, \omega_1) \\ \delta(p_2, c) &= (p_3, \varepsilon, \omega_1) \\ \delta(p_3, a) &= (p_3, \varepsilon, \omega_2) \\ \delta(p_3, b) &= (p_3, \varepsilon, \omega_2) \\ \delta(p_3, c) &= (p_3, \varepsilon, \omega_2) \end{cases} \tag{6.28}$$

On symbol \$:

$$\mathsf{path}_1 : \begin{cases} \delta(q_1, \$) &= (q_1, \varepsilon, \omega_1) \\ \delta(q_2, \$) &= \frac{1}{\sqrt{2}}(q_2, \varepsilon, \omega_1) + \frac{1}{\sqrt{2}}(q_3, \varepsilon, \omega_2) \\ \delta(q_3, \$) &= (q_3, \varepsilon, \omega_1) \end{cases} \tag{6.29}$$

$$\mathsf{path}_2 : \begin{cases} \delta(p_1, \$) &= (p_1, \varepsilon, \omega_1) \\ \delta(p_2, \$) &= \frac{1}{\sqrt{2}}(q_2, \varepsilon, \omega_1) - \frac{1}{\sqrt{2}}(q_3, \varepsilon, \omega_2) \\ \delta(p_3, \$) &= (q_3, \varepsilon, \omega_1) \end{cases} \tag{6.30}$$

Figure 6.5. The transitions of the RT-QFA-POS of Theorem 6.16

treat the right end-marker on the tape as the symbol c, and switch its head direction when it attempts to go past that symbol.) It is however known [35] that no PTM using $o(\log(n))$ space can recognize L_{pal} with bounded error. □

We are now able to state a stronger form of Corollary 6.12, which referred only to one-sided error:

Corollary 6.18. *QTM-WOMs are strictly superior to PTM-WOMs for any space bound $o(log(n))$ in terms of language recognition with bounded error.*

Open Problem 14. Can a one-way probabilistic pushdown automaton recognize L_{twin} with bounded error?

6.5. Realtime Quantum Finite Automata with Write-Only Memories

In this section, we present a bounded-error RT-QFA-WOM that recognizes a language for which we currently do not know a RT-QFA-POS algorithm, namely,

$$L_{rev} = \{wcw^{\mathrm{r}} \mid w \in \{a,b\}^*\}, \tag{6.31}$$

where w^{r} is the reverse of string w. Note that this language can also be recognized by a deterministic pushdown automaton.

Theorem 6.19. *There exists a RT-QFA-WOM that recognizes L_{rev} with negative one-sided error bound $\frac{1}{2}$.*

Proof. (Sketch) We use almost the same technique presented in the proof of Theorem 6.16. The computation is split into two paths (path_1 and path_2) with equal amplitude at the beginning of the computation. Each path checks whether the input string is of the form w_1cw_2, where $w_1, w_2 \in \{a,b\}^*$ and rejects with probability 1 if it is not. We assume that the input string is of the form w_1cw_2 in the rest of this proof. Until the c is read, path_1 copies w_1 to the WOM tape, and path_2 just moves the WOM tape head one square to the right at each step, without writing anything. After reading the c, the

direction of the WOM tape head is reversed in both paths. That is, path_1 moves the WOM tape head one square to the left at each step, without writing anything, while path_2 writes w_2 in the reverse direction (from the right to the left) on the WOM tape. When the right end-marker is read, the paths make a QFT, as in the proof of Theorem 6.16. It is easy to see that the two paths interfere if and only if $w_1 = w_2^{\mathrm{r}}$, and the input string is accepted with probability 1 if it is a member of L_{rev}, and with probability $\frac{1}{2}$ otherwise. □

By an argument similar to the one used in the proof of Lemma 6.17, L_{rev} can not be recognized with bounded error by any PTM using $o(\log(n))$ space, since the existence of any such machine would lead to a PTM that recognizes the palindrome language using the same amount of space.

Open Problem 15. Can a RT-QFA-POS recognize L_{rev} with bounded error?

It is easy to see that a WOM of constant size adds no power to a conventional machine. All the algorithms we considered until now used $\Omega(n)$ squares of the WOM tape on worst-case inputs. What is the minimum amount of WOM that is required by a QFA-WOM recognizing a nonregular language? Somewhat less ambitiously, one can ask whether there is any nonregular language recognized by a RT-QFA-WOM with sublinear space. We answer this question positively for middle-space usage, that is, when we are only concerned with the space used by the machine when the input is a member of the language.

Let $(i)_2^{\mathrm{r}}$ be the reverse of the binary representation of $i \in \mathbb{N}$. Consider the language

$$L_{rev-bins} = \{a(0)_2^{\mathrm{r}} a(1)_2^{\mathrm{r}} a \cdots a(k)_2^{\mathrm{r}} a \mid k \in \mathbb{Z}^+\}. \tag{6.32}$$

Theorem 6.20. *$L_{rev-bins}$ can be recognized by a RT-QFA-WOM $\mathcal{M}$ with negative one-sided error bound $\frac{3}{4}$, and the WOM usage of $\mathcal{M}$ for the members of $L_{rev-bins}$ is $O(\log n)$, where n is the length of the input string.*

Proof. It is not hard to modify the RT-QFA-POS recognizing L_{twin} to obtain a new RT-QFA-POS, say $\mathcal{M}'$, in order to recognize language $L_{twin'} = \{(i)_2^{\mathrm{r}} a (i+1)_2^{\mathrm{r}} \mid i \geq 0\}$ with negative one-sided error bound $\frac{1}{2}$. Our construction of $\mathcal{M}$ is based on $\mathcal{M}'$. The main idea is to use $\mathcal{M}'$ in a loop in order to check the consecutive blocks of $\{0,1\}^+ a \{0,1\}^+$ between two a's. In each iteration, the WOM tape head reverses direction, and so the previously used space can be used again and again. Note that, whenever $\mathcal{M}'$ executes a rejecting transition, $\mathcal{M}$ enters a path which rejects the input when it arrives at the right end-marker, and whenever $\mathcal{M}'$ is supposed to execute an accepting transition (except at the end of the computation), $\mathcal{M}$ enters the next iteration. At the end of the input, the input is accepted by $\mathcal{M}$ if $\mathcal{M}'$ accepts in its last iteration.

Let w be an input string. We assume that w is of the form

$$a\{0,1\}^+ a \{0,1\}^+ a \cdots a \{0,1\}^+ a. \tag{6.33}$$

(Otherwise, it is rejected with probability 1.) At the beginning, the computation is split equiprobably into two branches, branch_1 and branch_2. (These never interfere with each other.) branch_1 (resp., branch_2) enters the block-checking loop after reading the first (resp., the second) a. Thus, at the end of the computation, one of the branches is in the middle of an iteration, and the other one has just finished its final iteration. The branch whose iteration is interrupted by reading the end-marker accepts with probability 1.

If $w \in L_{rev-bins}$, neither branch enters a reject state, and the input is accepted with probability 1. On the other hand, if $w \notin L_{rev-bins}$, there must be at least one block $\{0,1\}^+ a \{0,1\}^+$ that is not a member of $L_{twin'}$, and so the input is rejected with probability $\frac{1}{2}$ in one branch. Therefore, the overall accepting probability can be at most $\frac{3}{4}$.

It is easy to see that the WOM usage of this algorithm for members of $L_{rev-bins}$ is $O(\log n)$. □

7. SUBLINEAR-SPACE REALTIME TURING MACHINES

In this chapter, we give space lower bounds of realtime classical Turing machines recognizing nonregular languages. In fact, we validate the lower bounds of one-way machines for realtime machines. We refer the reader to [2] for a detailed background.

By combining previous results with ours, we can obtain Figure 7.1, in which the lower bounds following from our results are presented in bold. Also note that the slots containing symbol "?" in the figure are still open.

Table 7.1. Lower bounds of {D,N,A}TMs in order to recognize a nonregular language

	general case			unary case		
	Strong	Middle	Weak	Strong	Middle	Weak
1DTM	$\log n$	$\log n$	$\log n$	$\log n$	$\log n$	$\log n$
1NTM	$\log n$	$\log n$	$\log \log n$	$\log n$	$\log n$	$\log \log n$
1ATM	$\log n$	$\log \log n$	$\log \log n$	$\log n$	$\log n$	$\log \log n$
RT-DTM	$\mathbf{\log} n$	$\mathbf{\log} n$	$\mathbf{\log} n$	$\mathbf{\log} n$	$\mathbf{\log} n$	$\mathbf{\log} n$
RT-NTM	$\mathbf{\log} n$	$\mathbf{\log} n$	$\mathbf{\log \log} n$	$\mathbf{\log} n$	$\mathbf{\log} n$	?
RT-ATM	$\mathbf{\log} n$	$\mathbf{\log \log} n$	$\mathbf{\log \log} n$	$\mathbf{\log} n$	$\mathbf{\log} n$	?

Let h_κ be a homomorphism such that

- $h_\kappa(x) = x$ if $x \neq \kappa$ and
- $h_\kappa(\kappa) = \varepsilon$;

$\mathcal{D}$ be the RT-DTM running in logarithmic space and $L_{upal\text{-}\kappa}$ satisfying $h_\kappa(L_{upal\text{-}\kappa}) = L_{upal}$ be the language recognized by it; $\Sigma = \{a, b, \kappa\}$ and $\Gamma = \{\langle\, , 0, 1, \rangle\, , \#\}$.

For a given input $w \in \Sigma^*$, the specifications of $\mathcal{D}$ are as follows:

i. During the computation, parallel to its main tasks, which is described in the

following items, $\mathcal{D}$ checks whether $h_\kappa(w)$ is of the form a^*b^*. If not, w is rejected.

ii. By reading three κ's, $\mathcal{D}$ moves the work tape head three squares to the right. By reading three more κ's, $\mathcal{D}$ consecutively writes symbols $\rangle$, 0, and $\langle$ on the work tape in the reverse direction. If any of the first six symbols is different than κ, then the input is rejected. After reading six consecutive κ's, $\mathcal{D}$ does nothing until reading a symbol different than κ.

iii. Now, the content of the work tape is "$\langle\, 0\, \rangle \#\# \cdots$" and the work tape head is positioned on symbol $\langle$. Our aim is to keep a counter, used to compare the number of a's and b's, in reverse direction between symbols $\langle$ and $\rangle$.

iv. Let P^+ (resp., P^-) be a deterministic procedure having a finite instruction set, that starts its movements when the work tape head on symbol $\langle$ and then increases (resp., decreases) the counter by one and finishes its movement by leaving the work tape head again on symbol $\langle$. The number of steps required by P^+ (resp., P^-) depends on the current content of the counter. Let p^+ (resp., p^-) : $\{\langle\, (0 \cup 1)^*\, \rangle\} \rightarrow \mathbb{N}$ be the function representing this number. Additionally, P^- always checks whether the value of the counter becomes 0 or not.

v. After reading an a (resp., a b), $\mathcal{D}$ expects to read at least $p^+(\langle\, n\, \rangle)$ (resp., $p^-(\langle\, n\, \rangle)$) κ's before reading a symbol different than κ in order to complete the task of P^+ (resp., P^-) successfully, where n is the value of the counter before updating. That is, $\mathcal{D}$ executes each step of P^+ (resp., P^-) by reading a single κ. If $\mathcal{D}$ reads a symbol different than κ before finishing the task of P^+ (resp., P^-), the input is rejected. After completing the task of P^+ (resp., P^-), $\mathcal{D}$ does nothing until reading a symbol different than κ.

vi. Whenever the counter becomes 0, $\mathcal{D}$ expects to read $\kappa^*\$$. If so, the input is accepted. Otherwise, the input is rejected.

It can easily be verified that the space used by $\mathcal{D}$ is $O(log(|w|))$. Assume that $L_{upal\text{-}\kappa}$ is regular. Then, L_{upal} must be regular, too, since regular languages are closed under homomorphism. We arrive at a contradiction. Therefore, $L_{upal\text{-}\kappa}$ is a nonregular language.

Now, we present a unary nonregular language, say $L_{power\text{-}\kappa} \subset \{\kappa\}^*$, recognized

by a RT-DTM, say $\mathcal{D}$, in logarithmic space. The idea behind is similar to the previous algorithm.

For a given input $w \in \Sigma^*$, the specifications of $\mathcal{D}$ are as follows:

i. By reading exactly six consecutive κ's, the counter, again kept in reverse direction, is prepared as $\langle\, 0\, \rangle$. If there are fewer leading κ's, then the input is rejected. (If the seventh symbol is \$, then the input is accepted.) The work tape head is placed on symbol $\langle$.

ii. By reading a block of κ's,
   - the work tape head goes from symbol $\langle$ to symbol $\rangle$ while incrementing the counter by 1, and then,
   - the work tape head goes from symbol $\rangle$ to symbol $\langle$ and while checking whether the counter value is a power of 2 or not.

   During the travel of the work tape head, we assume that it cannot be stationary on any symbol. Note that, if required, the place of symbol $\rangle$ on the work tape is shifted one square to the right. If $\mathcal{D}$ reads \$ before the work tape head is placed on symbol $\langle$, the input is rejected.

iii. When the work tape head is placed on symbol $\langle$ and the current scanned symbol from the input tape is \$, the input is accepted if the counter is a power of 2 and rejected otherwise.

The members of $L_{power\text{-}\kappa}$ can be enumerated as shorter strings come first. Let a_i be the i^{th} element of $L_{power\text{-}\kappa}$, where $i > 0$. It can be verified easily that the value of $|a_{i+1}| - |a_i|$ increases when i gets bigger. Therefore, for any pumping length p, we get a nonmember by pumping a_i once when $|a_i| > p$ and $|a_{i+1}| - |a_i| > p$. We can conclude that $L_{power\text{-}\kappa}$ is a nonregular language.

We assume in the following parts that all 1TMs never move their work tape heads to the square indexed by 0. Note that, any arbitrary 1**X**TM can be converted to a 1**X**TM having the restriction above without lose of any space resource, where **X** $\in$ {D, N, A, P}. Additionally, we assume that Σ does not contain symbol κ and

$\Sigma_\kappa = \Sigma \cup \{\kappa\}$.

For a given 1TM, a *stationary-transition* is a transition in which the 1TM does not move its input head to the right and a *to-right-transition* is a transition in which the 1TM moves its input head to the right.

Definition 7.1. *For a given 1DTM (resp., 1NTM or 1ATM) $\mathcal{D}$, $\mathcal{D}_\kappa$ is a RT-DTM (resp., RT-NTM or RT-ATM) implementing the instructions of $\mathcal{D}$ with the following specifications:*

- *on each non-κ symbol, i.e. a symbol different than κ, $\mathcal{D}_\kappa$ behaves as if $\mathcal{D}$ scans this symbol;*
- *$\mathcal{D}_\kappa$ requires to see a κ on the input tape after implementing a stationary-transition of $\mathcal{D}$; and, if this is not the case, called missing-κ case, $\mathcal{D}_\kappa$ rejects the input;*
- *on each κ, $\mathcal{D}_\kappa$ behaves as if $\mathcal{D}$ scans the symbol that is the last non-κ symbol scanned by $\mathcal{D}_\kappa$;*
- *after implementing a to-right-transition of $\mathcal{D}$, $\mathcal{D}_\kappa$ waits to scan a non-κ symbol and so discards all κ's until seeing a non-κ symbol.*

At the end, $\mathcal{D}_\kappa$ follows the decision of $\mathcal{D}$ unless the input is rejected due to missing-κ case.

Definition 7.2. *For a given 1DTM (resp., 1NTM or 1ATM) $\mathcal{D}$, if L is the language recognized by $\mathcal{D}$, then, $L_{\text{-}\kappa} \subset \Sigma_\kappa^*$ is the language recognized by $\mathcal{D}_\kappa$.*

Remark 7.3. $h_\kappa(L_{\text{-}\kappa}) = L$.

Remark 7.4. *If L is not a member of class $\mathcal{C}$ that is closed under homomorphism, then $L_{\text{-}\kappa}$ is not a member of $\mathcal{C}$, too.*

Theorem 7.5. *If L is recognized by a 1DTM (resp., a 1NTM or a 1ATM) $\mathcal{D}$ in strong-space $s(n)$ and there exists a nondecreasing function $t(n)$ such that $s(n) \in O(t(n))$, then $L_{\text{-}\kappa}$ is recognized by $\mathcal{D}_\kappa$ in strong-space $O(t(n))$.*

Proof. For any $w \in \Sigma^*$, the space used by $\mathcal{D}_\kappa$ on any input which is a member of $\{w_\kappa \in \Sigma_\kappa^* \mid h_\kappa(w_\kappa) = w\}$ is at most equal to the space used by $\mathcal{D}$ on w. □

Theorem 7.6. *If L is recognized by a 1DTM (resp., a 1NTM or a 1ATM) $\mathcal{D}$ in middle-space $s(n)$ and there exists a nondecreasing function $t(n)$ such that $s(n) \in O(t(n))$, then $L_{\text{-}\kappa}$ is recognized by $\mathcal{D}_\kappa$ in middle-space $O(t(n))$.*

Proof. Similar to the previous one. □

Let $L_{gcm} = \{a^n b^M \mid M \text{ is a common multiple of all } i \leq n\}$. Szepietowski [76] showed that L_{gcm} is in middle-one-way-ASPACE($\log\log(n)$). Let $\mathcal{A}$ be the 1ATM presented by Szepietowski.

Corollary 7.7. *$L_{gcm\text{-}\kappa}$ is a nonregular language and recognized by RT-ATM $\mathcal{A}_\kappa$ in middle-space $O(\log\log(n))$.*

$L_{m\neq n} = \{a^m b^n \mid m \neq n\}$ is in weak-one-way-NSPACE($\log\log(n)$) [2]. Let $\mathcal{N}$ be the 1NTM, presented on Page 23 in [2], recognizing $L_{m\neq n}$ in weak-space $O(\log\log(n))$. We give a brief description of $\mathcal{N}$ below. At the beginning of the computation, $\mathcal{N}$ nondeterministically selects a number $l > 1$. The work tape of $\mathcal{N}$ is divided into three tracks so that the top track can store the value of l, which is written at the beginning of the computation and the middle track (resp., bottom track) can store the value of $m \mod (l)$ (resp., $n \mod (l)$), which is iteratively calculated whenever an a (resp., a b) is read. When symbol \$ is read, the values of the middle and bottom tracks are compared. If they are not equal, the input is accepted. The nondeterministic path corresponding to l, namely npath_l, needs only $\Theta(\log(l))$ space. We can conclude that $L_{m\neq n}$ is recognized by $\mathcal{N}$ in weak-space($\log\log(n)$), by using the following fact.

Fact 7.8. *(On Page 22 of [2]) There exists a constant c such that for every pair of natural numbers m and n, $m \neq n$, there exists a number $l < c\log(m+n)$ such that $m \not\equiv n \mod (l)$.*

In the above algorithm, after reading a symbol on the input tape, $\mathcal{N}$ implements some operations on the work tape by making stationary steps on the input tape. The number of such stationary steps can be fixed to a value, say d_l, depending on only the

number written at the top track of the work tape ($l > 1$). More specifically, we can use the following strategy:

- the work tape head is always placed on the leftmost nonblank symbol before reading the next symbol on the input tape and
- the operations on the work tape can be completed by alternating the work tape head, which operates with the speed of one square per step, between the leftmost and rightmost nonblank symbols $2d_1 > 0$ times.

By selecting d_1 sufficiently bigger, d_l can be set to $2d_1\lceil \log l \rceil + d_2$ for nondeterministic path npath_l, where $d_2 \in \mathbb{Z}$ depends on how to store the numbers on the work tape. Thus, we can guarantee that for any $l' > l$, the number of the stationary steps (on the input tape) of $\mathsf{npath}_{l'}$ cannot be less than that of npath_l. Let $\mathcal{N}'$ be the 1NTM implementing the above algorithm and $L_{m\neq n\text{-}\kappa}$ be the language recognized by $\mathcal{N}'_\kappa$.

Lemma 7.9. *$L_{m\neq n\text{-}\kappa}$ is a nonregular language and is recognized by RT-NTM $\mathcal{N}'_\kappa$ in weak-space $O(\log\log(n))$.*

Proof. $L_{m\neq n\text{-}\kappa}$ is a nonregular language due to the fact that $L_{m\neq n}$ is nonregular and REG is closed under homomorphism.

Let $w = a^m b^n$ be a member of $L_{m\neq n}$ and l be the smallest number satisfying that $m \not\equiv n \mod (l)$. Then, for all $w_\kappa \in L_{m\neq n\text{-}\kappa}$ satisfying $h_\kappa(w_\kappa) = w$, the nondeterministic path of $\mathcal{N}'_\kappa$ corresponding to l always accepts the computation. □

8. RELATED WORK

In this chapter, we present many results not classified as a separate section. In most cases, we give the sketch of the proofs.

8.1. Counter and Pushdown Automata

Let $L_{ijk} = \{a^i b^j c^k \mid i \neq j, i \neq k, j \neq k, 0 \leq i, j, k\}$ be a language over alphabet $\Sigma = \{a, b, c\}$.

Theorem 8.1. *L_{ijk} is in $S_{\mathbb{Q}}^{\neq}$ (NQAL).*

Proof. (Sketch) Let w be an input string of the form $a^*b^*c^*$. We can design a GFA to calculate the value of $(|w|_a - |w|_b)$. Thus, we can also construct a GFA to calculate the value of

$$(|w|_a - |w|_b)^2(|w|_a - |w|_c)^2(|w|_b - |w|_c)^2. \tag{8.1}$$

This value is a positive integer if w is a member of L_{ijk} and it is zero if w is not a member of L_{ijk}. □

Since L_{ijk} is also a member of RT-NQSTACK(1) and not in CFL, we can obtain the following corollary.

Corollary 8.2. *For any $s \in O(n)$,*

$$\textit{one-way-NSTACK(s)} \subsetneq \textit{one-way-NQSTACK(s)} \tag{8.2}$$

and

$$\textit{RT-NSTACK(s)} \subsetneq \textit{RT-NQSTACK(s)}. \tag{8.3}$$

By using Theorem 6.20, we have the following result.

Corollary 8.3. *$L_{rev-bins} \in$ middle-RT-BQSTACK($\log n$).*

Remember form the previous section that $(i)_2$ is the binary representation of $i \in \mathbb{N}$ and $(i)_2^{\mathrm{r}}$ is the reverse of the binary representation of i.

$$L_{twin-rev-bins} = \{a(0)_2 a(1)_2^{\mathrm{r}} a(2)_2 a(3)_2^{\mathrm{r}} a \cdots a(2k)_2 a(2k+1)_2^{\mathrm{r}} \mid k > 0\} \tag{8.4}$$

Theorem 8.4. *$L_{twin-rev-bins}$ can be recognized by a RT-PPDA with negative one-sided error bound $\frac{1}{2}$. The machine uses $O(\log n)$ space on its stack for the members.*

Proof. (Sketch) The computation splits equiprobable into two branch. One of the branch (resp., the other branch) consecutively checks the binary numbers positioned in 1 and 2, 3 and 4, etc. (resp., 2 and 3, 4 and 5, etc.) If the given input string is a member of the language, then all checks succeed and the input is accepted with probability 1. If it is not a member of the language, then at least one check fails and the input is accepted with probability at most $\frac{1}{2}$. □

Theorem 8.5. *$L_{rev-bins}$ and $L_{twin-rev-bins}$ can be recognized by a RT-P1CA with negative one-sided unbounded error. The machine uses $O(\log n)$ space on its counter for the members.*

Proof. (Sketch) The proof is similar to the above one. By using a counter, L_{twin} or L_{rev} can be recognized by a PFA with negative one-sided unbounded error. Adding or subtracting a constant from a binary number can be easily implemented if it is accessed in reverse direction. □

Corollary 8.6. *$L_{rev-bins}$ and $L_{twin-rev-bins}$ can be recognized by a RT-A1CA. The machine uses $O(\log n)$ space on its counter for the members.*

$L_{\mathbb{N}} = \{baba^2ba^3b \cdots ba^k \mid k \in \mathbb{N}\}$ is recognized by a RT-D2CA exactly and a

RT-P1CA with negative one-sided error bound $\frac{1}{2}$. The RT-D2CA uses $O(\sqrt{n})$ space for all strings but the RT-P1CA uses $O(\sqrt{n})$ space for the members.

$L_{twin-rev} = \{w_1 c w_2 c w_1^{\mathrm{r}} c w_2^{\mathrm{r}} \mid w_1, w_2 \in \{a,b\}^*, |w_1| = |w_2|\}$ cannot be recognized by a 1NPDA with an auxiliary tape on which $o(n)$ space is used [77].

Theorem 8.7. *$L_{twin-rev}$ is recognized by a RT-PPDA with negative one-sided error bound $\frac{2}{3}$.*

Proof. (Sketch) The members of the language mainly have three deterministic checks, each of which can be implemented by a RT-DPDA. Therefore, each deterministic check is selected with probability $\frac{1}{3}$. □

Theorem 8.8. *$L_{twin-rev}$ is recognized by a RT-QPDA with negative one-sided error bound $\frac{5}{8}$.*

Proof. (Sketch) The computation split equiprobably into two branches, say branch_1 and branch_2. Suppose the input is of the form $w_1 c w_2 c w_3 c w_4$. branch_1 (resp., branch_2) splits into two paths, say path_{11} and path_{12} (resp., path_{21} and path_{22}) with equal amplitude.

path_{11} (resp., path_{21}) checks the length of w_1 and w_2 (resp., w_3 and w_4). If fails, the input is rejected.

path_{12} (resp., path_{22}) checks the equality of w_1 and w_3^r (resp., w_2 and w_4^r). If fails, the input is rejected.

At the end of the computation, path_{11} and path_{12} (resp., path_{21} and path_{22}) makes a 2-way QFT with the distinguished target of an accepting case.

In a branch, if both paths succeed, the input is accepted with probability 1. On the other hand, if a path fails, then the input is accepted with probability at most $\frac{1}{4}$ in the branch. Therefore, the overall accepting probability for nonmembers can be at most $\frac{5}{8} (= \frac{1}{2} + \frac{1}{8})$. □

$$L_{ij-i\vee j} = \{a^i b^j c^k \mid i \neq j, (i = k \vee j = k), 0 \leq i, j, k\} \text{ is not in CFL.}$$

Theorem 8.9. *$L_{ij-i\vee j}$ can be recognized by a RT-Q1BCA with one-sided error bound $\frac{3}{4}$.*

Proof. Suppose that the input is of the form $a^i b^j c^k$ for some $i, j, k > 0$. The computation is split into two paths and one of them (resp., the other) write i (resp., j) on the counter. Then, both makes a 2-way QFT with the distinguished target of an rejecting case. If $i = j$, the input is rejected with probability 1. Otherwise, each path continuous the computation with probability $\frac{1}{4}$. In the remaining part, both paths decrease their counter value by 1 whenever reading a c and then enter an accepting case after reading \$. Since $i \neq j$, only the counter of a one path can be zero, which can only be occurred for the members of the language. □

Theorem 8.10. *Any language recognized by a RT-DkBCA is in $S^{=}_{\mathbb{Q}}$.*

Proof. (Sketch) We can design a GFA for any given RT-DkBCA. The values of the counters are represented by a single number, say c. Initially c is set to 1. Let p_i be the i^{th} smallest prime, where $1 \leq i \leq k$. If the value of the i^{th} counter is updated by 1 (resp., -1), then c is multiplied by p_i (resp., $\frac{1}{p_i}$). It can be easily verified that $c = 1$ if and only if the values of all counters are zeros. □

Lemma 8.11. *$L_{NH} \notin uS$ can be recognized by a RT-N1BCA.*

Corollary 8.12. *The class of languages recognized by RT-DkBCA is a proper subset of the class of the languages recognized by RT-NkBCA for any $k > 0$.*

Lemma 8.13. *L_{neq} can be recognized by a RT-N1BCA.*

Proof. At the beginning of the computation, the machine assumes that the number of a's is bigger than the number of b's, (or vice versa). Then, any nondeterministic path test the equality of the number of a's and the number of b's without counting some a's (or b's). □

Fact 8.14. *[78] L^*_{upal} cannot be recognized by any 1NkBCA, where $k > 0$.*

Lemma 8.15. *L^*_{upal} is recognized by a RT-Q1BCA(m) with negative one-sided error bound $\frac{1}{m}$, where $m > 1$.*

Proof. (Sketch) The computation is split into m paths, i.e. path_j $(1 \leq j \leq m)$, at the beginning of each a^+b^+ block. In path_j, the counter value is increased (resp., decreased) by j when reading a a (resp., b). After finishing the block, all paths make a m-way QFT, i.e., (i) the computation continuous on the distinguished target if symbol a is read and the input is rejected on the other targets; (ii) if symbol \$ is read, the input is accepted on the distinguished target. □

8.2. Promise Problems

In this section, we use the Hadamard transform (2-way QFT) with the following setup. Let v be a 2×1 column vector and H be the Hadamard transform i.e.

$$\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{pmatrix}. \tag{8.5}$$

If $v = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}$ or $v = \begin{pmatrix} \frac{-1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{pmatrix}$, then $v' = Hv = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ or $v' = Hv = \begin{pmatrix} -1 \\ 0 \end{pmatrix}$, respectively. On the other hand, If $v = \begin{pmatrix} \frac{-1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}$ or $v = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{pmatrix}$, then $v' = Hv = \begin{pmatrix} 0 \\ -1 \end{pmatrix}$ or $v' = Hv = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$, respectively.

Suppose that the computation is split into two paths with equal amplitude. In each path, a deterministic check is done and the amplitude is multiplied with -1 if the check fails. After that, two paths make the Hadamard transformation by assuming the

targets surely interfere. Thus, we can obtain the following logical relation:

path_1	path_2	target_1	target_2
1	1	1	0
0	0	-1	0
1	0	0	1
0	1	0	-1

(8.6)

In other words, if both paths succeed or fail, then target_1 appears with probability 1, and, in the other cases, target_2 appears with probability 1.

Consider the strings of the form $w_1cw_2cw_3cw_4$, where $w_{1\leq i\leq 4} \in \{a,b\}^*$ and $|w_1| = |w_2| = |w_3| = |w_4|$:

- If $w_1 = w_3^{\mathrm{r}} \wedge w_2 = w_4^{\mathrm{r}}$ or $w_1 \neq w_3^{\mathrm{r}} \wedge w_2 \neq w_4^{\mathrm{r}}$, then it is a member of language A;
- It is a member of B, otherwise.

Theorem 8.16. *A 1-rev-RT-QPDA can separate languages A and B with zero error.*

Proof. (Sketch) Since the length of strings w_i $(1 \leq i \leq 4)$ are equal, they can interfere after the Hadamard. □

8.3. Multihead Finite Automata

The following language can be recognized by 1QFAs with 1-head and 1PFAs with 3-heads for any negative one-sided error bound $\epsilon \in (0, \frac{1}{2})$,

$$\{a^{n_1}ba^{n_2}b\cdots ba^{n_k}ba^{n_k}b\cdots ba^{n_2}ba^{n_1} \mid n_i > 0 \text{ for any } i \in \{1,\ldots,k\}\}, \tag{8.7}$$

where $k \in \mathbb{Z}^+$.

9. CONCLUSION

In this thesis, we define a new kind of quantum Turing machine for space bounded computation and some of its restricted variants, i.e. quantum pushdown, counter, and finite automata. Moreover, we present many results related to sublogarithmic and linear space both in classical and quantum computation.

As seen from Section 3, a common open problem for most of the space-bounded quantum computational models is whether their computational powers decrease or not when restricted to unidirectional ones.

In [6–8], for any space-constructable $s \in \Omega(\log n)$, Watrous showed the equivalence of the classes $\mathrm{PrSPACE}_{\mathbb{X}}(s)$ and $\mathrm{PrQSPACE}_{\mathbb{X}}(s)$, where $\mathbb{X} \in \{\mathbb{Q}, \mathbb{A}\}$. For sublogarithmic space bounds, we have obtained the results:

$$\text{one-way-PrSPACE}(s) \subsetneq \text{one-way-PrQSPACE}(s), \quad \text{for any } s \in o(\log n) \tag{9.1}$$

and

$$\mathrm{PrSPACE}(s) \subsetneq \mathrm{PrQSPACE}(s), \quad \text{for any } s \in o(\log \log n). \tag{9.2}$$

For $s \in \Omega(\log \log n) \cap o(\log(n))$, it still remains as an open problem whether $\mathrm{PrSPACE}(s)$ is a proper subset of $\mathrm{PrQSPACE}(s)$. It is also interesting to ask whether the results presented by Watrous [6–8] can be extended for the classes defined by computable or arbitrary real numbers amplitudes.

Some of our results for the constant space-bounded computation together with previously known ones are shown in Figure 9.1, in which the containment hierarchy is defined from bottom to top such that dotted arrows indicate subset relationships and unbroken arrows represent the cases where it is known that the inclusion is proper. One natural question is which dotted arrows in the figure can be replaced by unbroken

arrows. Some other more specific questions are presented below.

Open Problem 16. Is L_{pal} or L_{lt} in one-way-BQSPACE(1)?

Open Problem 17. Is there any nonstochastic language in BQSPACE(1)?

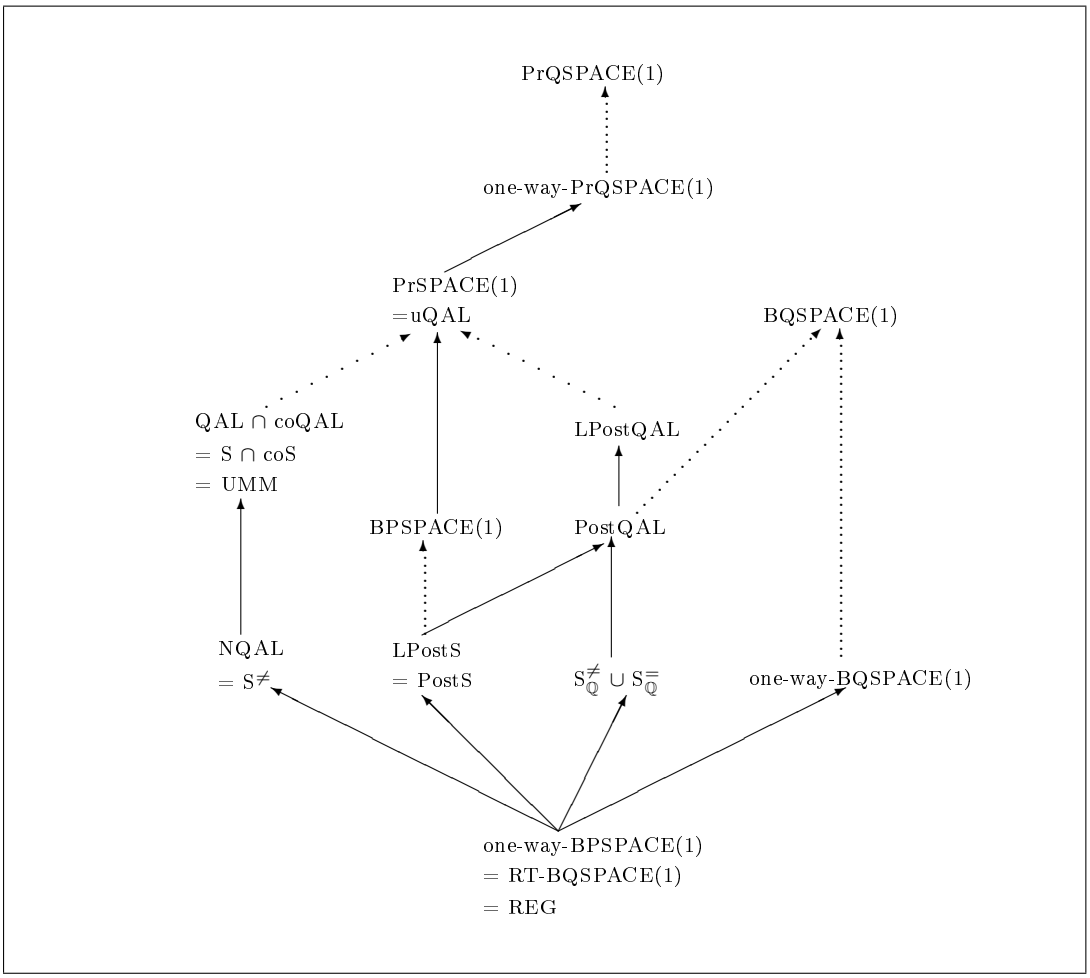

Figure 9.1. The relationships among classical and quantum constant space-bounded classes

In a more general way, the following question is also interesting.

Open Problem 18. Is there any hierarchy theorem for space-bounded quantum classes?

We also give a special emphasis to the quantum models with WOM (Chapter 6) since they can be seen as good steps to understand the nature of quantum computation, specifically the power of configuration interferences, which is impossible in classical computation.

Moreover, we also do not know of lower bounds for RT-QTMs and RT-PTMs in order to recognize a nonregular language. The same question can also be extended for CAs and PDAs by considering the space usage on their counters and stacks, respectively.

APPENDIX A: LOCAL CONDITIONS FOR QUANTUM MACHINES

A.1. Quantum Turing Machines

We present the list of the local conditions for QTM wellformedness in below: Let c_{j_1} and c_{j_2} be two configurations with corresponding columns v_{j_1} and v_{j_2} in E (See Figure 3.1). The value of $v_{j_1}[i]$ is determined by δ if the i^{th} entry of v_{j_1} corresponds to a configuration to which c_{j_1} can evolve in one step, and it is zero otherwise. Let x_1 and x_2 be the positions of the input tape head and y_1 and y_2 be the positions of the work tape head for the configurations c_{j_1} and c_{j_2}, respectively. In order to evolve to the same configuration in one step, the difference between x_1 and x_2 and/or y_1 and y_2 must be at most 2. Therefore, we obtain a total of 13 different cases, listed below, that completely define the restrictions on the transition function. Note that, by taking the conjugates of each summation, we handle the symmetric cases that are shown in the parentheses.

For all $q_1, q_2 \in Q$; $\sigma, \sigma_1, \sigma_2 \in \Sigma$; and $\gamma_1, \gamma_2, \gamma_3, \gamma_4 \in \Gamma$ (the summations are taken over $q' \in Q$; $\gamma' \in \Gamma$; $d, d_1, d_2 \in <>$; and $\omega \in \Omega$),

1. $x_1 = x_2$ and $y_1 = y_2$:

$$\sum_{q',\gamma',d_1,d_2,\omega} \overline{\delta(q_1,\sigma,\gamma_1,q',d_1,\gamma',d_2,\omega)}\delta(q_2,\sigma,\gamma_2,q',d_1,\gamma',d_2,\omega) = \begin{cases} 1 & q_1 = q_2, \gamma_1 = \gamma_2 \\ 0 & \text{otherwise} \end{cases} \tag{A.1}$$

2. $x_1 = x_2$ and $y_1 = y_2 - 1$ ($x_1 = x_2$ and $y_1 = y_2 + 1$):

$$\begin{aligned} \sum_{q',d,\omega} \quad & \overline{\delta(q_1,\sigma,\gamma_1,q',d,\gamma_2,\downarrow,\omega)}\delta(q_2,\sigma,\gamma_3,q',d,\gamma_4,\leftarrow,\omega) \\ + \quad & \overline{\delta(q_1,\sigma,\gamma_1,q',d,\gamma_2,\rightarrow,\omega)}\delta(q_2,\sigma,\gamma_3,q',d,\gamma_4,\downarrow,\omega) = 0 \end{aligned} \tag{A.2}$$

3. $x_1 = x_2$ and $y_1 = y_2 - 2$ ($x_1 = x_2$ and $y_1 = y_2 + 2$):

$$\sum_{q',d,\omega} \overline{\delta(q_1,\sigma,\gamma_1,q',d,\gamma_2,\rightarrow,\omega)}\delta(q_2,\sigma,\gamma_3,q',d,\gamma_4,\leftarrow,\omega) = 0 \tag{A.3}$$

4. $x_1 = x_2 - 1$ and $y_1 = y_2$ ($x_1 = x_2 + 1$ and $y_1 = y_2$):

$$\begin{aligned}\sum_{q',\gamma',d,\omega} & \overline{\delta(q_1,\sigma_1,\gamma_1,q',\downarrow,\gamma',d,\omega)}\delta(q_2,\sigma_2,\gamma_2,q',\leftarrow,\gamma',d,\omega) \\ + & \overline{\delta(q_1,\sigma_1,\gamma_1,q',\rightarrow,\gamma',d,\omega)}\delta(q_2,\sigma_2,\gamma_2,q',\downarrow,\gamma',d,\omega) = 0\end{aligned} \tag{A.4}$$

5. $x_1 = x_2 - 1$ and $y_1 = y_2 - 1$ ($x_1 = x_2 + 1$ and $y_1 = y_2 + 1$):

$$\begin{aligned}\sum_{q',\omega} & \overline{\delta(q_1,\sigma_1,\gamma_1,q',\downarrow,\gamma_2,\downarrow,\omega)}\delta(q_2,\sigma_2,\gamma_3,q',\leftarrow,\gamma_4,\leftarrow,\omega) \\ + & \overline{\delta(q_1,\sigma_1,\gamma_1,q',\downarrow,\gamma_2,\rightarrow,\omega)}\delta(q_2,\sigma_2,\gamma_3,q',\leftarrow,\gamma_4,\downarrow,\omega) \\ + & \overline{\delta(q_1,\sigma_1,\gamma_1,q',\rightarrow,\gamma_2,\downarrow,\omega)}\delta(q_2,\sigma_2,\gamma_3,q',\downarrow,\gamma_4,\leftarrow,\omega) \\ + & \overline{\delta(q_1,\sigma_1,\gamma_1,q',\rightarrow,\gamma_2,\rightarrow,\omega)}\delta(q_2,\sigma_2,\gamma_3,q',\downarrow,\gamma_4,\downarrow,\omega) = 0\end{aligned} \tag{A.5}$$

6. $x_1 = x_2 - 1$ and $y_1 = y_2 + 1$ ($x_1 = x_2 + 1$ and $y_1 = y_2 - 1$):

$$\begin{aligned}\sum_{q',\omega} & \overline{\delta(q_1,\sigma_1,\gamma_1,q',\downarrow,\gamma_2,\downarrow,\omega)}\delta(q_2,\sigma_2,\gamma_3,q',\leftarrow,\gamma_4,\rightarrow,\omega) \\ + & \overline{\delta(q_1,\sigma_1,\gamma_1,q',\downarrow,\gamma_2,\leftarrow,\omega)}\delta(q_2,\sigma_2,\gamma_3,q',\leftarrow,\gamma_4,\downarrow,\omega) \\ + & \overline{\delta(q_1,\sigma_1,\gamma_1,q',\rightarrow,\gamma_2,\downarrow,\omega)}\delta(q_2,\sigma_2,\gamma_3,q',\downarrow,\gamma_4,\rightarrow,\omega) \\ + & \overline{\delta(q_1,\sigma_1,\gamma_1,q',\rightarrow,\gamma_2,\leftarrow,\omega)}\delta(q_2,\sigma_2,\gamma_3,q',\downarrow,\gamma_4,\downarrow,\omega) = 0\end{aligned} \tag{A.6}$$

7. $x_1 = x_2 - 1$ and $y_1 = y_2 - 2$ ($x_1 = x_2 + 1$ and $y_1 = y_2 + 2$):

$$\begin{aligned}\sum_{q',\omega} & \overline{\delta(q_1,\sigma_1,\gamma_1,q',\downarrow,\gamma_2,\rightarrow,\omega)}\delta(q_2,\sigma_2,\gamma_3,q',\leftarrow,\gamma_4,\leftarrow,\omega) \\ + & \overline{\delta(q_1,\sigma_1,\gamma_1,q',\rightarrow,\gamma_2,\rightarrow,\omega)}\delta(q_2,\sigma_2,\gamma_3,q',\downarrow,\gamma_4,\leftarrow,\omega) = 0\end{aligned} \tag{A.7}$$

8. $x_1 = x_2 - 1$ and $y_1 = y_2 + 2$ ($x_1 = x_2 + 1$ and $y_1 = y_2 - 2$):

$$\begin{aligned}\sum_{q',\omega} & \overline{\delta(q_1,\sigma_1,\gamma_1,q',\downarrow,\gamma_2,\leftarrow,\omega)}\delta(q_2,\sigma_2,\gamma_3,q',\leftarrow,\gamma_4,\rightarrow,\omega) \\ + & \overline{\delta(q_1,\sigma_1,\gamma_1,q',\rightarrow,\gamma_2,\leftarrow,\omega)}\delta(q_2,\sigma_2,\gamma_3,q',\downarrow,\gamma_4,\rightarrow,\omega) = 0\end{aligned} \tag{A.8}$$

9. $x_1 = x_2 - 2$ and $y_1 = y_2$ ($x_1 = x_2 + 2$ and $y_1 = y_2$):

$$\sum_{q',\gamma',d,\omega} \overline{\delta(q_1,\sigma_1,\gamma_1,q',\rightarrow,\gamma',d,\omega)}\delta(q_2,\sigma_2,\gamma_2,q',\leftarrow,\gamma',d,\omega) \; = 0 \tag{A.9}$$

10. $x_1 = x_2 - 2$ and $y_1 = y_2 - 1$ ($x_1 = x_2 + 2$ and $y_1 = y_2 + 1$):

$$\begin{aligned}\sum_{q',\omega} & \overline{\delta(q_1,\sigma_1,\gamma_1,q',\rightarrow,\gamma_2,\rightarrow,\omega)}\delta(q_2,\sigma_2,\gamma_3,q',\leftarrow,\gamma_4,\downarrow,\omega) \\ + & \overline{\delta(q_1,\sigma_1,\gamma_1,q',\rightarrow,\gamma_2,\downarrow,\omega)}\delta(q_2,\sigma_2,\gamma_3,q',\leftarrow,\gamma_4,\leftarrow,\omega) = 0\end{aligned} \tag{A.10}$$

11. $x_1 = x_2 - 2$ and $y_1 = y_2 + 1$ ($x_1 = x_2 + 2$ and $y_1 = y_2 - 1$):

$$\begin{aligned}\sum_{q',\omega} & \overline{\delta(q_1,\sigma_1,\gamma_1,q',\rightarrow,\gamma_2,\leftarrow,\omega)}\delta(q_2,\sigma_2,\gamma_3,q',\leftarrow,\gamma_4,\downarrow,\omega) \\ + & \overline{\delta(q_1,\sigma_1,\gamma_1,q',\rightarrow,\gamma_2,\downarrow,\omega)}\delta(q_2,\sigma_2,\gamma_3,q',\leftarrow,\gamma_4,\rightarrow,\omega) = 0\end{aligned} \tag{A.11}$$

12. $x_1 = x_2 - 2$ and $y_1 = y_2 - 2$ ($x_1 = x_2 + 2$ and $y_1 = y_2 + 2$):

$$\sum_{q',\omega} \overline{\delta(q_1, \sigma_1, \gamma_1, q', \rightarrow, \gamma_2, \rightarrow, \omega)} \delta(q_2, \sigma_2, \gamma_3, q', \leftarrow, \gamma_4, \leftarrow, \omega) = 0 \tag{A.12}$$

13. $x_1 = x_2 - 2$ and $y_1 = y_2 + 2$ ($x_1 = x_2 + 2$ and $y_1 = y_2 - 2$):

$$\sum_{q',\omega} \overline{\delta(q_1, \sigma_1, \gamma_1, q', \rightarrow, \gamma_2, \leftarrow, \omega)} \delta(q_2, \sigma_2, \gamma_3, q', \leftarrow, \gamma_4, \rightarrow, \omega) = 0 \tag{A.13}$$

A.2. Quantum Pushdown Automata

Local conditions for 2QPDA well-formedness:

Let x_i and x_j be the positions of the input tape head and y_i and y_j be the positions of the stack head for the configurations c_i and c_j, respectively. In order to evolve to the same configuration in one step, the difference between x_i and x_j and/or y_i and y_j must be at most 2. Therefore, we obtain totally 13 different cases that completely define the restrictions on the transition function. Note that, by taking conjugate of each summation, we obtain the symmetric cases that are shown in the parenthesis.

For each choice of $q_1, q_2 \in Q$; $\sigma, \sigma_1, \sigma_2 \in \tilde{\Sigma}$; and $\gamma_1, \gamma_2, \gamma_3, \gamma_4 \in \Gamma$ (the summations are taken over $q' \in Q$; $\gamma' \in \tilde{\Gamma}$; $d, d_1, d_2 \in <>$; and $\omega \in \Omega$),

1. $x_i = x_j$ and $y_i = y_j$:

$$\sum_{q',d_1,d_2,\gamma',\omega} \overline{\delta(q_1, \sigma, \gamma_1, q', d_1, d_2, \gamma', \omega)} \delta(q_2, \sigma, \gamma_2, q', d_1, d_2, \gamma', \omega) = \begin{cases} 1 & q_1 = q_2, \gamma_1 = \gamma_2 \\ 0 & \text{otherwise} \end{cases} \tag{A.14}$$

2. $x_i = x_j$ and $y_i = y_j - 1$ ($x_i = x_j$ and $y_i = y_j + 1$):

$$\begin{aligned} \sum_{q',d,\omega} & \overline{\delta(q_1, \sigma, \gamma_1, q', d, \downarrow, \gamma_2, \omega)} \delta(q_2, \sigma, \gamma_3, q', d, \leftarrow, \gamma_3, \omega) \\ + & \overline{\delta(q_1, \sigma, \gamma_1, q', d, \rightarrow, \gamma_2, \omega)} \delta(q_2, \sigma, \gamma_3, q', d, \downarrow, \gamma_2, \omega) = 0 \end{aligned} \tag{A.15}$$

3. $x_i = x_j$ and $y_i = y_j - 2$ ($x_i = x_j$ and $y_i = y_j + 2$):

$$\sum_{q',d,\omega} \overline{\delta(q_1, \sigma, \gamma_1, q', d, \rightarrow, \gamma_2, \omega)} \delta(q_2, \sigma, \gamma_3, q', d, \leftarrow, \gamma_3, \omega) = 0 \tag{A.16}$$

4. $x_i = x_j - 1$ and $y_i = y_j$ ($x_i = x_j + 1$ and $y_i = y_j$):

$$\begin{aligned}\sum_{q',d,\gamma',\omega} & \overline{\delta(q_1,\sigma_1,\gamma_1,q',\downarrow,d,\gamma',\omega)}\delta(q_2,\sigma_2,\gamma_2,q',\leftarrow,d,\gamma',\omega) \\ & + \overline{\delta(q_1,\sigma_1,\gamma_1,q',\rightarrow,d,\gamma',\omega)}\delta(q_2,\sigma_2,\gamma_2,q',\downarrow,d,\gamma',\omega) = 0\end{aligned} \tag{A.17}$$

5. $x_i = x_j - 1$ and $y_i = y_j - 1$ ($x_i = x_j + 1$ and $y_i = y_j + 1$):

$$\begin{aligned}\sum_{q',\omega} & \overline{\delta(q_1,\sigma_1,\gamma_1,q',\downarrow,\downarrow,\gamma_2,\omega)}\delta(q_2,\sigma_2,\gamma_3,q',\leftarrow,\leftarrow,\gamma_3,\omega) \\ & + \overline{\delta(q_1,\sigma_1,\gamma_1,q',\downarrow,\rightarrow,\gamma_2,\omega)}\delta(q_2,\sigma_2,\gamma_3,q',\leftarrow,\downarrow,\gamma_2,\omega) \\ & + \overline{\delta(q_1,\sigma_1,\gamma_1,q',\rightarrow,\downarrow,\gamma_2,\omega)}\delta(q_2,\sigma_2,\gamma_3,q',\downarrow,\leftarrow,\gamma_3,\omega) \\ & + \overline{\delta(q_1,\sigma_1,\gamma_1,q',\rightarrow,\rightarrow,\gamma_2,\omega)}\delta(q_2,\sigma_2,\gamma_3,q',\downarrow,\downarrow,\gamma_2,\omega) = 0\end{aligned} \tag{A.18}$$

6. $x_i = x_j - 1$ and $y_i = y_j + 1$ ($x_i = x_j + 1$ and $y_i = y_j - 1$):

$$\begin{aligned}\sum_{q',\omega} & \overline{\delta(q_1,\sigma_1,\gamma_1,q',\downarrow,\downarrow,\gamma_2,\omega)}\delta(q_2,\sigma_2,\gamma_3,q',\leftarrow,\rightarrow,\gamma_2,\omega) \\ & + \overline{\delta(q_1,\sigma_1,\gamma_1,q',\downarrow,\leftarrow,\gamma_1,\omega)}\delta(q_2,\sigma_2,\gamma_3,q',\leftarrow,\downarrow,\gamma_4,\omega) \\ & + \overline{\delta(q_1,\sigma_1,\gamma_1,q',\rightarrow,\downarrow,\gamma_2,\omega)}\delta(q_2,\sigma_2,\gamma_3,q',\downarrow,\rightarrow,\gamma_2,\omega) \\ & + \overline{\delta(q_1,\sigma_1,\gamma_1,q',\rightarrow,\leftarrow,\gamma_1,\omega)}\delta(q_2,\sigma_2,\gamma_3,q',\downarrow,\downarrow,\gamma_4,\omega) = 0\end{aligned} \tag{A.19}$$

7. $x_i = x_j - 1$ and $y_i = y_j - 2$ ($x_i = x_j + 1$ and $y_i = y_j + 2$):

$$\begin{aligned}\sum_{q',\omega} & \overline{\delta(q_1,\sigma_1,\gamma_1,q',\downarrow,\rightarrow,\gamma_2,\omega)}\delta(q_2,\sigma_2,\gamma_3,q',\leftarrow,\leftarrow,\gamma_3,\omega) \\ & + \overline{\delta(q_1,\sigma_1,\gamma_1,q',\rightarrow,\rightarrow,\gamma_2,\omega)}\delta(q_2,\sigma_2,\gamma_3,q',\downarrow,\leftarrow,\gamma_3,\omega) = 0\end{aligned} \tag{A.20}$$

8. $x_i = x_j - 1$ and $y_i = y_j + 2$ ($x_i = x_j + 1$ and $y_i = y_j - 2$):

$$\begin{aligned}\sum_{q',\omega} & \overline{\delta(q_1,\sigma_1,\gamma_1,q',\downarrow,\leftarrow,\gamma_1,\omega)}\delta(q_2,\sigma_2,\gamma_3,q',\leftarrow,\rightarrow,\gamma_4,\omega) \\ & + \overline{\delta(q_1,\sigma_1,\gamma_1,q',\rightarrow,\leftarrow,\gamma_1,\omega)}\delta(q_2,\sigma_2,\gamma_3,q',\downarrow,\rightarrow,\gamma_4,\omega) = 0\end{aligned} \tag{A.21}$$

9. $x_i = x_j - 2$ and $y_i = y_j$ ($x_i = x_j + 2$ and $y_i = y_j$):

$$\sum_{q',d,\gamma',\omega} \overline{\delta(q_1,\sigma_1,\gamma_1,q',\rightarrow,d,\gamma',\omega)}\delta(q_2,\sigma_2,\gamma_2,q',\leftarrow,d,\gamma',\omega) \ = 0 \tag{A.22}$$

10. $x_i = x_j - 2$ and $y_i = y_j - 1$ ($x_i = x_j + 2$ and $y_i = y_j + 1$):

$$\begin{aligned}\sum_{q',\omega} & \overline{\delta(q_1,\sigma_1,\gamma_1,q',\rightarrow,\rightarrow,\gamma_2,\omega)}\delta(q_2,\sigma_2,\gamma_3,q',\leftarrow,\downarrow,\gamma_2,\omega) \\ & + \overline{\delta(q_1,\sigma_1,\gamma_1,q',\rightarrow,\downarrow,\gamma_2,\omega)}\delta(q_2,\sigma_2,\gamma_3,q',\leftarrow,\leftarrow,\gamma_3,\omega) = 0\end{aligned} \tag{A.23}$$

11. $x_i = x_j - 2$ and $y_i = y_j + 1$ ($x_i = x_j + 2$ and $y_i = y_j - 1$):

$$\begin{aligned}\sum_{q',\omega} \quad & \overline{\delta(q_1,\sigma_1,\gamma_1,q',\rightarrow,\leftarrow,\gamma_1,\omega)}\delta(q_2,\sigma_2,\gamma_3,q',\leftarrow,\downarrow,\gamma_4,\omega) \\ + \quad & \overline{\delta(q_1,\sigma_1,\gamma_1,q',\rightarrow,\downarrow,\gamma_2,\omega)}\delta(q_2,\sigma_2,\gamma_3,q',\leftarrow,\rightarrow,\gamma_2,\omega) = 0\end{aligned} \tag{A.24}$$

12. $x_i = x_j - 2$ and $y_i = y_j - 2$ ($x_i = x_j + 2$ and $y_i = y_j + 2$):

$$\sum_{q',\omega} \overline{\delta(q_1,\sigma_1,\gamma_1,q',\rightarrow,\rightarrow,\gamma_2,\omega)}\delta(q_2,\sigma_2,\gamma_3,q',\leftarrow,\leftarrow,\gamma_3,\omega) = 0 \tag{A.25}$$

13. $x_i = x_j - 2$ and $y_i = y_j + 2$ ($x_i = x_j + 2$ and $y_i = y_j - 2$):

$$\sum_{q',\omega} \overline{\delta(q_1,\sigma_1,\gamma_1,q',\rightarrow,\leftarrow,\gamma_1,\omega)}\delta(q_2,\sigma_2,\gamma_3,q',\leftarrow,\rightarrow,\gamma_4,\omega) = 0 \tag{A.26}$$

Local conditions for 1QPDA well-formedness:

In order to evolve to the same configuration in one step, the difference between x_i and x_j must be at most 1 in case of 1QPDA. Therefore, we obtain totally 8 different cases that completely define the restrictions on the transition function. Similar to 2QPDA, by taking conjugate of each summation, we obtain the symmetric cases that are shown in the parenthesis.

For each choice of $q_1, q_2 \in Q$; $\sigma, \sigma_1, \sigma_2 \in \tilde{\Sigma}$; and $\gamma_1, \gamma_2, \gamma_3, \gamma_4 \in \Gamma$ (the summations are taken over $q' \in Q$; $\gamma' \in \tilde{\Gamma}$; $d \in \rhd$, $e \in <>$; and $\omega \in \Omega$),

1. $x_i = x_j$ and $y_i = y_j$:

$$\sum_{q',d,e,\gamma',\omega} \overline{\delta(q_1,\sigma,\gamma_1,q',d,e,\gamma',\omega)}\delta(q_2,\sigma,\gamma_2,q',d,e,\gamma',\omega) = \begin{cases} 1 & q_1 = q_2, \gamma_1 = \gamma_2 \\ 0 & \text{otherwise} \end{cases} \tag{A.27}$$

2. $x_i = x_j$ and $y_i = y_j - 1$ ($x_i = x_j$ and $y_i = y_j + 1$):

$$\begin{aligned}\sum_{q',d,\omega} \quad & \overline{\delta(q_1,\sigma,\gamma_1,q',d,\downarrow,\gamma_2,\omega)}\delta(q_2,\sigma,\gamma_3,q',d,\leftarrow,\gamma_3,\omega) \\ + \quad & \overline{\delta(q_1,\sigma,\gamma_1,q',d,\rightarrow,\gamma_2,\omega)}\delta(q_2,\sigma,\gamma_3,q',d,\downarrow,\gamma_2,\omega) = 0\end{aligned} \tag{A.28}$$

3. $x_i = x_j$ and $y_i = y_j - 2$ ($x_i = x_j$ and $y_i = y_j + 2$):

$$\sum_{q',d,\omega} \overline{\delta(q_1,\sigma,\gamma_1,q',d,\rightarrow,\gamma_2,\omega)}\delta(q_2,\sigma,\gamma_3,q',d,\leftarrow,\gamma_3,\omega) = 0 \tag{A.29}$$

4. $x_i = x_j - 1$ and $y_i = y_j$ ($x_i = x_j + 1$ and $y_i = y_j$):

$$\sum_{q',e,\gamma',\omega} \overline{\delta(q_1,\sigma_1,\gamma_1,q',\rightarrow,e,\gamma',\omega)}\delta(q_2,\sigma_2,\gamma_2,q',\downarrow,e,\gamma',\omega) = 0 \tag{A.30}$$

5. $x_i = x_j - 1$ and $y_i = y_j - 1$ ($x_i = x_j + 1$ and $y_i = y_j + 1$):

$$\begin{aligned}\sum_{q',\omega} & \overline{\delta(q_1,\sigma_1,\gamma_1,q',\rightarrow,\downarrow,\gamma_2,\omega)}\delta(q_2,\sigma_2,\gamma_3,q',\downarrow,\leftarrow,\gamma_3,\omega) \\ + & \overline{\delta(q_1,\sigma_1,\gamma_1,q',\rightarrow,\rightarrow,\gamma_2,\omega)}\delta(q_2,\sigma_2,\gamma_3,q',\downarrow,\downarrow,\gamma_2,\omega) = 0\end{aligned} \tag{A.31}$$

6. $x_i = x_j - 1$ and $y_i = y_j + 1$ ($x_i = x_j + 1$ and $y_i = y_j - 1$):

$$\begin{aligned}\sum_{q',\omega} & \overline{\delta(q_1,\sigma_1,\gamma_1,q',\rightarrow,\downarrow,\gamma_2,\omega)}\delta(q_2,\sigma_2,\gamma_3,q',\downarrow,\rightarrow,\gamma_2,\omega) \\ + & \overline{\delta(q_1,\sigma_1,\gamma_1,q',\rightarrow,\leftarrow,\gamma_1,\omega)}\delta(q_2,\sigma_2,\gamma_3,q',\downarrow,\downarrow,\gamma_4,\omega) = 0\end{aligned} \tag{A.32}$$

7. $x_i = x_j - 1$ and $y_i = y_j - 2$ ($x_i = x_j + 1$ and $y_i = y_j + 2$):

$$\sum_{q',\omega} \overline{\delta(q_1,\sigma_1,\gamma_1,q',\rightarrow,\rightarrow,\gamma_2,\omega)}\delta(q_2,\sigma_2,\gamma_3,q',\downarrow,\leftarrow,\gamma_3,\omega) = 0 \tag{A.33}$$

8. $x_i = x_j - 1$ and $y_i = y_j + 2$ ($x_i = x_j + 1$ and $y_i = y_j - 2$):

$$\sum_{q',\omega} \overline{\delta(q_1,\sigma_1,\gamma_1,q',\rightarrow,\leftarrow,\gamma_1,\omega)}\delta(q_2,\sigma_2,\gamma_3,q',\downarrow,\rightarrow,\gamma_4,\omega) = 0 \tag{A.34}$$

Local conditions for RT-QPDA well-formedness:

For RT-QPDA, we only consider the stack tape head positions. In order to evolve to the same configuration in one step, the difference between y_i and y_j must be at most 2. Therefore, we obtain totally 3 different cases that completely define the restrictions on the transition function. Similar to the previous ones, by taking conjugate of each summation, we obtain the symmetric cases that are shown in the parenthesis.

For each choice of $q_1, q_2 \in Q$; $\sigma, \sigma_1, \sigma_2 \in \tilde{\Sigma}$; and $\gamma_1, \gamma_2, \gamma_3, \gamma_4 \in \Gamma$ (the summations are taken over $q' \in Q$; $\gamma' \in \tilde{\Gamma}$; $e \in <>$; and $\omega \in \Omega$),

1. $x_i = x_j$ and $y_i = y_j$:

$$\sum_{q',e,\gamma',\omega} \overline{\delta(q_1,\sigma,\gamma_1,q',e,\gamma',\omega)}\delta(q_2,\sigma,\gamma_2,q',e,\gamma',\omega) = \begin{cases} 1 & q_1 = q_2, \gamma_1 = \gamma_2 \\ 0 & \text{otherwise} \end{cases} \tag{A.35}$$

2. $x_i = x_j$ and $y_i = y_j - 1$ ($x_i = x_j$ and $y_i = y_j + 1$):

$$\begin{aligned} \sum_{q',\omega} \quad & \overline{\delta(q_1, \sigma, \gamma_1, q', \downarrow, \gamma_2, \omega)} \delta(q_2, \sigma, \gamma_3, q', \leftarrow, \gamma_3, \omega) \\ + \quad & \overline{\delta(q_1, \sigma, \gamma_1, q', \rightarrow, \gamma_2, \omega)} \delta(q_2, \sigma, \gamma_3, q', \downarrow, \gamma_2, \omega) = 0 \end{aligned} \tag{A.36}$$

3. $x_i = x_j$ and $y_i = y_j - 2$ ($x_i = x_j$ and $y_i = y_j + 2$):

$$\sum_{q',\omega} \quad \overline{\delta(q_1, \sigma, \gamma_1, q', \rightarrow, \gamma_2, \omega)} \delta(q_2, \sigma, \gamma_3, q', \leftarrow, \gamma_3, \omega) = 0 \tag{A.37}$$

APPENDIX B: QUANTUM COMPUTATION

We review some basic concepts related to quantum computation in this appendix. We refer the reader to [32] for a complete reference. We focus on finite dimensional systems.

The *state space* of a quantum system (A) with n *classical states*, $Q = \{q_i \mid 1 \leq i \leq n\}$, is an n-dimensional Hilbert space[29] , denoted as $\mathcal{H}_A$. The set $\mathcal{B}_A = \{|q_i\rangle \mid 1 \leq i \leq n\}$ is an orthonormal basis for $\mathcal{H}_A$, where the i^{th} entry of $|q_i\rangle$ is 1 and the remaining ones are zeros. If isolated, A is completely described by its *state vector*, say $|\psi\rangle$, that is a linear combination of *basis states*[30]

$$|\psi\rangle = \alpha_1|q_1\rangle + \ldots + \alpha_n|q_n\rangle, \tag{B.1}$$

where α_i is the amplitude of $|q_i\rangle$, whose modulus squared, $|\alpha_i|^2$, is the probability of being in state q_i, and $\sum_i |\alpha_i|^2 = 1$ $(1 \leq i \leq n)$. When $|\psi\rangle$ contains more than one basis state with nonzero amplitude, then it is said that A is in a *superposition* (of the corresponding basis states).

The evolution of an isolated quantum system, i.e. closed quantum system, is governed by unitary transformations, which are norm preserving operators. If a unitary transformation, say U, is applied on state $|\psi\rangle$, then the new state is obtained by

$$|\psi'\rangle = U|\psi\rangle. \tag{B.2}$$

To retrieve the information from a quantum system (A), it is measured by a quantum operator (in physical sense, it is observed by a measurement apparatus). Throughout the thesis, Von Neumann measurements[31] are used. That is, the set of the

[29] It is a complex vector space with inner product.
[30] We fixed it as $\mathcal{B}_A$. However, note that, one can also use different orthonormal basis.
[31] They are a special case of projective measurements.

classical states (Q) are divided into some pairwise-disjoint subsets, $Q = Q_1 + \cdots + Q_k$ and so $\mathcal{H}_A$ becomes a composition of mutually orthogonal subspaces, $\mathcal{H}_{A_1}, \ldots, \mathcal{H}_{A_k}$, i.e.

$$\mathcal{H} = \mathcal{H}_{A_1} \oplus \cdots \oplus \mathcal{H}_{A_k} \tag{B.3}$$

and

$$\mathcal{H}_{A_i} = \text{span}\{|q\rangle \mid q \in Q_i\}, \tag{B.4}$$

where $1 \leq i \leq k$. Thus, we define Von Neumann measurement P with k outcomes as follows: P is composed of a list of measurement operators, or specifically orthogonal projectors, $P_1, \ldots, P_k$, defined as

$$P = \{P_i \mid P_i = \sum_{q \in Q_i} |q\rangle\langle q|, 1 \leq i \leq k\}. \tag{B.5}$$

If A is measured by P when it is in state $|\psi\rangle$, the outcome $i \in \{1, \ldots, k\}$ is obtained with probability

$$|||\psi_i\rangle||^2 = \langle\psi_i|\psi_i\rangle = \langle\psi|P_i|\psi\rangle, \tag{B.6}$$

where $|\psi_i\rangle = P_i|\psi\rangle$, and then the system collapses to state

$$\frac{|\psi_i\rangle}{\sqrt{\langle\psi_i|\psi_i\rangle}}, \tag{B.7}$$

which is normalized.

Let A and B be quantum systems with sets of the classical states $Q = \{q_1, \ldots, q_n\}$ and $R = \{r_1, \ldots, r_m\}$, respectively. $A \otimes B$, or shortly AB, denotes the joint system composed by A and B. The state space of AB is denoted by $\mathcal{H}_{AB}$, which is obtained by $\mathcal{H}_A \otimes \mathcal{H}_B$. Similarly, $\mathcal{B}_{AB} = \mathcal{B}_A \otimes \mathcal{B}_B$, i.e. $\mathcal{B}_{AB} = \{|q_i r_j\rangle = |q_i\rangle|r_j\rangle \mid 1 \leq i \leq n, 1 \leq j \leq$

$m\}$. If A is in state $|\psi_A\rangle$ and B is in state $|\psi_B\rangle$, then AB is in state $|\psi_{AB}\rangle = |\psi_A\rangle|\psi_B\rangle$. However, it is not always possible for a joint system (AB) to be in a state that is a composition of two separate states belonging to the participant subsystems (A and B, respectively). This situation is named as the *entanglement* of two subsystems.

In some cases, a quantum system can be in more than one state with some probabilities, i.e. $\{(p_l, |\psi_l\rangle) \mid 1 < l \leq M < \infty\}$, where p_l is the probability of being in state $|\psi_l\rangle$, $\sum_l p_l = 1$ and $\langle\psi_l|\psi_l\rangle = 1$. If the system is in exactly one state, then the system is said to be in *pure state*, but the cases in which the system is in more then one pure state with some probabilities, an ensemble of pure states, are said to be in *mixed states*. A convenient representation tool describing the state of a quantum system in mixed states is the density matrix. The *density matrix*[32] representation of $\{(p_l, |\psi_l\rangle) \mid 1 \leq l \leq M < \infty\}$ is as follows:

$$\rho = \sum_l p_l |\psi_l\rangle\langle\psi_l|. \tag{B.8}$$

If the system is observed with measurement $P = \{P_i \mid 1 \leq i \leq k\}$, then the outcome i is obtained with probability

$$tr(\tilde{\rho}_i), \tag{B.9}$$

where

$$\tilde{\rho}_i = P_i \rho P_i^\dagger, \tag{B.10}$$

and so the corresponding density matrix becomes

$$\rho_i = \frac{\tilde{\rho}_i}{tr(\tilde{\rho}_i)}. \tag{B.11}$$

In fact, the mixture is replaced by one of these sub-mixtures with the corresponding

[32]The trace of a density matrix is 1 and each density matrix is positive semidefinite.

probability after the measurement. If the measurement is *non-selective*, i.e. there is no distinction between the outcomes, the new state of the system can be seen as a mixture of sub-mixtures

$$\{(tr(\tilde{\rho}_i), \frac{\tilde{\rho}_i}{tr(\tilde{\rho}_i)}) \mid 1 \leq i \leq k\} \tag{B.12}$$

that forms the density matrix

$$\rho' = \sum_i \tilde{\rho}_i = \sum_i P_i \rho P_i^\dagger. \tag{B.13}$$

In the case of *selective* measurements, the new states (sub-mixtures) are kept separately. Here, each sub-mixture can be considered as the state of a sub-computation obtained by splitting the computation due to the measurement.

Note that, as seen from the above, $\tilde{\rho}_i$ alone represents $(tr(\tilde{\rho}_i), \rho_i)$. It may be helpful in some contexts to consider the unnormalized state representation[33] $(\tilde{\rho}_i)$, in which the probability of being in state, the trace of the matrix $(tr(\tilde{\rho}_i))$, exists unconditionally.

By a suitable setup, a quantum system, as a part of a bigger closed system which is governed by a unitary transformation, can evolve with respect to the general quantum operators, namely *superoperators*, that form a superset of both unitary and classical (stochastic) operators. That is, we compose the system of interest, *the principal system*, with a system, *an environment*, which is prepared in a specified state. The whole system evolves unitarily and then the environment is discarded, which is measured in some cases before discarding. If the environment is selectively measured, the operator applied on the principal system is called *selective quantum operator*. Otherwise, it is called *admissible operator*. The above setup is known as *open quantum systems*.

Specifically, in the case of admissible operators, if the principal system is in ρ and the environment is prepared in ρ_{env}, then a unitary operator, say U, is applied on $\rho \otimes \rho_{env}$. After that, the environment is discarded and new state becomes $\rho' = \mathcal{E}(\rho)$,

[33] It is still positive semidefinite but the trace may be less than 1.

where $\mathcal{E}$ is the admissible operator, a part of U, applied on the principal system. There are many equivalent representations of them. We follow *operator-sum representation*. An admissible operator $\mathcal{E}$ is a finite collection of matrices (operation elements) $\{E_i \mid 1 \leq i \leq m\}$ satisfying that

$$\sum_i E_i^\dagger E_i = I. \tag{B.14}$$

When admissible operator $\mathcal{E}$ is applied on a density matrix ρ, then the new density matrix is obtained by

$$\rho' = \mathcal{E}(\rho) = \sum_i E_i \rho E_i^\dagger. \tag{B.15}$$

One of the nice properties of the admissible operator is that when two of them are composed, we obtain a new one[34] . If $\mathcal{E} = \{E_i \mid 1 \leq i \leq m\}$ and $\mathcal{E}' = \{E_i' \mid 1 \leq j \leq m'\}$, then

$$\mathcal{E}' \circ \mathcal{E} = \{E_j' E_i \mid 1 \leq i \leq m, 1 \leq j \leq m'\}. \tag{B.16}$$

In case of selective quantum operators, the environment is measured before the discarding. In fact, selective quantum operators[35] are the operators in order to handle the case of the selective measurements. Let Δ be a finite set of outcomes. A selective quantum operator $\mathcal{E}$ is a collection of matrices $\{E_{\tau,i} \mid \tau \in \Delta, 1 \leq i \leq m_\tau\}$ satisfying that

$$\sum_{\tau,i} E_{\tau,i}^\dagger E_{\tau,i} = I. \tag{B.17}$$

We can classify the matrices of $\mathcal{E}$ with respect to the elements of the set of the outcomes:

$$\mathcal{E}_\tau = \{E_{\tau,i}\}. \tag{B.18}$$

[34]Note that, the composed operator is implemented probably with a bigger environment.
[35]We also refer to the reader to [8].

When selective quantum operator $\mathcal{E}$ is applied on a density matrix ρ, then the unnormalized matrix representing the sub-mixture $\tau \in \Delta$ is obtained by

$$\tilde{\rho_\tau} = \tilde{\mathcal{E}}_\tau(\rho) = \sum_i E_{\tau,i} \rho E_{\tau,i}^\dagger, \tag{B.19}$$

with probability $tr(\tilde{\rho_\tau})$. The normalized version is obtained as

$$\rho_\tau = \mathcal{E}_\tau(\rho) = \frac{\tilde{\rho_\tau}}{tr(\tilde{\rho_\tau})}. \tag{B.20}$$

The distinction between admissible and selective quantum operators as an issue of this thesis is that a space-bounded computation (which may go on forever), should be checked regularly to see whether it has halted or not. However, note that, if the computation is time bounded, all sub-mixtures are kept alive until the end of the computation, and so selective quantum operators can be replaced with admissible ones with the addition of some suitable measurements.

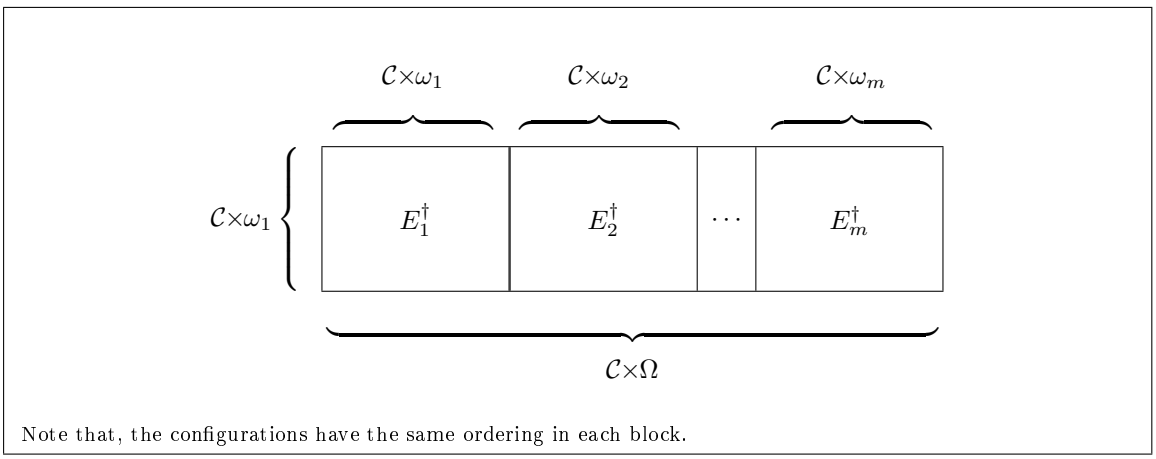

Figure B.1. The matrix obtained by deleting all rows of $U^\dagger$ except the ones in $\mathcal{C}\times\omega_1$.

In the remaining part, we present the general setup implementing superoperators. Let $\mathcal{H}_\mathcal{C}$ and $\mathcal{H}_\Omega$ be the principal and environment systems, respectively, where $\mathcal{C} = \{c_1, \ldots, c_n\}$ is the finite dimensional configuration set and $\Omega = \{\omega_1, \ldots, \omega_m\}$, having a special symbol ω_1, *the initial symbol*, is the set of symbols for a finite register. Suppose that $\mathcal{H}_\mathcal{C}$ is in (mixed) state ρ. We prepare $\mathcal{H}_\Omega$ with $|\omega_1\rangle$ and apply unitary operator U

on the joint system $\mathcal{H}_\mathcal{C} \otimes \mathcal{H}_\Omega$, then we discard the environment $\mathcal{H}_\Omega$. In Figure B.1, the parts of U effecting $\mathcal{H}_\mathcal{C}$, which form an admissible operator $\mathcal{E}$ with operation elements $\{E_1, \ldots, E_m\}$, are seen. A straightforward calculation can validate that $\mathcal{E}$ satisfies Equation B.14 and the new density matrix of $\mathcal{H}_\mathcal{C}$ is obtained as in Equation B.15.

In order to handle selective quantum operators, we add the following specifications to the *general setup*:

i. Ω is partitioned into $|\Delta| = k$ disjoint subsets, $\Omega_{\tau_1}, \ldots, \Omega_{\tau_k}$, i.e. $|\Omega_{\tau_i}| = m_i$, $1 \leq i \leq k$, and
ii. the environment is measured by $P = \{P_{\tau \in \Delta} \mid P_\tau = \sum_{\omega \in \Omega_\tau} |\omega\rangle\langle\omega|\}$ before being discarded.

Hence, the operation elements that are explicitly shown in Figure B.1 are grouped with respect to the partitioning of Ω (see Figure B.2), i.e., $\mathcal{E} = \{\mathcal{E}_\tau\}$ and $\mathcal{E}_\tau = \{E_i \mid \omega_i \in \Omega_\tau\} = \{E_{\tau,j} \mid 1 \leq j \leq |\Omega_\tau|\}$. Note that, the cardinality of Ω_τ's may not be the same.

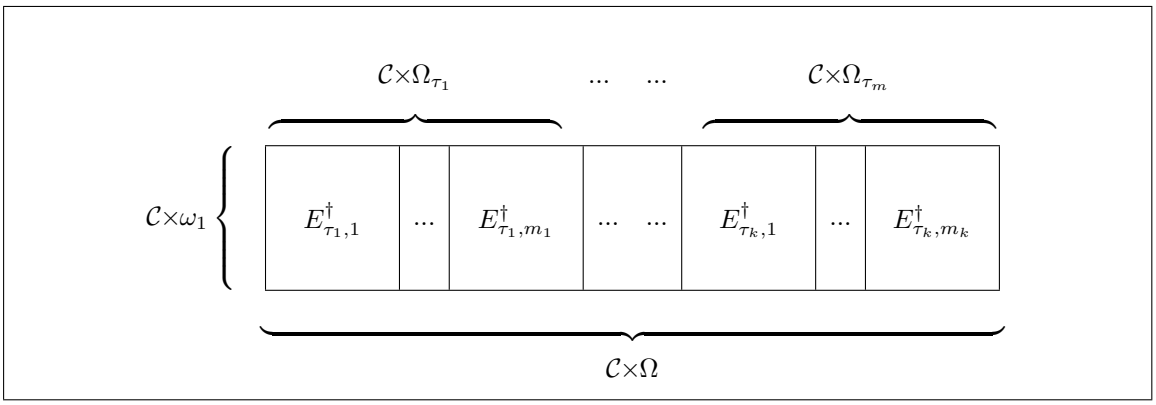

Figure B.2. Re-partitioning of the matrix in Figure B.1

REFERENCES

1. Gabarró, J., "Pushdown Space Complexity and Related Full-AFLs", *STACS'84: Proceedings of the Symposium of Theoretical Aspects of Computer Science*, pp. 250–259, 1984.

2. Szepietowski, A., *Turing Machines with Sublogarithmic Space*, Springer-Verlag, 1994.

3. Freivalds, R., "Probabilistic Two-Way Machines", *Proceedings of the International Symposium on Mathematical Foundations of Computer Science*, pp. 33–45, 1981.

4. Kondacs, A. and J. Watrous, "On the Power of Quantum Finite State Automata", *FOCS'97: Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, pp. 66–75, 1997.

5. 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.

6. Watrous, J., *Space-Bounded Quantum Computation*, Ph.D. thesis, University of Wisconsin - Madison, USA, 1998.

7. Watrous, J., "Space-bounded Quantum Complexity", *Journal of Computer and System Sciences*, Vol. 59, No. 2, pp. 281–326, 1999.

8. Watrous, J., "On the Complexity of Simulating Space-bounded Quantum Computations", *Computational Complexity*, Vol. 12, No. 1-2, pp. 48–84, 2003.

9. Brodsky, A. and N. Pippenger, "Characterizations of 1-Way Quantum Finite Automata", *SIAM Journal on Computing*, Vol. 31, No. 5, pp. 1456–1478, 2002.

10. Aaronson, S., "Quantum Computing, Postselection, and Probabilistic Polynomial-

time", *Proceedings of the Royal Society A*, Vol. 461, No. 2063, pp. 3473–3482, 2005.

11. Lāce, L., O. Scegulnaja-Dubrovska, and R. Freivalds, "Languages Recognizable by Quantum Finite Automata with Cut-point 0", *SOFSEM'09: Proceedings of the 35th International Conference on Current Trends in Theory and Practice of Computer Science*, Vol. 2, pp. 35–46, 2009.

12. Scegulnaja-Dubrovska, O., L. Lāce, and R. Freivalds, "Postselection Finite Quantum Automata", Vol. 6079 of *LNCS*, pp. 115–126, 2010.

13. Say, A. C. C. and A. Yakaryılmaz, "Quantum Function Computation Using Sublogarithmic Space", Poster presentation at the QIP2010 (arXiv:1009.3124), 2010.

14. Yakaryılmaz, A., R. Freivalds, A. C. C. Say, and R. Agadzanyan, "Quantum Computation with Devices Whose Contents Are Never Read", *Unconventional Computation*, Vol. 6079 of *LNCS*, pp. 164–174, 2010.

15. Rabin, M. O., "Probabilistic Automata", *Information and Control*, Vol. 6, pp. 230–243, 1963.

16. Blondel, V. D., E. Jeandel, P. Koiran, and N. Portier, "Decidable and Undecidable Problems about Quantum Automata", *SIAM Journal on Computing*, Vol. 34, No. 6, pp. 1464–1473, 2005.

17. Yakaryılmaz, A. and A. C. C. Say, "Unbounded-error Quantum Computation with Small Space Bounds", *Information and Computation*, accepted, 2011.

18. Bernstein, E. and U. Vazirani, "Quantum Complexity Theory", *SIAM Journal on Computing*, Vol. 26, No. 5, pp. 1411–1473, 1997.

19. Rabin, M. O., "Real Time Computation", *Israel Journal of Mathematics*, Vol. 1, No. 4, pp. 203–211, 1963.

20. Aanderra, S. O., "On k-Tape versus $(k-1)$-Tape Real Time Computation", Karp,

R. M. (editor), *SIAM AMS Proceedings*, Vol. 7 (Complexity of Computation), pp. 75–96, 1974.

21. Rabin, M. and D. Scott, "Finite Automata and Their Decision Problems", *IBM Journal of Research and Development*, Vol. 3, pp. 114–125, 1959.

22. Masaki Nakanishi, K. H., Takao Indoh, "On the Power of Quantum Pushdown Automata with a Classical Stack and 1.5-way Quantum Finite Automata", Technical Report NAIST-IS-TR2001005, Nara Institute of Science and Technology, http://isw3.naist.jp/IS/TechReport/report/2001005.ps (accessed in January 2011), 2001.

23. Yakaryılmaz, A. and A. C. C. Say, "Languages Recognized with Unbounded Error by Quantum Finite Automata", *CSR'09: Proceedings of the Fourth International Computer Science Symposium in Russia*, Vol. 5675 of *LNCS*, pp. 356–367, 2009.

24. 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, 1999.

25. Chan, T., "Reversal Complexity of Counter Machines", *STOC'81: Proceedings of the thirteenth annual ACM symposium on Theory of computing*, pp. 146–157, 1981.

26. Turakainen, P., "Generalized Automata and Stochastic Languages", *Proceedings of the American Mathematical Society*, Vol. 21, pp. 303–309, 1969.

27. 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.

28. Kaņeps, J., "Stochasticity of the Languages Acceptable by Two-Way Finite Probabilistic Automata", *Diskretnaya Matematika*, Vol. 1, pp. 63–67, 1989 (Russian).

29. Paz, A., *Introduction to Probabilistic Automata*, Academic Press, New York, 1971.

30. van Melkebeek, D. and T. Watson, "A Quantum Time-Space Lower Bound for the Counting Hierarchy", *Electronic Colloquium on Computational Complexity (ECCC)*, Vol. 15, No. 017, http://eccc.hpi-web.de/eccc-reports/2008/TR08-017/index.html (accessed in January 2011), 2008.

31. le Gall, F., "Exponential Separation of Quantum and Classical Online Space Complexity", *Theory of Computing Systems*, Vol. 45, No. 2, pp. 188–202, 2009.

32. Nielsen, M. A. and I. L. Chuang, *Quantum Computation and Quantum Information*, Cambridge University Press, 2000.

33. Aharonov, D., A. Kitaev, and N. Nisan, "Quantum Circuits with Mixed States", *STOC'98: Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, pp. 20–30, 1998.

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

35. Freivalds, R. and M. Karpinski, "Lower Space Bounds for Randomized Computation", *ICALP'94: Proceedings of the 21st International Colloquium on Automata, Languages and Programming*, pp. 580–592, 1994.

36. Hirvensalo, M., "Various Aspects of Finite Quantum Automata", *DLT'08: Proceedings of the 12th international conference on Developments in Language Theory*, pp. 21–33, 2008.

37. Moore, C. and J. P. Crutchfield, "Quantum Automata and Quantum Grammars", *Theoretical Computer Science*, Vol. 237, No. 1-2, pp. 275–306, 2000.

38. Li, L. and D. Qiu, "Determining the Equivalence for One-Way Quantum Finite Automata", *Theoretical Computer Science*, Vol. 403, No. 1, pp. 42–51, 2008.

39. Yakaryılmaz, A. and A. C. C. Say, "Language Recognition by Generalized Quantum Finite Automata with Unbounded Error", Poster presentation at the TQC2009

(arXiv:0901.2703), 2009.

40. Nasu, M. and N. Honda, "A Context-Free Language Which is not Acceptable by a Probabilistic Automaton", *Information and Control*, Vol. 18, No. 3, pp. 233–236, 1971.

41. Freivalds, R., A. Yakaryılmaz, and A. C. C. Say, "A New Family of Nonstochastic Languages", *Information Processing Letters*, Vol. 110, No. 10, pp. 410–413, 2010.

42. Salomaaa, A. and M. Soittola, *Automata-Theoretic Aspects of Formal Power Series*, Texts and Monographs in Computer Science, Springer-Verlag, 1978.

43. Bertoni, A. and M. Carpentieri, "Analogies and Differences Between Quantum and Stochastic Automata", *Theoretical Computer Science*, Vol. 262, No. 1-2, pp. 69–81, 2001.

44. Nakanishi, M., T. Indoh, K. Hamaguchi, and T. Kashiwabara, "On the Power of Non-deterministic Quantum Finite Automata", *IEICE Transactions on Information and Systems*, Vol. E85-D, No. 2, pp. 327–332, 2002.

45. Macarie, I., "Closure Properties of Stochastic Languages", Technical Report 441, University of Rochester, 1993.

46. Yakaryılmaz, A. and A. C. C. Say, "Languages Recognized by Nondeterministic Quantum Finite Automata", *Quantum Information and Computation*, Vol. 10, No. 9&10, pp. 747–770, 2010.

47. Alt, H., V. Geffert, and K. Mehlhorn, "A Lower Bound for the Nondeterministic Space Complexity of Context-free Recognition", *Information Processing Letters*, Vol. 42, No. 1, pp. 25–27, 1992.

48. Ambainis, A. and R. Freivalds, "1-way Quantum Finite Automata: Strengths, Weaknesses and Generalizations", *FOCS'98: Proceedings of the 39th Annual Symposium on Foundations of Computer Science*, pp. 332–341, 1998.

49. Mereghetti, C., B. Palano, and G. Pighizzini, "Note on the Succinctness of Deterministic, Nondeterministic, Probabilistic and Quantum Finite Automata", *Theoretical Informatics and Applications*, Vol. 35, No. 5, pp. 477–490, 2001.

50. Diêu, P. D., "On a Class of Stochasic Languages", *Mathematical Logic Quarterly*, Vol. 17, No. 1, pp. 421–425, 1971.

51. Ravikumar, B., "Some Observations on 2-way Probabilistic Finite Automata", *Proceedings of the 12th Conference on Foundations of Software Technology and Theoretical Computer Science*, pp. 392–403, Springer-Verlag, 1992.

52. Lipton, R. J. and Y. Zalcstein, "Word Problems Solvable in Logspace", *Journal of the ACM*, Vol. 24, No. 3, pp. 522–526, 1977.

53. Lyndon, R. C. and P. E. Schupp, *Combinatorial Group Theory*, Springer-Verlag, 1977.

54. Turakainen, P., "On Nonstochastic Languages and Homomorphic Images of Stochastic Languages", *Information Sciences*, Vol. 24, No. 3, pp. 229–253, 1981.

55. Turakainen, P., "On Languages Representable in Rational Probabilistic Automata", *Annales Academiae Scientiarum Fennicae, Ser.A*, , No. 439, pp. 4–10, 1969.

56. Świerczkowski, S., "A Class of Free Rotation Groups", *Indagationes Mathematicae*, Vol. 5, No. 2, pp. 221–226, 1994.

57. D'Alessandro, F. and A. D'Andrea, "A Non-Commutativity Statement for Algebraic Quaternions", *International Journal of Algebra and Computation*, Vol. 16, No. 3, pp. 583–602, 2006.

58. Fliess, M., "Automates Stochastiques et Séries Rationnelles Non Commutatives", *Automata, Languages, and Programming*, pp. 397–411, 1972.

59. Fliess, M., "Propriétés Booléennes des Langages Stochastiques", *Mathematical Sys-*

tems Theory, Vol. 7, No. 4, pp. 353–359, 1974.

60. Lapiņš, J., "On Nonstochastic Languages Obtained as the Union and Intersection of Stochastic Languages", *Avtom. Vychisl. Tekh.*, , No. 4, pp. 6–13, 1974 (Russian).

61. Bukharaev, R. G., "Theory of Probabilistic Automata", *Kibernetika*, Vol. 4, No. 2, pp. 6–23, 1968.

62. Turakainen, P., "On Stochastic Languages", *Information and Control*, Vol. 12, No. 4, pp. 304–313, 1968.

63. Cole, S. N., "Real-Time Computation by n-Dimensional Iterative Arrays of Finite-State Machines", *IEEE Transactions on Computers*, Vol. 18, No. 4, pp. 349–365, 1969.

64. Turakainen, P., "Some Closure Properties of the Family of Stochastic Languages", *Information and Control*, Vol. 18, No. 3, pp. 253–256, 1971.

65. Yakaryılmaz, A. and A. C. C. Say, "Succinctness of Two-Way Probabilistic and Quantum Finite Automata", *Discrete Mathematics and Theoretical Computer Science*, Vol. 12, No. 2, pp. 19–40, 2010.

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

67. Turakainen, P., *Discrete Mathematics*, Vol. 7 of *Banach Center Publications*, chapter Rational Stochastic Automata in Formal Language Theory, pp. 31–44, PWN-Polish Scientific Publishers, Warsaw, 1982.

68. 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.

69. Paschen, K., "Quantum Finite Automata Using Ancilla Qubits", Technical report, University of Karlsruhe, http://digbib.ubka.uni-karlsruhe.de/volltexte/1452000

(accessed in January 2011), 2000.

70. Ambainis, A., A. Nayak, A. Ta-Shma, and U. Vazirani, "Dense Quantum Coding and Quantum Finite Automata", *Journal of the ACM*, Vol. 49, No. 4, pp. 496–511, 2002.

71. Freivalds, R., M. Ozols, and L. Mančinska, "Improved Constructions of Mixed State Quantum Automata", *Theoretical Computer Science*, Vol. 410, No. 20, pp. 1923–1931, 2009.

72. Yakaryılmaz, A. and A. C. C. Say, "Efficient Probability Amplification in Two-Way Quantum Finite Automata", *Theoretical Computer Science*, Vol. 410, No. 20, pp. 1932–1941, 2009.

73. Scegulnaja-Dubrovska, O. and R. Freivalds, "A Context-Free Language not Recognizable by Postselection Finite Quantum Automata", Freivalds, R. (editor), *Randomized and Quantum Computation (MFCS & CSL 2010 satellite workshop)*, pp. 35–48, 2010.

74. Diêu, P. D., "Criteria of Representability of Languages in Probabilistic Automata", *Cybernetics and Systems Analysis*, Vol. 13, No. 3, pp. 352–364, 1977, translated from Kibernetika, No. 3, pp. 39–50, May–June, 1977.

75. Freivalds, R., "Fast Probabilistic Algorithms", *Mathematical Foundations of Computer Science 1979*, Vol. 74 of *LNCS*, pp. 57–69, 1979.

76. Szepietowski, A., "Remarks on Languages Acceptable in $\log n$ Space", *Information Processing Letters*, Vol. 27, pp. 201–203, 1988.

77. Brandenburg, F., "On One-Way Auxiliary Pushdown Automata", *Theoretical Computer Science*, pp. 132–144, 1977.

78. Greibach, S. A., "Remarks on Blind and Partially Blind One-Way Multicounter Machines", *Theoretical Computer Science*, Vol. 7, pp. 311–324, 1978.