Quantum Walks and Quantum Computation

Katharine Elizabeth Barr

Department of Physics and Astronomy

University of Leeds

Submitted in accordance with the requirements for the degree of

Doctor of Philosophy

31st May 2013

Declaration

The candidate confirms that the work submitted is her own, except where work which has formed part of jointly-authored publications has been included. The contribution of the candidate and the other authors to this work has been explicitly indicated below. The candidate confirms that appropriate credit has been given within the thesis where reference has been made to the work of others.

Several chapters of this thesis include work which was undertaken in collaboration with summer project students, which I now detail:

1. The work in Chapters 4 and 5 was undertaken with Tim Proctor and Daniel Allen. Tim and Daniel made the initial choice of graphs to test and ran the simulations which gave rise to the results detailed in Chapter 4 concerning these graphs. They also noted that three of the graphs gave rise to infinite families of graphs, the properties of which are detailed in Chapter 5. My contribution was the systematic testing of graphs based on cycles in Chapter 4, and the robustness tests and work relating to the continuous time walk in Chapter 5. The results from these chapters have been submitted to QIC in the paper 'Periodicity and perfect state transfer in quantum walks on variants of cycles.'

2. The work in Chapter 6 was undertaken with Tim Proctor, Ben Hanson, Simon Martiel, Alex Bullivant and Vladan Pavlovic. Tim designed the general non-reversal coin operator and the other students investigated the transport properties of the induced walk under a variety of initial conditions. I first noted the independence of the first and second moments on the initial condition and undertook wider and more systematic investigations into the walk. Tim provided the proof that the moments of the permuted non-reversal coin are independent of the initial condition. The results from this chapter are detailed in the paper 'Non-reversal quantum walks,' which is currently in preparation.

3. In Chapter 2, Section 2.7.4, Toby Fleming used the `networkx` package to visualise graphs and remove the isomorphisms from a set of structures I had created. This work is mentioned in the paper 'Simulation methods for quantum walks on graphs applied to perfect state transfer and formal language recognition,' to appear in the proceedings of CoSMoS 2013.

List of papers

These are the papers which have resulted from my PhD:

1. Formal languages analysed by quantum walks. K. Barr and V. Kendon, accepted for oral presentation and proceedings of QPL 2013. ArXiv preprint arXiv:1209.5238v1.

2. Simulation methods for quantum walks on graphs applied to perfect state transfer and formal language recognition. K. Barr, T. Fleming, V. Kendon, accepted for oral presentation and proceedings of CoSMoS 2013.

3. Non-reversal quantum walks. K. Barr, T. Proctor, B. Hansen, S. Martiel, V. Pavlovic, A. Bullivant and V. Kendon, in preparation for resubmission to Physical Review A. ArXiv preprint arXiv:1303.1966.

4. Periodicity and perfect state transfer in quantum walks on variants of cycles. K. Barr, T. Proctor, D. Allen and V. Kendon, submitted to QIC, currently preparing a revised version for further consideration. ArXiv preprint arXiv:1204.5937.

Acknowledgements

Firstly I would like to thank my PhD supervisor Viv Kendon. Without such a dedicated, understanding and motivated supervisor, I am sure that I would not have completed my doctoral work. Without her patience and constant belief in me, I would not have been able to maintain the self belief required to undertake such difficult work. Viv's support has gone beyond academic advice, and well above and beyond the call of duty. Getting the opportunity to work with a truly inspirational person has been one of the best aspects of my PhD and made it an experience I will look fondly on forever, in spite of not always having enjoyed the work!

Secondly I would like to thank my parents and closest friends for their ongoing support. I know this hasn't always been easy for them, and they have sometimes found it difficult to understand why I am following the path I have chosen. Shand and Aidan in particular deserve a special mention, for giving me something to look forward to at the end of every working day. I could not ask for better people in my life, and do not feel capable of fully expressing my level of gratitude.

Finally, I feel privileged to have met many fun and interesting people during the course of my PhD. I have enjoyed getting to know everyone in the QI group in Leeds, and my current and former office mates have been a reliable source of amusement and distraction in addition to providing a relaxed and productive working environment. I look forward to coming into work every day, and know that I am immensely lucky in that respect.

Abstract

The field of quantum information applies the concepts of quantum physics to problems in computer science, and shows great potential for allowing efficient computation. In this thesis I concentrate on a particular quantum information theoretic tool known as the quantum walk. There are two widely studied versions of the quantum walk, the continuous time walk, and the discrete time walk. The discrete time walk is particularly amenable to investigation using numerical methods, from which most of the results in this thesis are derived, and is the main focus of the work presented.

Two aspects of the discrete time walk are investigated: their transport properties and their interpretation as quantum computers. I investigated the transport properties in two ways, by looking for a particular type of transport known as perfect state transfer, and examining the transport properties of a new type of coin operator. The search for perfect state transfer concentrated on modifications of small cycles. I found that perfect state transfer is rare for the choices of coin operators tested. The structures tested for perfect state transfer were based on cycles, and it appears that the type of modification has more of an effect than the size of the cycle. This makes intuitive sense, as the modifications found to lead to walks exhibiting perfect state transfer affected only the initial and target node of the cycle.

I then investigated a new type of coin operator which does not allow amplitude to return to the node it has come from. This effectively simulates a dimer. Using the general form of this type of operator and random variables for each parameter, I found that the expected distance of the walker from the origin, and standard deviation, were independent of the initial condition.

The second half of the thesis concentrates on computational applications of quantum walks using the language acceptance model. I first note their equivalence to a

type of quantum automaton known as the QFA-WOM, and this provides an intuitive understanding of the role of the WOM. I then use a more direct construction to show that they can accept a range of formal languages. Using this construction allows us to use superpositions of words as inputs, and the insights provided by investigating these suggest a new way of approaching the problem of quantum state discrimination.

Contents

List of Figures

Abbreviations

C	Coin operator
S	Shift operator
$\mathcal{G}$	Graph
$A_{\mathcal{G}}$	Adjacency matrix of a graph
K_n	The complete graph of n vertices
P_n	The path of n vertices
C_n	The cycle of n vertices
V	Set of vertices
E	Set of edges
G_d	d-dimensional Grover operator
DFT	Discrete Fourier Transform
H	Hadamard operator
FSA	Finite State Automaton
QFA	Quantum Finite Automaton
$2QFA$	Two-way QFA
$MO-QFA$	'Measure-once' QFA
RMO	Restricted class of languages accepted by an $MO-QFA$
UMO	Unrestricted class of languages accepted by an $MO-QFA$
$MM-QFA$	'Measure- many' QFA
$QFA-WOM$	QFA with 'write-only' memory
Σ	Input alphabet
Q	Set of computational states
δ	Transition function for an automaton
$\mathcal{L}_{eq}$	Language defined by $\{a^m b^m \vert m \in \mathbb{N}\}$
$\mathcal{L}_{ab}$	Language defined by $\{(ab)^m \vert m \in \mathbb{N}\}$
$\mathcal{L}_{even}$	Language denoted by $\{a,b\}^* a$
$\mathcal{L}_{abc}$	Language defined by $\{a^m b^m c^m \vert m \in \mathbb{N}\}$

Chapter 1

Introduction

The 20th century saw many great advances in science, from the extraordinary, such as using the curvature of spacetime to enable extremely precise global positioning systems, to the more mundane, such as using microwaves to heat our food. Two theories which helped shape the world as we know it today are the theory of computation and quantum theory. The field of quantum information synthesises the two by applying the principles of quantum mechanics to computational problems. This promises to enable us to perform computations far faster than our current technology allows. As our understanding of science increases, we need more detailed models, and more powerful computers to investigate them. Thus quantum computation provides tantilising theoretical problems and the potential to facilitate progress in many areas of science.

1.1 Thesis Aim

In this thesis I focus on one particular tool used by quantum information theorists, namely the discrete time quantum walk. I have undertaken largely numerical work looking into two particular applications: transport and computation. I investigate various aspects of transport in the discrete time walk. Of particular interest is when and how a state can be transmitted from one place to another with certainty. This question has been extensively studied in the case of the continuous time quantum walk, which I also examine briefly, but has received little attention in the case of the discrete time walk. I look into the role that the coin plays on the transport properties of the walk, including using a novel type of coin which prevents the walker from stepping back into the position it has just come from.

I then investigate discrete time quantum walks in the context of language acceptance models of computation. Firstly we see that they are equivalent to a new type of quantum finite automaton known as the QFA-WOM. Then I show how the walk can be used to solve language acceptance problems more directly. The concept of a quantum input arises quite naturally from this application of the walk, and we finish by investigating these using the walks developed.

1.2 Thesis Overview

I start by giving a detailed overview of the relevant aspects of quantum walks in Chapter 2 and quantum computation in Chapter 3. This motivates study into quantum walks and quantum computers, and indeed quantum walks as quantum computers. I give examples of the dynamics of both the discrete and continuous time walks on the line in the purely quantum regime, and undergoing decoherence in order to show that the classical random walk dynamics are obtained from the quantum walk dynamics. Chapter 3 includes introductions to classical computation in terms of two paradigms: circuits and finite automata. Quantum computation is then described, initially in terms of the circuit model, and the discrete time quantum walk is shown to simulate a universal gate set. Quantum finite automata are also defined, and compared to their classical counterparts.

The following three Chapters, 4-6, concern transport properties of the discrete time quantum walk. In Chapter 4 we start by surveying past results concerning perfect state transfer in quantum walks, before discussing the results of a brute force search for perfect state transfer using three types of coin operator. This search looked at many variations of cycles of different sizes, and generalisations of some of the graphs found to exhibit perfect state transfer are examined in detail in Chapter 5. These generalisations form three infinite families of graphs, and the dynamics in the discrete time walk are compared to that of the continuous time walk.

In Chapter 6 a new type of coin operator, which, for the right choice of shift operator, prevents amplitude leaving a node of a graph via the edge it arrived from, is introduced. This would cause trivial dynamics in the one dimensional case, as it would cause deterministic propagation away from the origin. I therefore investigate the dynamics of this type of walk on a two dimensional lattice, and compare these to the dynamics which arise when other coin operators are used,

I then turn to quantum computation in Chapters 7-9. In Chapter 7 we see how special types of environmental interaction enable quantum finite automata to accept more languages. One of

the models which takes advantage of environmental interaction, the QFA-WOM, is equivalent to a discrete time quantum walk. This enables an intuitive understanding of the behaviour of the QFA-WOM, which is seen to control interference between computational states by controlling when they are situated at the same node of a graph.

In Chapter 8 I show that discrete time quantum walks can be used to solve language acceptance problems by directing amplitude to an accepting node, without using explicit computational states. Two ways of initialising the input are introduced and it is observed that it is easy to initialise the walk with an input which is a superposition of two words. This novel, quantum, form of input is the subject of the preliminary investigations detailed in Chapter 9. The key result from these investigations is that the language accepting walks can be seen to perform a type of quantum state discrimination.

The final chapter concludes with a summary of the results detailed in the main body of the thesis. Open questions and ideas for future work are also discussed.

Chapter 2

Quantum Walks

2.1 Introduction

As classical random walks have found many applications, particularly in mathematics, physics and computer science, it is natural to consider their quantum counterparts. Classical random walks have recently been used to develop new algorithms which outperform their predecessors. For example, the best known algorithms to solve the constraint satisfiability problem, whereby one must find objects with a certain set of properties, use the classical random walk [1; 2].

Prior to the development of quantum walks, quantum algorithms which solved problems more efficiently than any known classical algorithm had been developed. Examples of these are mentioned in the introduction to Chapter 3 below. The development of quantum walks can be traced back to the study of quantum diffusion, which mostly focuses on the evolution of particles on regular lattices. There is a wealth of literature on this topic, for example Feynman *et al* [3]. There were many early models of quantum random walks [4; 5; 6], developed with a variety of applications in mind. Quantum walks were first shown to be interesting from an algorithmic point of view by Ambainis *et al* [7] in the discrete time case, and Farhi and Guttmann [8] in the continuous time case. Both walks were shown to provide speedups in comparison to the classical cases, on the line for the discrete time walk there is a quadratic speedup and on a binary tree structure in the continuous time walk the speedup is exponential. These results prompted a proliferation of further study into both types of quantum walk.

Quantum walks have many specific algorithmic applications, for example the element disctinctness algorithm [9], constraint satisfiability [1; 10] and the quantum walk search algorithm. The search algorithm has both continuous [11] and discrete [12] time versions which have been

extensively studied over a variety of structures [13; 14; 15; 16; 17; 18; 19; 20]. In addition to these applications, quantum walks have been shown to be universal for quantum computation. This was first shown by Childs for the continuous time walk [21; 22], then by Lovett *et al* for the discrete time walk [23]. Both of the proofs work by specifying graph structures, and in the discrete time case appropriate coins, which simulate an elementary gate set required for universal quantum computation. Overviews of the applications of quantum walks to computer science can be found in [9; 24; 25].

The fact that they can achieve exponentially faster hitting times than their classical counterparts [8; 26] gives rise to many of the algorithmic applications [8; 27; 28] of quantum walks. This also leads to the consideration of quantum walks in the context of quantum transport. A particular transport problem, that of finding conditions in which perfect state transfer can occur, has received much attention. The seminal work on perfect state transfer in unmodulated spin chains was undertaken by Bose [29]. Since then, there have been many results, both analytic and numerical, relating to perfect state transfer, and these have been recently reviewed by Kendon and Tamon in [30]. Further reviews and discussions of applications of perfect state transfer can be found in [31; 32]

Decoherence in quantum walks has been found to be useful under certain circumstances. From an algorithmic point of view, decoherence in the position basis at the correct rate gives rise to a perfectly random probability distribution for the particles position [33]. From the transport perspective, an analysis of transport of an exciton through a light-harvesting antenna complex showed that the successful trajectories were continuous time quantum walks [34]. Further investigations into environment assisted quantum transport followed, see, for example, [35; 36]. Decoherence is not a main focus of this thesis, but will be briefly discussed in relation to transport over particular structures.

The rest of this introductory chapter provides the background material necessary for the discussions of quantum walks in the following chapters. First, some basic graph theoretic concepts are introduced. Then the discrete and continuous time classical random walks are defined. I then introduce the discrete time quantum walk on the line (Section 2.4.1), briefly summarising its behaviour, before defining these walks over general structures. Continuous time quantum walks over arbitrary structures are then briefly defined in Section 2.4.3. Perfect state transfer conditions for both cases are defined in Section 2.5 and the effects of decoherence are discussed in Section 2.6. The chapter concludes with Section 2.7 where I describe how to simulate and visualise discrete time walks over arbitrary structures. The code generated using

the algorithm described was used for every discrete time walk described in subsequent chapters, apart from the self-avoiding walks in Chapter 6.

2.2 Basic Graph Theory

Both discrete and continuous time quantum walks evolve over an underlying graph structure. In this section I introduce the graph theoretic apparatus which will be used throughout this thesis.

A graph $\mathcal{G} = \{E, V\}$ is a set of vertices, or nodes, V and a set of edges E of the form (v, w) where $v, w \in V$. If the pairs (v, w) are unordered, then the graph is said to be *undirected*. All the graphs considered in this thesis are undirected, though there are quantum walk models which use directed graphs [37; 38]. The *degree* of a vertex, d_v, is the number of vertices it is joined to. The *order* of a graph is the number of vertices, $|V|$ and the *size* is the cardinality of the set of edges. It is convenient to define the graph in terms of its *adjacency matrix* A with entries specified by :

$$A_{v,w} = \begin{cases} 1 & (v, w) \in E \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

The adjacency matrix $A_{\mathcal{G}}$ of $\mathcal{G}$ has ones in the entries (v, w) if vertex v is connected to vertex w and has zeroes elsewhere. A graph is *complete* if every vertex is joined by an edge to every other vertex. Graphs with the same number, k, of edges incident on each vertex is *k-regular*, with the k often omitted if its value is contextually clear. A particular type of regular graph known as a *hypercube* has $N = 2^n$ vertices, each of which have n edges incident on them. Some other common graphs, which will be used in this thesis are K_n, the complete graph with

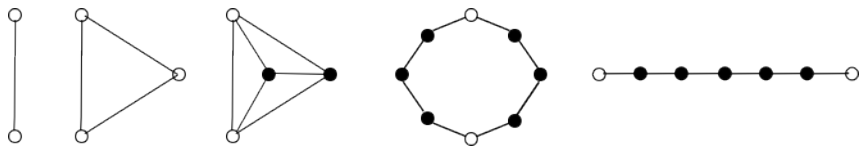


Figure 2.1: Examples of graphs, from left to right: the complete graph with 2 vertices, $K_2 = P_2$, the path with 2 vertices; the complete graph with 3 vertices $K_3 = C_3$ the cycle with 3 vertices; the complete graph K_4; the cycle C_9 and the path P_7. The open circles indicate potential end points for examining transport properties.

n vertices; P_n the path of n vertices and C_n, the cycle of n vertices, which differs from the path by only one edge. Examples of these graphs are shown in Figure 2.1. As an example, the adjacency matrix of K_4 is

$$\begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}. \tag{2.2}$$

One can define the *complement* of a graph $\mathcal{G}$, denoted by $\overline{\mathcal{G}}$, which shares the same vertices as $\mathcal{G}$ but its set of edges is given by $\overline{E} = \{(v, w) \mid (v, w) \notin E\}$. The *join* of graphs $\mathcal{G}$ and $\mathcal{H}$ is denoted $\mathcal{G} + \mathcal{H}$ and its adjacency matrix is given by

$$A_{\mathcal{G}+\mathcal{H}} = \begin{pmatrix} A_{\mathcal{G}} & J \\ J & A_{\mathcal{H}} \end{pmatrix}. \tag{2.3}$$

J is the all ones matrix of the relevant dimensions, so the join operation links all nodes of $\mathcal{G}$ to all nodes of $\mathcal{H}$. The notion of the adjacency matrix can be generalised to allow it to have entries not equal to either 0 or 1, which varies the strength with which vertices are joined. A graph with only integer values in its adjacency matrix is called an *integral graph*. Classically, integral graphs with adjacency matrices whose entries are greater than one represent multiple edges between nodes. If the diagonals of $A_{\mathcal{G}}$ are integers, then the graph $\mathcal{G}$ is interpreted as having self loops.

2.3 Classical random walks

2.3.1 Discrete time case

The classical discrete time random walk is simply a succession of steps over some structure, the directions of which are decided at random. The classic example is the walk on the line, where the decision to take a step left or right is made using an unbiased coin. The walker can be considered moving along a number line which stretches from $-N$ to N much like the small portion depicted in Figure 2.3. The average position of the walker at any time t is the origin, but if run forever a walk over $\mathbb{Z}$ will pass each point an infinite number of times. If a sufficient number of walks with t timesteps are performed and their final positions plotted, a binomial distribution with standard deviation t is obtained, as can be seen in Figure 2.2.

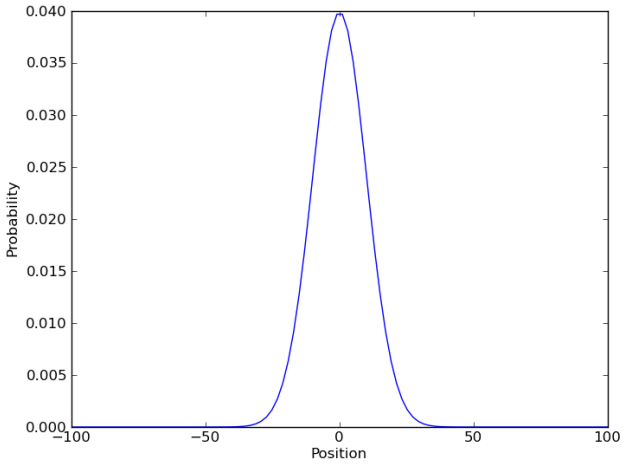


Figure 2.2: The probability distribution of final positions of a classical random walker after multiple runs each of 100 steps

2.3.2 Continuous time case

The classical continuous time random walk was introduced by Montroll and Weiss [39]. In contrast to the discrete time case, in the continuous time case there is a certain probability of the walker hopping within a certain time interval. This gives rise to a hopping rate γ. The probability distribution across the graph over which the walk takes place is determined by a matrix M. This matrix specifies how the probabilities are updated in every time step. The entries $M_{v,w}$ of M indicate the probability of jumping from the vertex v to vertex w and are defined to be

$$M_{v,w} = \begin{cases} -\gamma & \text{if } v \neq w \text{ and } v \text{ is connected to } w \\ 0 & \text{if } v \neq w \text{ and there is no connection between } v \text{ and } w \\ d_v\gamma & \text{if } v = w \end{cases} \tag{2.4}$$

The probability distribution over all vertices at time t, $\vec{p}(t)$ can then be obtained by

$$\vec{p}(t) = e^{-Mt}\vec{p}(0) \tag{2.5}$$

where $\vec{p}(0)$ is the initial probability distribution. The evolution of the probability of finding the walker at a specific vertex v on the graph with adjacency matrix A is expressed in terms of the

differential equation

$$\frac{d}{dt}p(v,t) = \gamma \sum_{w} \{A_{v,w}p(w,t) - A_{w,v}p(v,t)\}. \tag{2.6}$$

This just sums the incoming and outgoing probability at each vertex w where $(v,w) \in E$.

2.4 Quantum walks

2.4.1 Discrete time quantum walks on the line

Before introducing the discrete time walk on a general graph, I describe the walk on the line in some detail by way of example. The first way one may consider 'quantising' the classical random walk would be to allow the walker to move in a superposition of all possible walks. This has been shown in [40] not to be reversible. The specification that all operations in quantum mechanics must be unitary means that all purely quantum dynamics must be reversible, hence this suggestion is not viable. Another approach is to to make a quantum coin flip operation, by allowing the result of the coin flip to be a superposition of heads and tails. This means that, analogously to the classical walk, each step is divided into two subroutines, one for the coin toss and the other for shifting the probability amplitude along the line. Thus the Hilbert space of the walk is the tensor product of the position space $H_x = span\{|x\rangle | x \in \mathbb{Z}\}$, and the coin space, which has basis vectors specifying the 'coin state.' Coin states will be called -1 to indicate that the walker is to move left, and $+1$ when the walker is to move right. The walker can start the walk at position $x = 0$ in a superposition of coin states, as indicated by

$$|\psi(t=0)\rangle = a|0,-1\rangle + be^{i\phi}|0,+1\rangle \tag{2.7}$$

where a and b are real numbers such that $a^2 + b^2 = 1$ and $0 \leq \phi < 2\pi$ is an arbitrary phase. As the coin state specifies the direction in which the shift operation should move the walker, having a superposition of coin states will result in the walker moving in a superposition of left and right, leading to the state

$$|\psi(t=1)\rangle = a|-1,-1\rangle + be^{i\phi}|1,+1\rangle. \tag{2.8}$$

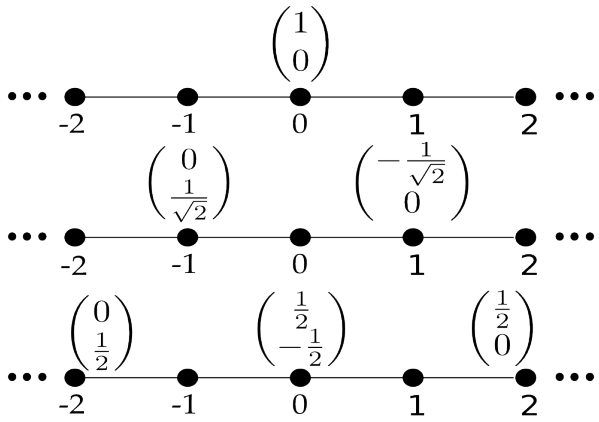


Figure 2.3: The first two steps of the discrete time quantum walk on the line with an initially localised walker

The coin and shift operators, denoted **C** and **S** respectively give the time evolution of the state $|\psi(t)\rangle$, as $|\psi(t+1)\rangle = \mathbf{SC}|\psi(t)\rangle$. **C** can be any unitary operator on the Hilbert Space of the coin states, the most general of which, in the case of the walk on the line, can be written

$$\mathbf{C}^{gen} = \begin{pmatrix} \sqrt{\rho} & e^{i\theta}\sqrt{1-\rho} \\ e^{i\theta}\sqrt{1-\rho} & -e^{i(\theta+\phi)}\sqrt{\rho} \end{pmatrix} \tag{2.9}$$

with arbitrary angles θ and ϕ and probability that the walker moves right such that $0 \leq \rho \leq 1$. Taking $\theta = \phi = 0$ and $\rho = \frac{1}{2}$ gives the Hadamard coin operator, denoted H. If the columns of H are swapped around we have H_2

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix} \qquad H_2 = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ -1 & 1\end{pmatrix}. \tag{2.10}$$

It has been shown in [7] that the full range of dynamics for the walk on the line can be obtained with just the biased Hadamard operator, taking $\theta = \phi = 0$ and varying ρ in Equation 2.9, by varying the initial state of the walker. The evolution of the first two steps of the walk on the line using the unbiased Hadamard operator can be seen in Figure 2.3.

The classical random walk propagates at a rate proportional to $\sqrt{T}$, where T is the number of timesteps, whereas the rate for the quantum walk is proportional to T. This speedup is

intuitively because some amplitude is always moved left, and some is always moved right for each step of the walk, so every step results in the distribution being two positions wider than it was at the previous step. Whilst a classical random walk may propagate away from the origin at each timestep, this is unlikely. Due to the interference which takes place in the quantum walk, it is possible to select initial conditions such that the walker will be measured at a position which is some distance proportional to T from the origin with high probability.

2.4.2 Discrete time quantum walks over general structures

Whilst the transport properties of the discrete time quantum walk on the line are themselves interesting, it can also be useful to consider quantum walks in higher dimensions. For example the quantum walk implementation of Grover's algorithm [12] takes place over an n-dimensional hypercube where each node represents an n-bit binary string. Varying the structure over which Grover's algorithm is implemented has been found to affect its efficiency [41]. Walks on irregular, higher dimensional graphs are used in the proof that discrete time quantum walks are universal for quantum computation, see Chapter 3 Section 3.3, and to solve language acceptance problems in Chapter 8.

A discrete time quantum walk over an arbitrary graph structure $\mathcal{G} = \{E, V\}$ evolves, just as for the walk on the line, according to a coin operator and a shift operator. In order to accommodate walks over more complex structures both of these operators must be modified. The coin operator acting at vertex $v \in V$ has dimension $d \times d$, where d is the degree of the vertex, is any unitary. Coin operations which will be particularly relevant to this thesis are the Grover operator:

$$G_d = \begin{pmatrix} \frac{2-d}{d} & \frac{2}{d} & \cdots & \frac{2}{d} \\ \frac{2}{d} & \frac{2-d}{d} & \cdots & \frac{2}{d} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{2}{d} & \frac{2}{d} & \cdots & \frac{2-d}{d} \end{pmatrix} \tag{2.11}$$

and the unitary Discrete Fourier Transform (DFT), specified by:

$$DFT = \frac{1}{\sqrt{d}} \begin{pmatrix} \omega_d^{0\times 0} & \omega_d^{0\times 1} & \dots & \omega_d^{0\times(d-1)} \\ \omega_d^{1\times 0} & \omega_d^{1\times 1} & \dots & \omega_d^{1\times(d-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \omega_d^{(d-1)\times 0} & \omega_d^{(d-1)\times 1} & \dots & \omega_d^{(d-1)\times(d-1)} \end{pmatrix} \tag{2.12}$$

with $\omega_d = e^{\frac{-2\pi i}{d}}$. The $d = 2$ version of the DFT is more commonly known as the Hadamard, H.

After the coin toss, a shift operation is applied. This is simply a permutation between the relevant coin states at different vertices, and is hence a unitary transformation. The shift operator acts like $\mathbf{S}|v, c\rangle = |w, n\rangle$, so moves amplitude from the c^{th} coin state of v to the n^{th} coin state of w [42]. The probability of the walker being measured at position v after T steps is the summation over coin states at v, $p(v, t) = \sum_i |v, c_i|^2$. The state of the quantum walk must describe both the position of the walker, and the configuration of coin states. A general state is written

$$\psi(T) = \sum_{v,c} \alpha_{v,c}(T)|v, c\rangle \tag{2.13}$$

where $\alpha_{v,c} \in \mathbb{C}$ and $|v, c\rangle$ denotes a basis state on vertex v with coin state c.

Various attributes of discrete time quantum walks are of potential interest, such as the limiting distribution, [43; 44], which is the limit as $T \to$ inf of the time averaged probability distribution

$$\overline{P}_T(v|\Psi(t = 0)) = \frac{1}{T} \sum_{t=0}^{T-1} P_T(v|\Psi(t = 0)) \tag{2.14}$$

where

$$P_T(v|\Psi(t = 0)) = \sum_c |\langle c, v|\Psi(t)\rangle|^2. \tag{2.15}$$

The hitting time, which is characterised as the probability that a walker starting in state $|\Psi\rangle$ will be at position $|v\rangle$ after T steps with probability greater than p

$$|\langle x|\mathbf{S}\mathbf{C}^T|\Psi\rangle|^2 \geq p, \tag{2.16}$$

has also been extensively studied [26; 45; 46]. In contrast to the walk on the line, different choices of coin operation can produce dramatically different dynamics. This is discussed further in Chapter 6 in the context of a new type of coin.

The quantum walks discussed in this thesis all take place on undirected graphs. In general, it is difficult to ensure that the dynamics of the walk will be unitary if they take place on a directed graph as the adjacency matrix is not symmetrical so the shift operator is not guaranteed to be Hermitian. There have been a few approaches to the problem of defining quantum walks on directed graphs, such as [37; 38] and some necessary and sufficient conditions which the graph must fulfil to allow for unitary dynamics are shown in [47; 48]. The method used in [38] uses a mapping which turns each edge of an undirected graph into a pair of directed edges, however for the Grover coin this method was then shown to replicate the dynamics of the walk over the original undirected graph [49]. There has been little follow up work concerning walks on directed graphs, though they have been shown to give rise to faster transport than the classical walk on the line in [50].

2.4.3 Continuous time quantum walks

Continuous time quantum walks were first introduced by Farhi and Gutmann in [8]. It is the quantum analogue of the classical continuous time random walk. A *continuous time quantum walk* evolves over graph $\mathcal{G} = \{E, V\}$ according to the Schrödinger equation. This means that, in contrast with the discrete time case, no dynamical control is required to implement the continuous time walk. In this thesis the adjacency matrix $A_\mathcal{G}$ gives rise to the Hamiltonian, which is taken to be $\mathcal{H} = \gamma A_\mathcal{G}$ where γ is the hopping rate, though there are models where the Laplacian, whose elements (v, w) are defined by

$$\mathbf{L}_{(v,w)} = \begin{cases} -1 & \text{if } v \neq w \text{ and } v \text{ is connected to } w \\ 0 & \text{if } v \neq w \text{ and there is no connection between } v \text{ and } w \\ d_v & \text{if } v = w \end{cases} \tag{2.17}$$

is used instead. In the case of regular graphs, the evolution generated by using the Laplacian rather than the adjacency matrix for the Hamiltonian is identical, as the diagonal elements of the Laplacian are all equal and contribute a global phase. I used the adjacency matrix version of the walk, as the vast majority of results concerning the continuous time walk use this formulation [30; 51; 52], facilitating comparison of my results with prior work.

As there is no coin space, the walk evolves entirely in the position basis. The walker evolves in an N dimensional Hilbert Space with basis states $|v\rangle$ where $v \in V$. In units where $\gamma = \hbar = 1$, the state of a continuous time walk at time t is given by

$$|\phi(t)\rangle = e^{-i\gamma A_{\mathcal{G}} t}|\phi(0)\rangle. \tag{2.18}$$

The evolution of the amplitude at a specific node v is given by

$$i\frac{d}{dt}\langle v|\phi(t)\rangle = \sum_{w}\langle v|\mathcal{H}|w\rangle\langle w|\phi(t)\rangle. \tag{2.19}$$

The amplitude at vertex v at time a particular time t, $\alpha_v(t)$, is $\langle v|\phi(t)\rangle$.

Unlike in the discrete time case, where multiple coin states can accommodate multiple edges between the same two vertices, there can be at most one edge between vertices of $\mathcal{G}$ in the continuous time case. Varying the value of the entry $A_{\mathcal{G}}(i, j)$ gives rise to a weighted edge between vertex i and vertex j, which is equivalent to repeating an edge when the value is made greater than one. The probability distribution for the walk on the line is quite similar to that of the discrete time walk, and can be seen in Figure 2.4 d) where the purely quantum version corresponds to $p = 0$.

In the seminal paper, continuous time walks were shown to achieve exponentially faster transport than their classical counterparts over a glued binary tree structure [8]. On the line, they show the same speedup as the discrete time walk when an appropriate value for γ is selected, as shown in Figure 2.4 d). It is not always the case that the continuous time quantum walk propagates faster than the classical continuous time walk. This is shown in [53] for a case which, for some initial conditions produces faster propagation but for others produces slower propagation. The walks which are shown analytically to have infinite hitting times in [46] also give rise to a class of structures over which continuous time quantum walks are slower than classical walks as hitting times for classical walks are always finite.

2.5 Perfect State Transfer

For the continuous time walk, there is *perfect state transfer* between vertices v and w at time t if the following condition is met:

$$\langle w|e^{-iA_{\mathcal{G}}t}|v\rangle = 1 \tag{2.20}$$

where $|v\rangle$ and $|w\rangle$ are unit vectors at vertices v and w respectively and $t > 0$. If the perfect state transfer condition holds for some $v, w \in V$ then we say the graph $\mathcal{G}$ has perfect state transfer. A special case of perfect state transfer occurs when $v = w$, in which case the graph $\mathcal{G}$ is *periodic*.

In the discrete time quantum walk, *perfect state transfer* occurs between vertices v and w after T steps if

$$\sum_{c,d}\langle w, d|(\mathbf{SC})^T|v, c\rangle = 1. \tag{2.21}$$

It is clear from this definition that we are not concerned about whether the coin states are in the same configuration at vertex w as they were at vertex v, which enables perfect state transfer between vertices of different degrees. For the definition of periodicity, the stricter requirement of returning to the same configuration of coin states is made:

$$\sum_{c}\langle v, c|(\mathbf{SC})^T|v, c\rangle = 1. \tag{2.22}$$

The perfect state transfer conditions may be stricter than necessary for some applications, in which case one can look for *high amplitude transfer* instead. A lower bound λ for the transmitted amplitude must be selected, and then we require that

$$\sum_{c,d}\langle w, d|(\mathbf{SC})^T|v, c\rangle \geq \lambda, \qquad \langle w|e^{-i\gamma A_{\mathcal{G}}t}|v\rangle \geq \lambda \tag{2.23}$$

in the discrete and continuous time cases respectively.

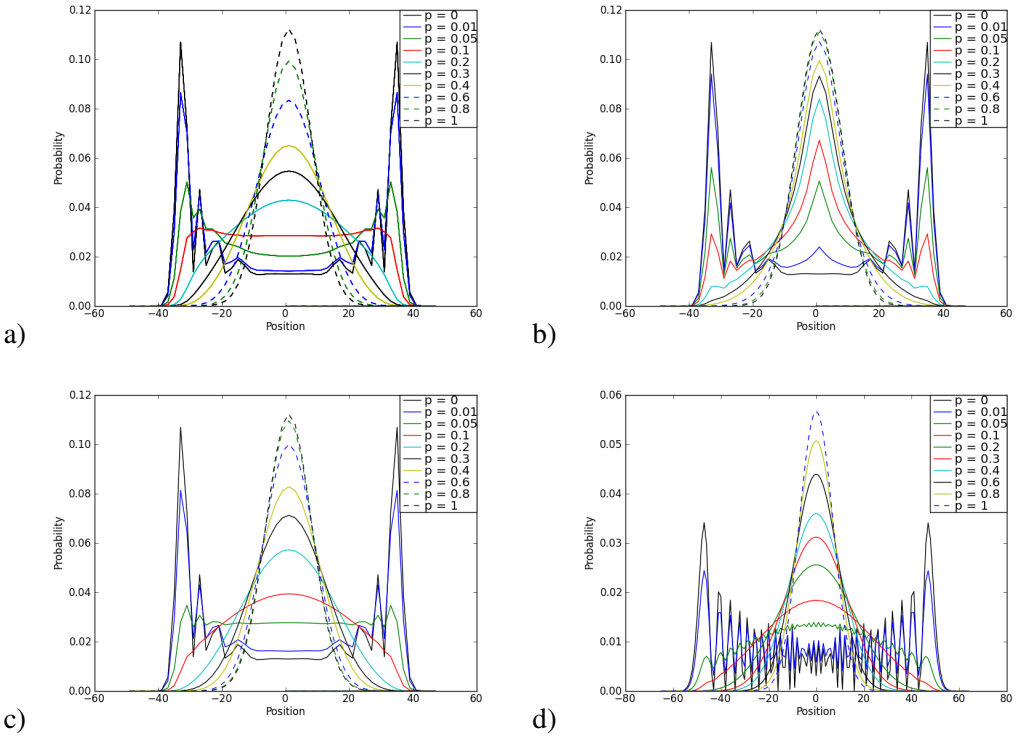


Figure 2.4: The effects of decoherence on the discrete time walk over a range of rates a) the position basis b) the coin basis and c) both the position and coin basis. Each walk was run for fifty steps and started in an equal superposition of coin states. d) Shows the effect of decoherence on the continuous time walk run for fifty timesteps with $\gamma = 0.5$

2.6 Decoherence in Quantum Walks

Quantum systems undergo decoherence when they interact with an environment, in which case their evolution is no longer unitary. One way that an environment can interact with a system is for it to perform measurements on the system using projection operators. This formalism is chosen because, when the decoherence rate is set equal to one, the classical random walk is recovered, justifying the description of the quantum walk as a quantum analogue of the classical random walk. A detailed account of how to model uncorrelated environmental interactions in various types of quantum walks can be found in Kendon [49]. For the discrete time quantum walks, there are two potential types of decoherence: in the coin basis and in the position basis. In both cases the density operator notation must be used, so the state of the quantum walker is written

$$\rho = \sum_{v,c} \sum_{v',c'} \rho_{v,c,v',c'} |v, c\rangle\langle v', c'| \tag{2.24}$$

The evolution operator $\mathbf{U} = SC$ acts on the density matrix in the required fashion, so $\mathbf{U}\rho = SC\rho = \mathbf{SC}\rho\mathbf{C}^\dagger\mathbf{S}^\dagger$. Decoherence in either basis can then be modelled by modifying $\mathbf{U}$:

$$\mathbf{U} :\rightarrow \sum_{j\in\Theta} \mathbb{P}_j \mathbf{SC}\rho\mathbf{C}^\dagger\mathbf{S}^\dagger\mathbb{P}_j^\dagger \tag{2.25}$$

where the projection operators $\mathbb{P}_j$ can project either in the coin basis, the position basis, or both. The set of projection operators Θ has cardinality one if the evolution is unitary, as $\mathbb{P}_j$ must be the identity in this case. The evolution of the quantum walk, with probability of a decoherence event occurring during a given timestep p is then given by

$$\rho(t+1) = (1-p)\mathbf{SC}\rho(t)\mathbf{C}^\dagger\mathbf{S}^\dagger + p\sum_j \mathbb{P}_j \mathbf{SC}\rho(t)\mathbf{C}^\dagger\mathbf{S}^\dagger\mathbb{P}_j^\dagger. \tag{2.26}$$

The effects of decoherence depend on which basis the projection operators project onto, as shown in Figure 2.4. In the case of the position basis, the decoherence rate can be selected so that after a given number of steps, a perfectly random distribution is obtained, as seen in Figure

2.4 a). Decoherence in the coin basis creates a central cusp (Figure 2.4 b)), which, as the rate increases becomes the peak of the expected classical distribution.

In the continuous time case, the purely quantum time evolution in density matrix notation is given by

$$\frac{d\rho(t)}{dt} = -i[A_{\mathcal{G}}, \rho(t)] \tag{2.27}$$

and when there is Markovian noise present the density matrix evolves according to

$$\frac{d\rho(t)}{dt} = -i[A_{\mathcal{G}}, \rho(t)] - p\rho(t) + p\sum_j \mathbb{P}_j \rho(t) \mathbb{P}_j^{\dagger}. \tag{2.28}$$

This is a specific version of the Lindblad equation, where we have a complete set of projection operators $\sum_j \mathbb{P}_j^{dagger} \mathbb{P}_j = 1$ and $\mathbb{P}_j^{dagger} = \mathbb{P}_j$. The effect of decoherence in the continuous time walk is shown in Figure 2.4 d).

2.7 Simulating quantum walks

Due to the degrees of freedom introduced by the coin space, the discrete time quantum walk is very difficult to investigate analytically. Some cases on regular lattices have been solved exactly using path counting and Fourier space techniques. The case of the walk on the line was solved by Ambainis *et al* [7], the hypercube was treated by Moore and Russell [54] and then Kempe [26], and higher dimensional lattices were treated by Gottlieb *et al* in [55]. General solutions for walks over arbitrary structures have not yet been attempted. In cases where the initial states and entries in the coin operator are represented by numbers which can be handled exactly by the simulation, such as those in the Grover operator with the walker initially localised in a specific coin state of a given node, they can be exactly simulated. In general, exact simulations are rare, only being possible for a small number of timesteps or in cases where the evolution is periodic. Our own simulations coupled with subsequent analytic work for a few exactly solvable, highly symmetric, cases indicate that the walks can be simulated to a very high degree of precision for at least 100 timesteps. Much longer walks can be simulated [56] but eventually

numerical accuracy may become an issue. Their difficulty in being treated analytically coupled with their suitability for numerical investigation is why, thus far, the vast majority of results concerning the discrete time quantum walk have been obtained numerically [23; 33; 43; 57]. This contrasts with the case of the continuous time walk where numerical methods are less suitable, as numerical integration is required to simulate the time evolution. These walks are, however more amenable to analytic techniques [29; 30; 52; 58]. As they obey the Schrödinger equation their evolution can be written in terms of the eigensystems of their Hamiltonian, which is either the adjacency matrix or Laplacian of their graph structure. In cases where the eigenvalues and vectors have simple expressions, it is easy to write out the full time evolution of the walk for a given initial state.

2.7.1 Algorithm to generate quantum walk from the adjacency matrix of a graph

The problem solved by the algorithm outlined in this section can be stated thus: Given the adjacency matrix of a graph, generate appropriate time evolution operators to simulate a discrete time quantum walk over that structure. Mathematically, this corresponds to performing the appropriate matrix operations. As should be clear from the definition of a quantum walk, two such operators are required, the coin operator $\mathbf{C}$, and the shift operator $\mathbf{S}$. The coin operator is simpler to generate, so I describe this first.

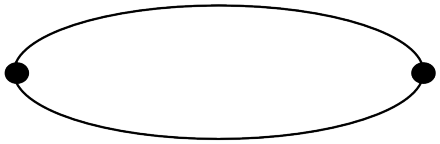


Figure 2.5: The cycle of two nodes

2.7.2 Generating the coin operation

The coin operator acts locally on each node of the graph. Due to the structure and ordering of the direct sum required to apply a coin to each vertex, we know that the operator will be represented by a block diagonal matrix. Each block acts on a different node of the graph,

and the block's dimension will be equal to the degree of that node. In order to generate the operator, the relevant coin acting at each node is generated, and then we must ensure that it is placed in the appropriate position in the large coin matrix. To do this, the degree of each node is required, and this can be found easily by taking the sum of the row representing that node in the adjacency matrix. Then the operations at each node must be specified, and there is a large amount of freedom here. The simplest case is to have coins of the same type operate at each node. For example, the Grover or DFT operators of appropriate dimensions can be specified easily. In some walks discussed below, different types of coin are used at different nodes. Exceptions for nodes of a specific degree, or even for specific nodes, can easily be added. A very simple example of generating the coin operation can be given if we consider a cycle of two nodes, which in standard notation is referred to as C_2 and is depicted in Figure 2.5. Using the two dimensional DFT operator, also known as the Hadamard used at both nodes gives rise to the following coin operator:

$$\mathbf{C} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{pmatrix} \tag{2.29}$$

2.7.3 Generating the shift operation

In order to ensure that the consistent labelling scheme required to guarantee unitarity of the walk is taken into account, care must be taken in generating the shift operator. For each node, we need not only the degree, but the indices of the other nodes it is joined to. In the programming language Python, which was used for the majority of simulations in this thesis, it is easy to generate a list of these by looping over the relevant row in the adjacency matrix $A_{\mathcal{G}}$, and this list can then be used to specify the ordering of the coin states at that node. This list has the form:

$$\text{list\_1} = ([i, j, k], [l, m] ... [v, w])$$

where the index of each entry specifies which node the nodes i, j, k etc. are attached to. It is also useful to create another list:

$$\text{list\_2} = [0, 3, 5 ...]$$

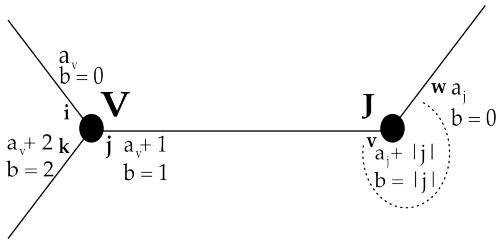


Figure 2.6: Relation between nodes (bold capitals), edge labels (bold lower case) and book keeping quantities for a pair of nodes on an arbitrary graph structure. The dashed line at node j indicates that there are possibly other edges

specifying the index of the first coin state for each node. So the nth entry of list_1 tells us which nodes node n is joined to, and the corresponding entry of list_2 tells us which coin state joins n to the first entry list_1 $[n]$. The dimension of the shift operator is simply the sum of the number of coin states at each node. At a given node v, which corresponds to the row/column numbered v in $A_{\mathcal{G}}$, the correct permutation between coin states is the most difficult part to find. The algorithm can be described thus, with the full Python code shown in Appendix A:

```
Initialise list_1:  an empty list of length |V|

Initialise list_2:  an empty list of length |V|

Set coin_state_counter to zero

For v < |V|

    For x < |v|

       If A_G[v][x] == 1

          Append x to list_1[v]

    list_2 [v] = coin_state_counter

    Add |v| to coin_state_counter
```

```
Shift dimension = coin_state_counter

Initialise operator, a square array of size shift dimension

For v < |V|

    // The first coin state, a_v, at the node v is list_2 [v]

    a_v = list_2[v]

    For x < |v|

        // The node a_v + x joins to, j, is given by list_1[v][x]

        j = list_1[v][x]

        a_j = list_2[j]

        For y < |j|

            // Find the coin state of j which joins to node v

            When list_1[j][y] == v

                b = y

        Shift[a_v + x][a_j + b] = 1
```

In Figure 2.6 the relations between the values used by this algorithm are shown schematically. Performing this routine for each node of the graph gives half of the shift operator. For each nonzero entry (i, j) in the array simply populate the corresponding entry (j, i) and we have the required permutation matrix.

This routine can be used to generate an appropriate shift operator for any standard adjacency matrix, that is, any whose entries are only zeroes and ones, including those which contain self loops. More care must be taken if we wish to generate shift operators for graphs with multiple connections between edges, but the principles remain the same. Clearly, the size of the structure which can be simulated depends on the available memory. I have been able to use the functions created using the above routine to run walks over graphs with more than 7500

nodes for approximately 100 timesteps in less than five minutes using a standard desktop computer. Currently simulations of quantum walks on regular graphs, which can be simulated in a much more compact way, cannot have more than 10^{12} sites. The limits imposed by memory considerations for the size of quantum walk we can classically simulate are discussed in more detail and for a variety of situations in [56].

Once the appropriate shift and coin operators have been created, the only things left to be specified are the number of timesteps the walk should be run for, the initial state of the walker and the information we would like to gain about the walk.

2.7.4 Visualising walks

Due to the vast amounts of data that it is possible to generate regarding quantum walks, methods to make the data more tractable are required. There are various ways to do this, such as only looking at one specific aspect of the walk, as in Chapters 4 and 5. Visualisation of the graphs over which the walks take place, and the evolution of amplitude through the graphs, can also be helpful. I have used two packages to visualise graphs and walks, and briefly describe them here.

Python's `networkx` package is built specifically to facilitate analysis of complex networks. It has a wide range of functions, such as converting other standard data structures (such as a `numpy` array) into its specific graph data structure. Then further analysis of the graph can be done using built in functions. For example graphs can be tested for isomorphism, which facilitated analysis of the graphs generated for the study in Chapter 4. It is also possible to visualise the graphs in a number of ways, as shown for a single graph in Figure 2.8. In cases where there are many graphs to study and little intuition can be gained about them from their adjacency matrix representation, visualisation is extremely helpful.

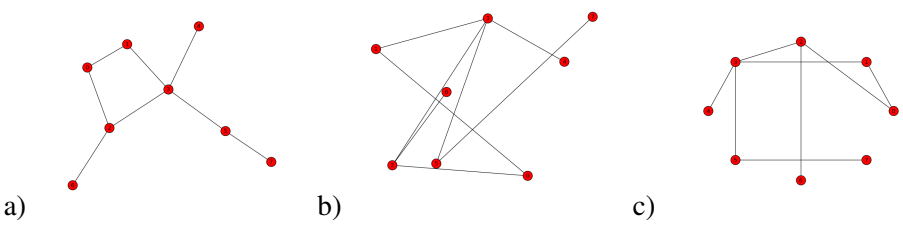
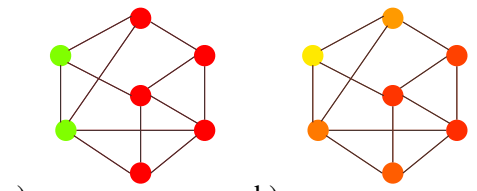


Figure 2.7: Automated visualisation of a graph structure using `networkx` and various options for node distribution: a) standard b) random distribution of nodes c) nodes positioned on a circle

For visualising entire walks, in which case animation is required, the `nodebox` package is useful. This package is exclusively for visualisation, so rather than automatically generating a picture of the graph structure, the choice of visual representation is left to the author and the position of the nodes and edges must be specified manually. Basic forms such as lines and circles are built in, so only the positions and colours of the nodes and edges need specifying. By relating the colour of a node to the probability of the walker being at that node, it is easy to see the walk taking place in the position basis.



a) b)

Figure 2.8: Visualisation using `nodebox` on a graph structure based on a protein pigment complex called the FMO complex with a) amplitude equally distributed between the leftmost two nodes and b) amplitude evolving away from these nodes

Chapter 3

Quantum Computation

3.1 Introduction

The idea of quantum computation dates back to Feynman [59] and the first theoretical quantum computer was proposed by Deutsch [60]. Interest in quantum computation started to take off when specific quantum algorithms were developed and shown to outperform their best known classical counterparts. Examples of such algorithms are Shor's algorithm [61], Grover's algorithm [11] and the Deutsch-Jozsa algorithm [62]. Shor's algorithm finds the prime factors of a given integer in polynomial time, in contrast to the most efficient known classical algorithm, which requires sub exponential time. Grover's algorithm finds a marked state in an unsorted database in $O(\sqrt{N})$ time, and is asymptotically the most efficient quantum algorithm for solving this problem, whereas the optimal classical algorithm requires linear time. The Deutsch-Jozsa algorithm is faster than any deterministic classical counterpart can be. This algorithm determines whether a function is constant, returning the same value for any input, or balanced, in which case for half of inputs it returns 0 and for the other half, it returns 1.

Whilst specific algorithms are useful for demonstrating the advantages of quantum computing over classical computing, shed new insights into the nature of the problems they address, and motivate research into quantum information as a field, there are also strong motivations for looking at more general models of quantum computation. In general, we want to find models which can solve more than one type of problem. Classical digital computers do this easily, and in order to be of use beyond limited settings quantum computers need similar versatility. The ultimate goal would be to experimentally implement a Turing universal quantum computer. In this chapter, I introduce quantum computation first via the most standardly used model, the

circuit model in Section 3.2. As it is of great relevance to later parts of this thesis, the outline of the proof that the discrete time quantum walk can simulate a universal gate set is given in Section 3.3. I then move onto computation in terms of language acceptance, by defining formal languages and the simplest classical computer, the finite state automaton in Section 3.4. The notion of probabilistic language acceptance is then defined. Then the definition of the simplest quantum analogue to a finite state machine is given and the most important theorems regarding its language acceptance properties are stated. The chapter concludes with a few remarks on the motivations, both theoretical and practical, for studying quantum language recognisers.

3.2 Quantum circuits

Classical computation is performed by manipulating *bits*, which take logical values of 0 or 1. The manipulations are performed by circuits composed of logic gates, which implement Boolean functions. Formally a *circuit* is a directed, acyclic graph, with the nodes corresponding to the gates. A directed acyclic graph contains no directed cycles, it can contain cycles. Circuits are often represented pictorially, with individual bits being represented by wires and operations performed on those, and perhaps other, bits as rectangles bisecting the wires. Generally the bits are considered to move right along the wire, so the leftmost gate is the first applied. An example gate is the NAND gate, which can be used to perform universal computation. This gate returns a 0 only when both inputs are 1, which is often interpreted as having a false output only when both inputs are true. There also exist gates, such as the Toffoli gate, a three bit gate sometimes called the 'controlled-controlled-NOT' gate, which are both universal (for classical, not quantum, computation) and logically reversible [63]. This work was motivated by the proof that any logically irreversible operation generates heat [64], so one way to minimise heat generation in computers would be to use reversible operations. However, the results showing that classical computing systems could be logically reversible formed part of the inspiration to consider quantum mechanical computers. Quantum circuits are conceptually identical to classical reversible circuits. A simple example of a small arbitrary quantum circuit is shown in Figure 3.1.

In quantum computation, instead of bits, the fundamental units of information are known as *qubits*. Qubits are represented by norm one vectors in a two dimensional Hilbert space. They can be in any linear combination, otherwise known as *superposition*, of the values 0 and 1. A

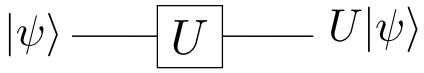


Figure 3.1: The quantum circuit which applies the unitary operation U to qubit in state $|\psi\rangle$

general qubit is written

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{3.1}$$

where α and $\beta \in \mathbb{C}$ are such that $\alpha\alpha^* + \beta\beta^* = 1$ and the vectors in Dirac notation, $|0\rangle$ and $|1\rangle$, represent the logical states 0 and 1. In this notation the vector dual, or conjugate transpose, of $|\psi\rangle$ is written $\langle\psi|$. The vector dual is required when calculating the effects of measurements on quantum states.

In quantum circuits the gates are unitary operators. Operations on individual qubits are represented by 2×2 matrices, for instance the phase shift gate

$$R_\theta = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{pmatrix}. \tag{3.2}$$

This gates leaves state $|0\rangle$ unaltered whilst adding a phase to the $|1\rangle$ state, taking it to $e^{i\theta}|1\rangle$. This corresponds to rotating a qubit by θ radians in a horizontal circle in its pictorial representation known as the Bloch sphere, see Figure 3.2.

In order to perform universal quantum computation, a gate set must include gates which operate on more than one qubit at a time. In fact, only one two qubit gate is required to form a universal gate set, the controlled NOT gate:

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{3.3}$$

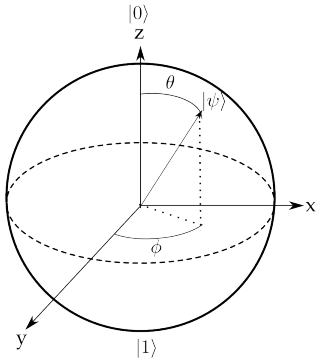


Figure 3.2: The Bloch sphere representation of the qubit, which can occupy any position on the surface of the sphere

which flips the logical state of the second qubit when the first qubit is in state $|1\rangle$, hence $|10\rangle$ is mapped to $|11\rangle$. Together with the Hadamard operator, reproduced from Chapter 2 for clarity

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{3.4}$$

the phase shift and CNOT gates form a universal gate set [65]. The universal gate set shown here is by no means unique [60; 66; 67; 68]. The exclusive-OR gate combined with a set of single qubit gates is another such set [67]. Any two qubit gate which generates entanglement between qubits, i.e. if they start in a product state, then they can no longer be written as such after the gate is applied, has been shown to be universal for quantum computation. This requirement has been shown to be necessary and sufficient for universal quantum computation [69].

3.3 Simulation of quantum circuits by quantum walks

As part of the focus of this thesis is about the applicability of quantum walks to problems in quantum computation, I now turn to their relation to the quantum circuit model. We have already noted in the introduction to Chapter 2 that quantum walks have been used to design specific algorithms. Both discrete and continuous time quantum walks are universal for quantum computation. This was first shown by Childs in [21] for the continuous time walk. Using an analysis of scattering theory on graphs, he was able to develop analogues to the wires along

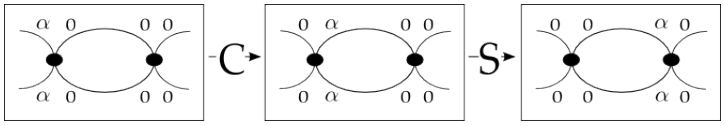


Figure 3.3: One step of walk simulating wires in a quantum circuit by using the Grover coin

which qubits propagate in the circuit model. Widgets attached to these wires were then shown to simulate the universal gate set.

The work in Chapter 8 was in part inspired by the fact that the discrete time quantum walk is a computational primitive. This implies that its applicability to computer science is not limited to the circuit model, as I show by construction in Chapter 8. As the work in this thesis is strongly related to the fact that the discrete time walk is universal for quantum computation, the proof of this is now stated.

Theorem 3.3.0.1. *(Lovett et al.) The discrete time quantum walk simulates the wires and elementary gate set required for universal quantum computation.*

Proof. To simulate the wires, four dimensional nodes, as indicated in Figure 3.3, are used. When the Grover coin is used, this gives the required unidirectional, deterministic evolution, as can be seen by:

$$\begin{pmatrix} \frac{-1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{-1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{-1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{-1}{2} \end{pmatrix} \begin{pmatrix} \alpha \\ \alpha \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \alpha \\ \alpha \end{pmatrix}. \tag{3.5}$$

The vector here represents the coin states on a node of the wire. As there are two edges incident on each node, a and b, and two leaving, transmitting amplitude to edges a and b of the next node, the effect of the Grover operator is to move amplitude from both 'entering' coin states to both 'leaving' coin states. For reasons which will shortly become clear, the Grover operator is in fact multiplied by $e^{-i\frac{\pi}{4}}$. As this adds a global phase it does not affect the evolution. Equation 3.5 holds only when the amplitude in the two populated states is equal in magnitude and phase. The walk must be initialised to ensure this is the case. If the initial state is $\beta|0\rangle + \eta|1\rangle$ such that $|\beta|^2 + |\eta|^2 = 1$ then the amplitude for each qubit will be equally split along edges a and b:

$$|\psi_{initial}\rangle = \frac{1}{\sqrt{2}}(\beta|0\rangle_a + \beta|0\rangle_b + \eta|1\rangle_a + \eta|1\rangle_b) \tag{3.6}$$

This state could represent, for example, the amplitude at the two leftmost nodes in Figure 3.4 (b). To simplify notation below, the $\frac{1}{\sqrt{2}}$ is omitted.

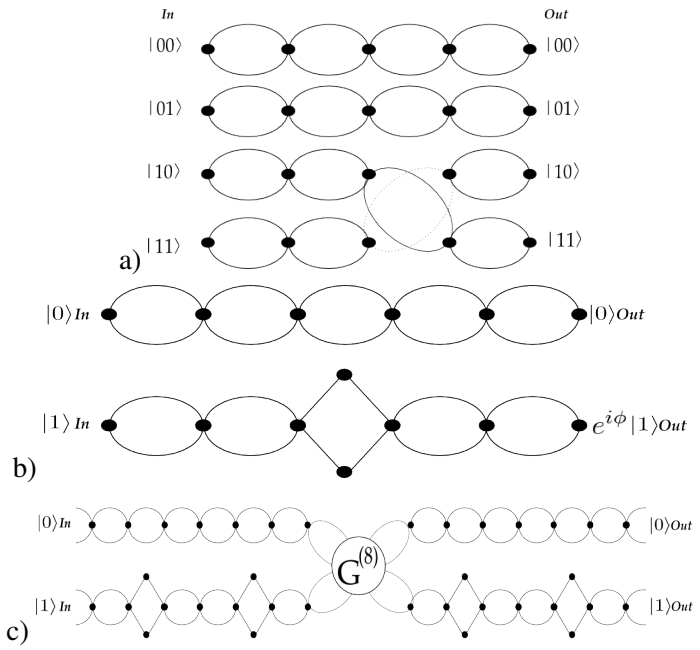


Figure 3.4: Discrete time quantum walk version of a) the controlled not gate b) the phase gate and c) the Hadamard gate

Now graphs and coins which simulate a universal gate set are required. The CNOT gate is trivial to implement, as the wires representing states $|10\rangle$ and $|11\rangle$ are simply swapped, as in Figure 3.4 a). To implement the phase gate, the construction used in Figure 3.4 b) is used. As the Grover operator applied at the two dimensional nodes does not add a phase, the qubit in state $|1\rangle$ picks up a phase of $e^{i\frac{\pi}{4}}$ relative to the qubit in state $|0\rangle$.

This leaves the Hadamard gate, which is the most complex to implement using the discrete time quantum walk. The graph structure, shown in Figure 3.4 c) is comprised of four phase gates along the wire for the qubit in state $|\psi\rangle$, and a specially developed operation which combines the inputs from both basis states and outputs them along the wires accordingly. This

operation is specified by

$$G^{(8)} = \frac{1}{2}\begin{pmatrix} 0 & 0 & 0 & 0 & 1 & i & i & -1 \\ 0 & 0 & 0 & 0 & i & 1 & -1 & i \\ 0 & 0 & 0 & 0 & i & -1 & 1 & i \\ 0 & 0 & 0 & 0 & -1 & i & i & 1 \\ i & -1 & 1 & i & 0 & 0 & 0 & 0 \\ -1 & i & i & 1 & 0 & 0 & 0 & 0 \\ 1 & i & i & -1 & 0 & 0 & 0 & 0 \\ i & 1 & -1 & i & 0 & 0 & 0 & 0 \end{pmatrix}. \tag{3.7}$$

When the $|1\rangle$ states reach the node where $G^{(8)}$ is applied, they are in states corresponding to $e^{-i\frac{\pi}{2}}\eta|1\rangle$. So after the application of $G^{(8)}$ we have:

$$G^{(8)}\begin{pmatrix} \beta \\ \beta \\ e^{-i\frac{\pi}{2}}\eta \\ e^{-i\frac{\pi}{2}}\eta \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \frac{1}{2}\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ (i-1)\beta + (i+1)e^{-i\frac{\pi}{2}}\eta \\ (i-1)\beta + (i+1)e^{-i\frac{\pi}{2}}\eta \\ (i+1)\beta + (i-1)e^{-i\frac{\pi}{2}}\eta \\ (i+1)\beta + (i-1)e^{-i\frac{\pi}{2}}\eta \end{pmatrix} \tag{3.8}$$

Hence $G^{(8)}$ has the required basis changing properties, and after the further two phase gates are applied the required relative phase of -1 is obtained.

Each of the gates begins and ends with one of the degree four nodes used to construct the wires. This means they can be easily composed and hence circuits of arbitrary size can be simulated. □

The graph structures are larger than their equivalent constructions in the circuit model, as there is one wire for each qubit, the quantum walk model requires 2^n wires to simulate an n-qubit computation. However, on a quantum computer, these 2^n wires can be simulated by n qubits, as a graph with N nodes can be represented by $log_2 N$ qubits. If multiple walkers are used, like in the more recent construction by Childs [22] for the continuous time walk, an exponential number of wires is not required.

3.4 Formal languages

A formal language is a set of words from some alphabet Σ. These languages are generated by sets of formation rules, otherwise known as grammars. Formal grammars dictate how strings in a given language can be formed from the language's alphabet. There is a hierarchy of formal grammars, known as the Chomsky hierarchy. At the bottom of the hierarchy lie the regular languages, which are accepted by finite state automata (FSAs). As this thesis discusses the quantum analogues of FSAs, I now introduce these in some detail.

3.4.1 Finite state automata

Finite-state automata (FSAs) are one of the simplest widely researched models for computation, and serve as the basis for other models of computation. Such automata consist of a machine head which has some computational state, which is controlled by transition functions, and an input tape. The machine head reads input symbols from the tape, and these control the state transition made by the machine head.

Formally an FSA is a 5-tuple:

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

with Q being a finite set of states, Σ being the finite input alphabet, δ:$Q \times \Sigma \rightarrow Q$ being the transition function, q_0 being the initial state of the automaton and $F \subset Q$ being the set of accepting states of the automaton. For convenience one often specifies an *end-marker symbol*, $\$ \notin \Sigma$, thus the tape alphabet is $\Gamma = \Sigma \cup \$$. Each symbol on the tape occupies one tape square.

If the machine is in an 'accepting' state after reading the end-marker symbol, the FSA is said to have made an *accepting computation*. The *language accepted*, or recognized, by an FSA M is the set of words $L(M)$, or input strings, such that M is in an accepting state having read the symbol \$. It should be noted that when one talks about the language accepted by an FSA, one refers to a unique set. However, if a language is accepted by an FSA, that language will also be accepted by other FSAs, so there is no unique FSA accepting a given language. The FSA with the smallest number of states accepting L is called the *minimal* FSA accepting L.

From the definition of δ for individual symbols one can specify a function $\hat{\delta}$ the action of an FSA on an entire string of symbols w, which can include the empty symbol ϵ:

- $\hat{\delta}(q, \epsilon) = q$
- $\hat{\delta}(q, wa) = \delta(\hat{\delta}(q, w), a)$ for all strings w and input alphabet symbols a.

$\hat{\delta}$ agrees with δ for individual symbols, as can be seen by computing $\hat{\delta}(q, a) = \delta(\hat{\delta}(q, \epsilon), a) = \delta(q, a)$.

FSAs accept the simplest type of formal languages, regular languages, which can be accepted by simply examining each symbol in a string from the language in order. These are at the bottom of a hierarchy of formal languages known as the Chomsky hierarchy. Regular languages from alphabet Σ are denoted by *regular expressions* and consist of:

- $\emptyset$: the empty set
- ϵ: the empty symbol
- $a \in \Sigma$: singleton symbol

From regular expressions a and b further regular expressions are formed inductively. If a and b denote regular expressions, their concatenation $a + b$, their union $a \cup b$ and their Kleene closure $\{a, b\}^*$ are regular expressions. The Kleene closure of an alphabet is the set of every string which can be formed from symbols in the alphabet, including the empty string. For example, the regular expression $\{ab\}^*$ denotes the language $\mathcal{L}_{ab} = \{(ab)^m | m \in \mathbb{N}\}$. The Kleene closure means that any word from an alphabet can be accepted by an FSA, but, as we will see, if we wish to only accept words with a certain structure, different models of computation are required.

As regular expressions are closed under concatenation and set union, all languages containing a finite number of symbols are regular. The proof of equivalence between regular expressions and FSAs can be found in any standard introductory textbook to theoretical computer science [70; 71]. There are many variations of FSAs, for example δ can be nondeterministic, or the tape head can be allowed to move both ways along the tape. Whilst these variations can provide gains in computational efficiency, they have been proven not to allow us to solve any problems which cannot be solved by a standard FSA.

The limitations of this model of computation become clear when one considers, for example, the language $\mathcal{L}_{eq} = \{a^m b^m | m \in \mathbb{N}\}$, which cannot be accepted by an FSA. The reason that this language cannot be accepted by an FSA is, heuristically, because some form of memory is required in order to determine whether the number of a's is equal to the number of b's. In

fact, the simplest form of memory possible can be used to perform this task. This form of memory is known as a 'stack' and operates on a 'first in- last out' basis and is included in the definition of a pushdown automaton. The languages accepted by these automata form the stage in the Chomsky hierarchy directly above the regular languages and are called context-free languages. Above these are languages called context-sensitive languages. In terms of language acceptance, a universal computer is one which can accept any recursively enumerable language. Such languages are defined to be subsets S of alphabet Σ for which there is some partial recursive function f which is only defined for inputs in S. Only a universal Turing machine can perform this task. In terms of computational power, these computers are equivalent to the circuit model.

3.4.2 Probabilistic acceptance

As the outcome of a quantum computation is probabilistic, in order to discuss such computations the notion of language acceptance must be extended to capture this. In this case, we simply carry over the definitions used when discussing classical probabilistic automata. These automata are best understood by considering computation as matrix multiplication. If one represents the state of an FSA as a vector, it will be all zeros apart from a single entry equal to one, in the basis state corresponding to the computational state that the FSA is currently in. The transition function then induces a zero-one matrix for every symbol in the input alphabet. If, in state q_x whilst reading symbol σ, the transition function maps the state to q_y, then the value in the induced transition matrix at the position (q_x, q_y) will be one. A *probabilistic automaton* is obtained simply by allowing stochastic transition matrices. This increases the computational power of the automaton, and the languages accepted are called *stochastic languages*. Clearly for a properly specified δ all the transition matrices for an FSA are stochastic, so probabilistic automata accept all regular languages. Probabilistic automata accept more languages than FSAs, so the regular languages are a proper subset of the stochastic languages.

In the probabilistic case language acceptance can be considered in terms of bounded or unbounded error. Formally a language L recognised with *cut-point* $\lambda \in [0, 1)$ by the probabilistic automaton M is $L = \{w \,|\, w \in \Sigma^*,\ f_M(w) > \lambda\}$ where $f_M(w)$ is the probability that word w is accepted by M. M is said to *accept L with bounded error* if there is some $\epsilon > 0$ such that for all $w \in L$, M accepts w with probability greater than $\lambda + \epsilon$ and accepts all words $w \notin L$ with probability less then $\lambda - \epsilon$. ϵ is the *error margin*. If there is no such ϵ then M accepts with *unbounded error*. Clearly, the choice of cut point is important, with lower cut points allowing

more languages to be accepted, but with greater probability of error in determining whether or not the input has indeed been accepted.

3.5 A simple model of quantum computation

Though there are many models of quantum computation [65], in this section I concentrate on the most relevant to the forthcoming chapters of this thesis. In Chapter 7 a specific variation of quantum finite automata (QFAs) is discussed, so the most basic incarnation of the QFA is introduced here.

3.5.1 Quantum finite automata

In [72] a 'Measure-Once'-QFA, or MO-QFA, M is defined by a 5-tuple:

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

where Q is a finite set of n states, Σ is the input alphabet, q_0 is the initial state of the MO-QFA, $F \subset Q$ is the set of accepting states and δ is discussed below. As in the case of the FSA, for convenience an end-marker symbol is used at either end of the word, let this be \$. The input alphabet and the end marker symbol form the tape alphabet Γ. This definition is equivalent to that given by Moore and Crutchfield in [73]. When q_0 is being considered as a vector in the Hilbert space where Q is the basis, rather than just a computational state, it is represented $|q_0\rangle$ and called the *initial state vector*.

The value of the transition function

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

for a 3-tuple (q, a, q') with $a \in \Gamma$ and $q, q' \in Q$ is the *amplitude* of the transition from configuration (q, a) to state q'. The probability of a given transition taking place is calculated by squaring the absolute value of the amplitude of that transition.

For each state the amplitude of the transition function for a given $a \in \Gamma$ can be considered as a matrix entry for the *transition matrix,* the matrix representing the transition operator U_a. This operator must be unitary for it to be considered a valid transition operator for an MO-QFA. For each letter a, δ induces a unitary operator, U_a, defined by:

$$U_a|q\rangle = \sum_{q_i,q_j \in Q} \delta(q_i, a, q_j)|q_j\rangle \tag{3.9}$$

where q_i is the state from which the transition will be made and q_j is the state that the transition might be made to. The induced transition matrix contrasts with the classical version, where the entries to the transition matrix are always 0 or 1.

The *probability of acceptance*, f_M, of an MO-QFA, M, accepting a given input word, w, is given by:

$$f_M(w) = |P_{acc}U_w q_0|^2 = |P_{acc}|\psi_{|w|}\rangle|^2 = \langle\psi_{|w|}|P_{acc}|\psi_{|w|}\rangle \tag{3.10}$$

Where $U_w = U_{w_{|w|}} U_{w_{|w|-1}} \dots U_{w_1}$, represents the successive operations on the initial state vector $|q_0\rangle$ by the transition operators for each symbol of w. The vector $|\psi_l\rangle$, where l is an index of a symbol in the word w, represents the state of M once the operator U_{w_l}, the operator on a substring of w has been applied to the initial state vector. So $|\psi_l\rangle = U_{w_l}|q_0\rangle$ is the state of M after M has read the string $w_1 \dots w_l$. M has read the entire word once it is in state $|\psi_{|w|}\rangle = U_w|q_0\rangle$.

Once the end-marker symbol \$ is read, the state of M must be measured. This measurement is specified by a projection operator, P_{acc}. For an *accepting computation* to be made a projection operator of the form $P_{acc} = \sum_{f\in F} P_f = \sum_{f\in F} |f\rangle\langle f|$ must project the state onto the accepting subspace F. Otherwise M has made a *rejecting computation*, or rejected the input.

Until the projection operator is applied the state of M can be in a non-trivial superposition $\sum_{i=0}^{n} c_i|q_i\rangle$ where the $|q_i\rangle$'s are states in Q and $\sum_{i=0}^{n} c_i = 1$. The *computation path of M for a word w* is the sequence of state vectors $\{|\psi_i\rangle\}_{i\in\mathbb{N}}$ such that $U_{w_i}|\psi_i\rangle = |\psi_{i+1}\rangle$. Each computation starts with just one computation path, as it does not start in a superposition, but operations such as the Quantum Fourier Transform (QFT) cause the path to split. Hence each path is the sequence of states entered by one part of the superposition of the *QFA*. In the absence of interference between paths, each path will then evolve independently. The *computation of M on word w* is the computation path taken by M on a specific instance of the input w. For MO-QFAs the computation paths for a given word will differ only by the final vector in the sequence $\{|\psi_i\rangle\}_{i\in\mathbb{N}}$. This is because the projection of the state vector is the only operation with a nondeterministic outcome performed by any MO-QFA.

The power of MO-QFAs depends on whether the machine must accept with bounded or unbounded error. In line with the definitions in [72] let UMO be the class of languages which can be accepted by MO-QFAs with unbounded error and RMO be the class of languages accepted with bounded error. In the acronyms, MO refers to the fact we are considered MO-QFAs, the R refers to a restricted class of languages, where the restriction is that the language must be accepted with bounded error. Hence the U refers to the fact that this class of languages is not so restricted, and stands for 'unrestricted' in this context. As I will drawing comparisons between the language acceptance properties of another type of QFA, one with 'write only memory' in Chapter 7 below, I now reproduce the proofs concerning which languages are accepted with bounded, and unbounded, error by MO-QFAs.

Languages accepted with bounded error by MO-QFAs

The following theorem shows precisely which languages are in the class RMO, but first requires the definition of a *Group Finite Automaton* (GFA):

Definition 1. *A GFA is an FSA such that for each state $q \in Q$ and every input symbol $a \in \Sigma$ there is exactly one state $q' \in Q$ such that $\delta(q', a) = q$, so δ must be a bijective function.*

In order to prove theorem 3.5.1 below, two lemmas are needed, the first concerns the bound on the error accumulated by repeatedly applying a unitary matrix; the second concerns the variation in distance between the probability distributions arising from measurement of two complex vectors $|\phi\rangle$ and $|\psi\rangle$ with norm 1. These lemmas prove general facts about quantum computations, so apply to all quantum systems fulfilling the requisite criteria rather than just those considered in the case of MO-QFAs.

Lemma 3.5.1.1. *(Brodsky and Pippenger 2008) For any unitary matrix U and $\epsilon > 0$ there is some integer $n > 0$ such that for all vectors v with $|v|^2 \leq 1$, $|(I - U^n)v|^2 < \epsilon$.*

Proof. As all unitary matrices are normal matrices, U^n can be written:

$$U^n = PD^nP^{-1}$$

where P is a unitary matrix and D is a diagonal matrix of the eigenvalues of U such that the jth eigenvalue has the form $e^{i\pi r_j}$. If the eigenvalues represent rotations through rational fractions of π, so r_j is rational, then letting $n = 2\prod_{j=1}^{m} q_j$, where $m = dim(U)$ and q_j is the denominator of r_j, gives $D^n = I$ as required.

If there are $1 \leq l \leq m$ eigenvalues of D corresponding to a rotation through an irrational fraction of π, one can compute n as above for the other $m - l$ eigenvalues to give $D' = D^N$.

The jth element on the diagonal of D' will be either 1 or $e^{i\pi r_j}$ with r_j being some irrational number. Taking a kth power of D' for some $k \in \mathbb{Z}^+$ does not change the entries of D' that are equal to 1, but the other l eigenvalues form a vector which varies through a dense subset of an l-dimensional torus. There will be some k that takes this vector arbitrarily close to $\vec{1}$. So for any $\epsilon' > 0$ there is some k such that $|(I - D'^k)\vec{1}|^2 < \epsilon'$. So:

$$\begin{aligned} |(I - U^{kn})v|^2 &= |(I - PD^{kn}P^{-1})v|^2 \\ &= |P(I - D'^k)P^{-1}x|^2 \leq |(I - D'^k)m\vec{1}|^2 \\ &= m^2|(I - D'^k)\vec{1}|^2 \leq m^2\epsilon' \end{aligned}$$

Selecting ϵ' such that $\epsilon' < \frac{\epsilon}{m^2}$ completes the proof.

□

The second lemma concerns the relation between the variation distance between the probability distributions resulting from the repeated measurement of two vectors and the difference between their Euclidean distance. The total variation distance between two probability distributions tells us, informally, the maximum difference between the probabilities assigned by the distributions to the same event.

Lemma 3.5.1.2. *(Bernstein and Vazirani 1997) If $|\phi\rangle$ and $|\psi\rangle$ are two complex vectors such that $||\phi\rangle|^2 = ||\psi\rangle|^2 = 1$ and $||\phi\rangle - |\psi\rangle|^2 < \epsilon$ then the total variation distance between the probability distributions arising from measuring $|\phi\rangle$ and $|\psi\rangle$ is at most 4ϵ.*

Proof. Let $|\phi\rangle = \sum_{i=0}^{n} a_i|q_i\rangle$ and $|\psi\rangle = \sum_{i=0}^{n} b_i|q_i\rangle$ where n is the number of basis vectors in the Hilbert space. The probability of observing each of the $|q_i\rangle$'s when measuring $|\phi\rangle$ is $|a_i|^2$ and when measuring $|\psi\rangle$ is $|b_i|^2$. Let $\theta = |\psi\rangle - |\phi\rangle = \sum_{i=0}^{n}(a_i - b_i)|q_i\rangle = \sum_{i=0}^{n}\gamma_i|q_i\rangle$, in which case $|b_i|^2$ can be rewritten:

$$b_i b_i^* = (a_i + \gamma_i)(a_i + \gamma_i)^* = |a_i|^2 + |\gamma_i|^2 + a_i\gamma_i^* + a_i^*\gamma_i$$

which means that the total variation distance between the probability distributions arising from measurement of ϕ and ψ is bounded by

$$\sum_{i=0}^{n} |\gamma_i|^2 + |a_i\gamma_i^*| + |a_i^*\gamma_i| \leq |\gamma_i|^2 + \langle a|\gamma\rangle + \langle\gamma|a\rangle \leq \epsilon^2 + 2\epsilon$$

as the superpositions have unit length, $\epsilon \leq 2$. □

Now we have covered the requisite material to prove the main theorem about language acceptance for MO-QFAs.

Theorem 3.5.1.1. *A language is accepted by an MO-QFA with bounded error if and only if it is accepted by a GFA.*

Proof. The 'if' direction of the theorem is trivial because the transition functions for GFAs can also be validly treated as transition functions for MO-QFAs.

The 'only if' direction is shown by contradiction, so assume that a language L is accepted with bounded error with margin ϵ by some MO-QFA, $M = (Q, \Sigma, \delta, q_0, F)$ but that L is not accepted by any GFA. L must be regular, as each MO-QFA can be simulated by another form of QFA called a 'Measure-Many QFA' and these only accept a proper subset of the regular languages (see Propositions 6 and 7 of [74]). Suppose strings x and y take M to the same configuration, then the probabilities of M accepting xz and M accepting yz are equal. So by the fact that regular languages are closed under concatenation $xz \in L$ if and only if $yz \in L$. This means that, by the Myhill-Nerode theorem, the space of reachable configurations during M's computation can be partitioned into a finite number of equivalence classes. The Myhill-Nerode theorem states that a language L is regular if and only if there exists a relation on L which divides the set of strings into a finite number of equivalence classes. This relation is called a right invariant equivalence relation.

Suppose $|\phi\rangle$ and $|\psi\rangle$ are reachable configurations of M and $\sim_L$ is the right invariant equivalence relation induced by L. By assumption, L cannot be accepted by a GFA, which means that there must be distinct equivalence classes $[y]$ and $[y']$ with distinguishing string z, an equivalence class $[x]$ and a symbol $a \in \Sigma$ such that $[ya] \sim_L [y'a] \sim_L [x]$. If the transition operator for a is U_a, $|\phi\rangle \in [y]$ and $|\psi\rangle \in [y']$ then $U_a|\psi\rangle \in [x]$ and $U_a|\phi\rangle \in [x]$.

By lemma 3.5.1.1 there is an integer $k > 0$ such that $|(I - U_a^k)|\phi\rangle|^2 < \frac{\epsilon}{4}$ and $|(I - U_a^k)|\psi\rangle|^2 < \frac{\epsilon}{4}$. Therefore $U_a^k|\phi\rangle \in [y]$ because if

$$|(I - U_a^k)|\phi\rangle|^2 = ||\phi\rangle - U^{kn}|\phi\rangle|^2 \ = |V(|\phi\rangle - U^{kn})|\phi\rangle|^2 < \tfrac{\epsilon}{4}$$

with V being an arbitrary unitary matrix. By lemma 3.5.1.2 above, the probability of $VU_a^k|\phi\rangle$ being measured in a given state is within ϵ, i.e. the error margin, of $V|\phi\rangle$ being observed in the same state. The same goes for $U_a^k|\phi\rangle \in [y']$. Hence $[y] \sim_L [ya^k]$ and $[y'] \sim_L [y'a^k]$.

By assuming that $[ya] \sim_L [y'a] \sim_L [x]$ it has been shown that $[y] \sim_L [ya^k]$ and $[y'] \sim_L [y'a^k]$, so $[y] \sim_L [xa^{k-1}] \sim_L [y']$. This means that $a^{k-1}z$ should partition $[x]$ into two distinct equivalence classes, contradicting the fact that $[x]$ is a single equivalence class. Hence the assumption that L is accepted by an MO-QFA but not a GFA cannot be correct. □

Languages accepted with unbounded error by MO-QFAs

This is the final part of the section on MO-QFAs and in it the proof that a language not accepted by a GFA is accepted with unbounded error by an MO-QFA is given. This means that the class RMO is a proper subclass of UMO.

Theorem 3.5.1.2. *(Brodsky and Pippenger 2000) The language* $L = \{w \in \{a,b\}^* | |w|_a \neq |w|_b\}$ *is accepted with cut-point 0 by a 2-state MO-QFA M. The notation* $|w|_i$ *indicates the number of symbols i which occur in word w.*

Proof. Let $M = (Q, \Sigma, \delta, q_0, F)$ with $Q = \{q_0, q_1\}$, $\Sigma = \{a, b\}$ $F = \{q_1\}$ and δ defined by the following matrices:

$$U_a = U_b^{-1} = \begin{bmatrix} cos\,\alpha & sin\,\alpha \\ -sin\,\alpha & cos\,\alpha \end{bmatrix}$$

where α is some irrational fraction of π. U_a causes a rotation of the state vector of M, and since α is an irrational fraction of π there is only one k such that $U_a^k|q_0\rangle = |q_0\rangle$, which is $k = 0$. This is also the case for U_b, so $U_w|q_0\rangle = |q_0\rangle$ where w is the input to M if and only if the number of rotations induced by M reading a is equal to the number of rotations induced by M reading b. This will only be the case if $|x|_a = |x|_b$. If $|x|_a \neq |x|_b$ then there is a nonzero probability that M will be in state q_1 once it finishes reading w. □

In [72] a sketch proof is given that all languages accepted with unbounded error by an MO-QFA are accepted with unbounded error by some PFA; and that all languages accepted with bounded error by an MO-QFA can also be accepted with bounded error by a PFA. Hence, MO-QFAs are exactly as powerful as classical PFAs. To conclude this section I state the theorem without proof.

Theorem 3.5.1.3. *(Brodsky and Pippenger 2000) If M is an MO-QFA accepting language L with cut-point* λ *then:*

- *There is a PFA which accepts L with cut-point* λ'
- *If M accepts L with bounded error then there exists a PFA which accepts L with bounded error.*

3.6 Language acceptance models in the context of quantum computation

From a theoretical perspective, there remain many open problems relating to language acceptance models in quantum computation. This should become clear in later chapters, as we see that unlike in the classical case, modifying a QFA leads to dramatic differences in language acceptance properties. This means that there is, thus far, no clear cut quantum analogue of the Chomsky hierarchy of langauges. The reasons as to why this is the case are still unclear, and a better understanding of why this is the case, or development of a similar quantum hierarchy, may provide new insights into both formal language theory and quantum computation. The circuit model and its relation to the quantum Turing machine [75; 76] is well understood, and hence does not raise such questions for the theoretical computer scientist.

This may not motivate the more practically minded to take an interest in quantum language recognisers. However quantum circuits have, despite much effort, not yet been realised experimentally beyond a few qubits. The best quantum computation devices currently in existence are based on liquid state NMR [77]. Other models of computation better model these devices, in particular the Latvian Quantum Finite Automaton (LQFA) [78], which has been specially designed for that purpose. Hence the development of more computationally powerful QFAs and classification of the types of problems they can solve may help us to build more powerful quantum computers. Whilst a general purpose quantum computer is the eventual goal, universality is not required in order for quantum computers to be useful, as should be clear from the speedups they provide for common algorithms.

Chapter 4

Quantum walks over small cycles

4.1 Introduction

In this chapter I outline results from investigations into one particular aspect of the transport properties of quantum walks: perfect state transfer. The problem of quantum state transfer was first posed by Bose [29] who considered using spin chains to transmit quantum states in quantum computers. Since then, quantum, and in particular perfect state transfer has been shown to be of interest for other reasons. It is a necessary ingredient in the quantum walk implementation of the universal quantum gate set, developed initially by Childs [21] for the continuous time walk, then by Lovett *et al* [23] for the discrete time walk. Quantum walks with good transport properties when undergoing decoherence may provide a model for exciton transport in photosynthetic complexes, as suggested by Mohseni *et al* [34], and in dendrimers, as suggested by Mülken *et al* [79; 80]. For modelling charge and energy transfer, the important factor is total transmitted amplitude, the phase does not also have to be transmitted.

Most efforts to classify when perfect state transfer can occur have used the continuous time quantum walk model, as the addition of the coin states to the discrete time model makes for more difficult analytical solutions [30; 58; 81; 82]. The purpose of the work presented here is to investigate how strongly the presence of perfect state transfer depends on the precise graph structure that the walk takes place over, and look for particular structural modifications of graphs which do not affect their perfect state transfer properties. Both discrete and continuous time walks were investigated but with far more attention paid to the less well examined discrete time case. The first structures examined were based on cycles with $n = 4$, 6 and 8 and diamond chains, as these are known to exhibit perfect state transfer [30; 83]. As the variations tested did

not give rise to much intuition about which types of graphs have perfect state transfer, a more systematic study was then initiated.

Whether we are looking for perfect state transfer or high amplitude transfer, the basis of the problem is the same: find graphs with pairs of points between which the desired transport takes place. The graphs considered here are all unweighted. Unweighted graphs are abstractions of unmodulated spin systems, with the edges representing coupling between spins [29; 51; 84]. Periodicity is closely related to perfect state transfer. A graph that has perfect state transfer will also be periodic for any continuous time walk [81] , as well as for discrete time walks over symmetric graphs.

The chapter proceeds as follows: Firstly previous results concerning perfect state transfer in both the discrete and continuous time case are outlined in Section 4.2; then results from the preliminary investigations into variants of diamond chains and cycles are given. The bulk of the results section concerns which structural modifications to C_4, C_6 and C_8 admit perfect state transfer and under what conditions. The chapter concludes with some brief discussion and outline of potential avenues for future research.

4.2 Prior Results

Some specific graphs arising in this and the following chapter are $\overline{K_2}$; P_n, the path with n nodes and C_n, the cycle with n nodes. Therefore to put the results presented in context, previous results concerning these graphs are discussed. It is clear that as there are no edges joining the vertices of $\overline{K_n}$, quantum walks on this structure do not evolve in the position basis. The complete graph K_n has been shown not to exhibit perfect state transfer in the continuous time case, but this can be achieved by the removal of an edge [85].

4.2.1 Discrete Time Quantum Walk

The quantum walk on the path P_n with $n = \infty$, or $n > 2T$ if the amplitude is initially situated at the middle vertex, is simply the quantum walk on the line [7]. The dynamics of the walk on the line are well known. The effect of varying the initial coin state, and coin operator, is investigated analytically by Bach *et al* in [86] and discussed by Tregenna *et al* in [43]. It is found that for a given coin, the whole range of dynamics available to the quantum walk on the line can be observed by varying the initial state. The evolution on the infinite line is solved exactly by Ambainis *et al* [7]. The most notable fact about the walk on the line is that the

standard deviation of the position of the walker varies proportionally to T, as opposed to with $\sqrt{T}$ in the classical case.

Discrete time quantum walks along finite paths P_n do not in general admit perfect state transfer for non trivial coin operators, i.e., those not equal to the Pauli X operator, which is also the 2 dimensional Grover operator. Inspired by the results from [51] indicated in Section 4.2.2 below, for the continuous time case, self loops were added to the ends of paths to see if this improved transport. Though adding self loops to the ends of P_4 created a walk with perfect state transfer, and hence due to symmetry, periodicity, adding self loops in general did not improve transport across the paths.

As very few cycles admit perfect state transfer for either the continuous, [52; 58], or discrete time walk, [83], the dynamics of quantum walks on C_n are more often analysed in terms of mixing times [44; 87; 88; 89; 90; 91]. Mixing times are the time which the walker takes to get arbitrarily close to its time averaged distribution, and are not of interest in the context of this chapter.

Periodicity in cycles was first noted for C_4 [83]. In cycles with even $n \leq 10$ and a suitable coin, periodicity is observed and perfect state transfer occurs halfway through the period. Varying n can have a dramatic effect on the dynamics of the walk, doubling it to go from $n = 8$ to $n = 16$ turns periodic evolution into highly irregular evolution. The quantum walk on the cycle has been solved analytically [44] by taking the Fourier transform of the walker. The following condition must be met in order for the walk on C_n to have period Ω:

$$\cos\left(\frac{\pi j}{\Omega}\right) = \sqrt{\rho}\cos\left(\frac{2\pi k}{n} - \frac{\pi m}{\Omega}\right) \quad \forall k \tag{4.1}$$

where ρ is the bias of the coin operator (see Equation 2.9 of Chapter 2), and m is an integer specifying the relative phases in the coin operator via the identity $m\pi/\Omega - \pi/2 = \delta = (\theta + \phi)/2$. The Fourier variable is j and can be different for each k, and its parity must be the same as that of m. Solving this for all k is clearly difficult, though solutions for $n \leq 10$ have been obtained and are summarised in [30]. Whilst Fourier techniques are excellent for analysing walks on graphs with some periodicity, they are not so suitable for analysing more irregular graphs, such as the ones examined in Section 4.3 below.

4.2.2 Continuous Time Quantum Walk

The investigation of perfect state transfer along paths was initiated by Bose [29] where a string of coupled qubits was considered. Later results, by Christandl *et al* [51], show that there is no perfect state transfer between antipodal points on paths of length ≥ 4 with unweighted edges.

Cycles are a type of graph known as an integral circulant. Integral circulants were first examined by So in [82], and results concerning their perfect state transfer properties were developed by Saxena [52] and built upon by Bašić [58]. Integral circulants with odd numbers of nodes cannot have perfect state transfer [52]. The only n-even cycle with perfect state transfer is C_4, in contrast to the discrete time case where it is exhibited up to C_{10}.

Kendon and Tamon [30] review many results, both analytic and numeric, concerning perfect state transfer in the continuous time walk, as well as some for the discrete time case. Kay [32] reviews the necessary and sufficient conditions for systems with nearest neighbour interactions to exhibit perfect state transfer and how these can be used to design systems which implement other protocols.

4.3 Discrete time quantum walks over small structures

Numerical simulations of discrete time quantum walks over a variety of structures based on graphs known to exhibit perfect state transfer under some circumstances were performed, in order to see if related structures displayed similar properties. All of the walks were tested using 3 types of coin operator:

- $\mathcal{O}_1$ = DFT at every node
- $\mathcal{O}_2$ = Grover at every node
- $\mathcal{O}_3$ = H at nodes of degree two and Grover at other nodes

$\mathcal{O}_2$ and $\mathcal{O}_3$ will be identical in graphs with no vertices of degree two. To assess the sensitivity of perfect state transfer to the initial state, walks using 1500 initial states with different configurations of coin states at a particular node, uniformly distributed according to the Haar measure [92], were run for 100 steps. This number of steps is sufficient to determine whether behaviour such as periodicity is exhibited for cycles with less than ten nodes, as the largest period for such cycles is 60 steps in the case of C_5.

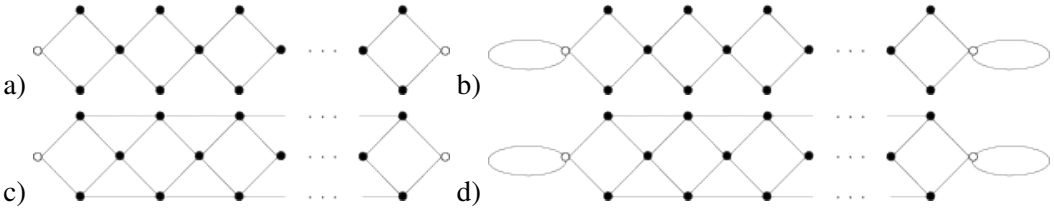


Figure 4.1: The basic diamond chain, (a), and the variants investigated. Vertices with self loops have degree 3. I look for perfect state transfer between the nodes represented by white circles.

4.3.1 Diamond Chains

Chains of n diamonds where all amplitude is initially equally distributed between the coin states at an end vertex are known to exhibit perfect state transfer after $2n$ steps when the Grover coin operator is used [30]. This is because, as shown in Chapter 3, on structures of even degree, if the amplitude is in initially equally distributed between half of the coin states, then it will be transferred into the remaining coin states. This observation enables many other structures exhibiting perfect state transfer to be specified. For example, the graph formed by combining chains of diamonds to create a rectangular structure will exhibit perfect state transfer if all amplitude is initially equally distributed between half of the coin states at the edge of the structure. Three variants of the diamond chain, shown in Figure 4.1, were investigated.

The perfect state transfer in the diamond chain was found to be highly dependent on both the structure and the choice of coin operator. For operators $\mathcal{O}_1$ and $\mathcal{O}_3$, no perfect state transfer occurs for any initial condition tested. The only variant of the chain tested found to have high amplitude transfer, with cutoff 0.9, was that depicted in Figure 4.1 (b), with self loops at either end of the chain. This transfer occurred for only 35 out of the 1500 initial states tested, so even the addition of a single edge to the end nodes along the chain destroys the perfect state transfer. This is because the additional coin state at the initial node makes achieving the required equal superposition between the other two coin states for the Grover coin to have the desired effect impossible. If an additional self loop is added, then the initial node has even degree so perfect state transfer would be recovered. The other variants of the diamond chain did not exhibit perfect state transfer or periodicity for any selection of operator tested, and the maximum probability observed at the end of the chain decreased roughly as n increased.

4.3.2 Variants of Cycles

Very few simple structures are known to exhibit perfect state transfer in the discrete time walk [30] so more complex structures were investigated. As noted in Section 4.2.1, even cycles with $n \leq 10$ have periodicity. In particular, cycles C_4 and C_8 are periodic when the Hadamard operator is used for the coin, and perfect state transfer occurs half way through the period. The cycle C_6 is periodic when a biased coin operator is used [43], though this specific operator is not used for the present discussion. As the present study is to investigate perfect state transfer rather than periodicity, and the walk on C_6 does not have this at any point in its period, this result does not motivate using the biased operator for the investigations I am discussing.

In order to test the sensitivity of the perfect state transfer to the underlying graph structure we first investigated how a small set of specific modifications (see Figure 4.2) affects the perfect state transfer. The modifications tested all resulted in the initial and target nodes of the original cycle being joined, either directly or via intermediate nodes. The intuition behind this selection is that creating new paths between these nodes gives rise to more ways for perfect state transfer to occur. The modifications ensured that the cycles remained symmetric, as some types of symmetry have been proven to be an important factor in determining which walks have perfect state transfer [45; 46]. In fact, most of these modifications destroy the perfect state transfer, the results of the simulations are summarised in Table 4.1 below. Operator $\mathcal{O}_2$ resulted in the most walks with high amplitude transfer, this is due to the fact that it simply performs a swap at nodes of degree 2 and the graphs are symmetrical. Typically the high amplitude transfer observed is very sensitive to initial conditions, for all variants but (a) with high amplitude transfer fewer than 3% of initial states tested exhibited this property. The variants of cycles found to have perfect state transfer, namely (a) and (k), between antipodal nodes are members of the three families of graphs discussed in Chapter 5 below.

Following this preliminary investigation I systematically simulated walks over structures based on C_4, C_6 and C_8. I modified the structures by adding up to four new nodes. This modification worked in the following way: First, a single node was added, in turn, to each existing node of the cycle. Technically this creates the same graph each time, adding a node joined to the cycle by a single edge, but as I always started with amplitude at the same node, the dynamics of the walk will be different depending on the node of the cycle that the new node has been added to. I then generated further edges between that new node, and the nodes of the existing cycle. After generating structures including each new node added to the cycle

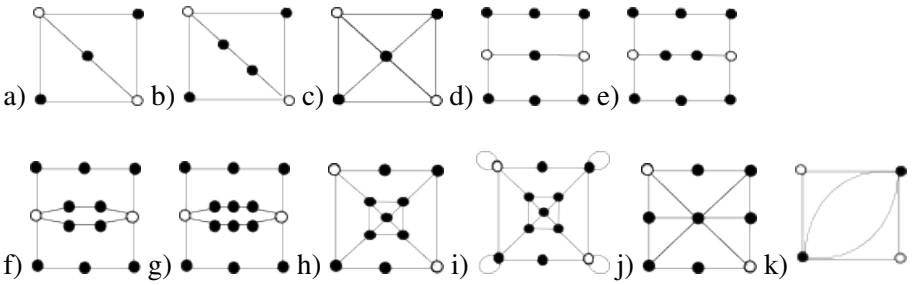


Figure 4.2: Variants of 4 cycles: a), b), c), d), k), 6 cycles: (h), (i) and 8 cycles: e), f), g), j) investigated. Open circles indicate the starting and end nodes, and due to the symmetries of the graphs these are interchangeable.

Graph	Coin (1) max	Coin (2) max	Coin (3) max
(a)	0.99	0.96	0.96
(b)	0.97	0.96	0.96
(c)	0.97	0.96	0.96
(d)	0.83	1.00	0.90
(e)	0.42	1.00	0.70
(f)	0.62	0.95	0.48
(g)	0.51	0.98	0.60
(h)	0.44	0.75	0.75
(i)	0.44	0.77	0.77
(j)	0.20	0.40	0.40
(k)	0.74	0.73	1.00

Table 4.1: Highest amplitude achieved for each operator and structure initially tested

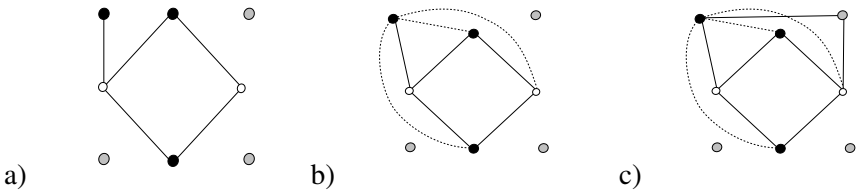


Figure 4.3: Generation of the variants of cycles, shown in the case of C_4: a) attach individual node to an existing node b) attach new node to every existing node c) combine each of these graphs, and add links between new nodes. Dotted lines indicate other connections added for a single node, the rest are omitted for clarity

in every possible way, see Figure 4.3 (b), it was necessary to combine these structures so that the new nodes were joined not only to the cycles, but to each other. This was performed by taking the sets of adjacency matrices which join each new node to the existing cycle (which are technically for identical graphs, but when combined will lead to new structures) and combining each of them in turn, then joining the new nodes to each other, see Figure 4.3 (c). I do not claim that this is the most efficient way to generate these structures. As many duplicates of adjacency matrices, which were removed before simulations were run, and permutations of the same graph are created, however we were more concerned by implementing a systematic study rather than a computationally efficient one.

As I was particularly interested in perfect state transfer between antipodal nodes, I now turn to the results for each set of cycles relating specifically to this aspect of the walks in turn. The limitation of our attention to perfect state transfer between antipodal nodes was motivated by further preliminary studies where an arbitrary node was examined instead. Whilst this study was not sufficiently systematic to make any conclusive remarks (using all examined variants of C_6 and C_8 and examining a single node), it yielded no perfect state transfer.

Results

The results were very similar for C_4, C_6 and C_8 so I discuss them together. The only operator I tested leading to perfect state transfer on the variants of cycles was $\mathcal{O}_2$. As it is clear that any variant of an even cycle which adds nodes only to the target node will have perfect state transfer after a number of steps determined by the size of the cycle using this operator, I do not discuss these results. In fact, the only variants of cycles found to exhibit perfect state transfer were those with modifications to the antipodal nodes. As it is already known that on structures with even degree, if amplitude populates half the coin states and is equal in magnitude and phase,

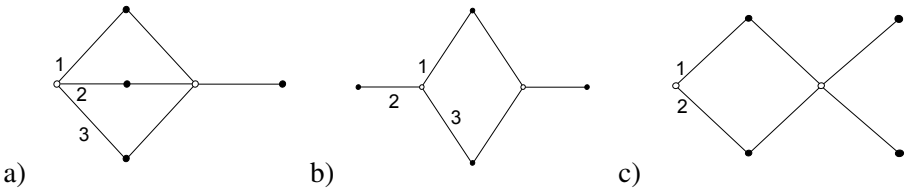


Figure 4.4: Variants of C_4 leading to non-trivial perfect state transfer between antipodal nodes. The edge labels correspond to the coin states they represent when the state at that vertex is written as a vector

then the Grover operator perfectly transmits the amplitude to the other half of the coin states [30], walks which reproduced these results are also not discussed. Two sets of graphs based on C_4 found to exhibit perfect state transfer can be generalised to families, these are discussed in detail in Chapter 5 below. After pruning and generalising the results, this leaves us with three new variants of C_4 exhibiting perfect state transfer, shown in Figure 4.4. Two of the variations also lead to perfect state transfer in C_6 and C_8 for the correct choices of initial conditions. The low number of positive results implies that perfect state transfer is heavily sensitive to the graph structure used for the walk.

The further analysis on these variants has been performed analytically, rather than numerically. This analysis was performed using MAPLE. Having graph structures and the timestep at which to look for perfect state transfer highlighted by the Python simulations, it was simple to generate equations to solve for initial conditions leading to perfect state transfer. Taking this approach, it is possible to avoid narrowly missing incidences of perfect state transfer which occur for initial states close to, but not identical to, those used in the Python simulations. Whilst MAPLE was not suitable for the initial brute force search, it was able to solve these equations, so we can be sure of the results for the graphs on which further analysis was performed.

Perfect state transfer occurs for the graph shown in Figure 4.4 (a) with initial states:

$$|\psi_{\pm}\rangle = \begin{pmatrix} 0 \\ \mp\frac{1}{\sqrt{2}} \\ \pm\frac{1}{\sqrt{2}} \end{pmatrix}$$

The coin states are represented in the order shown in Figure 4.4 a). This perfect state transfer occurs after 50 steps. As the coin states at the target node are not identical to the initial state, this perfect state transfer does not lead to periodicity.

For the graph shown in Figure 4.4 b) the following initial states lead to perfect state transfer after 20 steps:

$$|\psi_{\pm}\rangle = \begin{pmatrix} \pm 0.705 \\ \mp 0.709 \\ \mp 0.005 \end{pmatrix}$$

The decimal expansions for this set of initial conditions do not appear to have any simple algebraic forms. Again, as the coin states at the target node do not replicate the initial conditions, this perfect state transfer does not lead to periodicity. Additionally this variation admits perfect state transfer for any even cycle after the correct number of steps if the initial conditions are selected such that after the coin flip, only the coin states directing amplitude around the cycle are populated. If we call the amplitude in the coin states on the cycle x and y, and require that $xx^* + yy^* = 1$ then the initial states leading to this perfect state transfer when the Grover coin is used can be written:

$$|\psi\rangle = \begin{pmatrix} \frac{2y-x}{3} \\ \frac{2x-y}{3} \\ 2\{\frac{2x-y}{3} + \frac{2y-x}{3}\} \end{pmatrix}$$

For example, if we select $x = y = \frac{1}{\sqrt{2}}$ then the initial condition leading to perfect state transfer will be:

$$|\psi\rangle = \frac{1}{3\sqrt{2}} \begin{pmatrix} 1 \\ 1 \\ 2 \end{pmatrix}$$

This case is one of an entire class of cases, another example would be joining antipodal nodes by a single edge, where the initial condition is selected in order to guarantee that no amplitude is in the coin state for the new edge after the coin flip. In other words, the initial condition is selected so that the new structure does not affect the evolution. Clearly this will only preserve the initial perfect state transfer achieved by deterministically traversing the cycle, and after this, the additional coin states will affect the evolution.

The final graph I discuss, in Figure 4.4 c), exhibits perfect state transfer for all initial conditions. We denote these a and b as there are two coin states at the initial node. There is the expected perfect state transfer after two steps. Then after ten steps the state at the target node is:

$$|\psi_{target}\rangle = \begin{pmatrix} \frac{1}{2}b - \frac{1}{2}a \\ \frac{1}{2}a - \frac{1}{2}b \\ \frac{1}{2}a + \frac{1}{2}b \\ \frac{1}{2}a + \frac{1}{2}b \end{pmatrix} \tag{4.2}$$

Unlike the other walks found, this walk is periodic, after a further two steps the initial condition is recovered. As the form of Equation 4.2 does not depend on the length of the cycle, any even cycle with this modification will lead to perfect state transfer, with a corresponding scaling of period and perfect state transfer time. The same effect can be obtained with the addition of a single vertex joined to the target node by two edges to form a loop.

Continuous time walk on these structures

Whilst the purpose of this section of this thesis is to investigate perfect state transfer in the discrete time walk over small structures, simulations of the corresponding continuous time walks were also run. In the continuous time walk considerations regarding numerical accuracy are more important. To this end, results from simulations obtained from Python were compared to the corresponding simulations in MATLAB. The Python simulations were based on numerical integration, whereas the MATLAB simulations used exponentiation of the time evolution operator. The results of this comparison, where 1.000 in Python became 0.997 in MATLAB, indicate that our methods are not suitable for studying perfect state transfer in the continuous time walk. However, the methods used can help in narrowing down which graphs to look for perfect state transfer in. For example, the families discussed in the following chapter were highlighted by these simulations, and with further analytic work I was able to prove that perfect state transfer did take place. As the cut-points used to test for perfect state transfer were the same in the discrete and continuous time case, we were able to draw one comparison- namely that very high amplitude transfer is more common in the continuous time walk than the discrete time walk, for instance being admitted by 16 variants of C_4. Whilst there were commonalities between these 16 variants, such as the majority having a constant number of nodes along any path joining the initial and target node, without knowing precisely whether they admit perfect state transfer concrete conclusions cannot be drawn.

Some refinements can make examining the continuous time case more tractable. As mentioned briefly in Chapter 2 the Python `networkx` package can be used for further analysis of the graph structures created in order to perform the searches discussed in this chapter. In particular, it facilitated the removal of the isomorphisms, which were required for the brute force search as this always started from the same node, meaning evolution on isomorphic graphs would not necessarily be identical. From an initial 1042 graphs based on C_4, 63 were left once the isomorphisms were removed. Whilst removing the isomorphisms does not enable any new results concerning the discrete time walk, it appears to be very useful for the analysis of

the continuous time walks. This is because in order to obtain the analytic expression for the time evolution in the continuous time walk, the eigenvectors and eigenvalues of the adjacency matrix are required. Once these are found, the time evolution in terms of an arbitrary initial condition can be written down, so it is better to remove the isomorphic cases and then solve for specific initial conditions. Examining the eigensystems of the adjacency matrices of the graphs created in detail was, however, beyond the scope of the work undertaken here.

4.4 Conclusion

I investigated discrete time quantum walks over small graphs for a variety of choices of coin operation. The graphs were variations of graphs known to exhibit perfect state transfer under some circumstances, most notably diamond chains and cycles of length 4, 6 and 8. I have systematically investigated how a range of specific variations, namely the addition of up to four nodes, affects the perfect state transfer. I found that in general, varying the graphs in this way destroys the perfect state transfer. There are a small number of simple ways in which adding new nodes to the graph structure does not affect the perfect state transfer, and these held for each of the cycles tested. No dramatically new ways of achieving perfect state transfer were found, despite many of the structures investigated having properties shown analytically to be important for perfect state transfer- namely symmetry in the graph structures and coin operators [45; 46]. As some of these results [46] were obtained for graphs fulfilling very strict criteria, it is not surprising that these properties do not, in general lead to the specification of walks with perfect state transfer.

In terms of future work, if we intend to limit the size of the graphs examined, there is little point in adding further nodes to the graphs tested. Whilst every way of adding the new nodes to the existing cycles was tested, the effects of adding extra connections between the existing vertices of the cycle were not examined. As all of the cases in which perfect state transfer was not destroyed relied on the fact that our operator $\mathcal{O}_2$ is simply a swap operator at nodes of degree 2, increasing the degree of these nodes does not appear to be a good way of varying the structure for the purposes of investigating perfect state transfer. A systematic study of all graphs upto a certain size may be required in order to obtain some potentially more interesting results. Whilst there is also the freedom to vary the coin operators, my choices are natural as they arise in many discussions of the quantum walks [11; 26; 43; 49].

Chapter 5

Quantum walks over three related families of graphs

5.1 Introduction

In this chapter I discuss results which were first discovered during the investigation into walks based on cycles discussed in Chapter 4. These results concern graphs which all contain C_4 as a substructure. I introduce three related families of graphs which have perfect, or high amplitude, state transfer in the continuous time walk, and (in two cases) the discrete time walks. The discovery of such 'families' gives us new ways of adding an arbitrary number of nodes to a graph, without affecting its perfect state transfer properties. More general versions of such methods are known, for example the results by Angeles-Canul *et al* in [93] show that if a set of graphs have the requisite perfect state transfer properties, then their join will also. There are also other ways to specify how to add nodes to an existing graph with particular properties in a specific way to preserve perfect state transfer, most notably the theorem which proves that all n-cubes have perfect state transfer [51]. These results apply to the continuous time walk only. There are far fewer such results in the discrete time walk. The known results exploit the property of the Grover operator which ensures that, on graphs of even degree, if half the coin states are populated and their amplitude is all equal in magnitude and phase, then that amplitude will simply be transmitted to the unpopulated coin states. Hence in Section 5.2.1 I present the first known such method for the discrete time walk which does not rely on the Grover operator.

After introducing the families of graphs, and discussing specific discrete time quantum

walks over them, the robustness of the perfect state transfer to a number of possible variations is tested. The simple relations between the families allow us some insight into the effects of adding edges, which is further crystallised by interpolating between the graphs, as in Section 5.2.4. I also, in the case where I have to use a specific initial state to achieve the perfect state transfer, test the robustness of the transfer to variations in the initial state in Section 5.2.3. The chapter concludes by examining the continuous time quantum walk over the same structures in Section 5.2.5.

5.2 Three related families of graphs admitting perfect state transfer

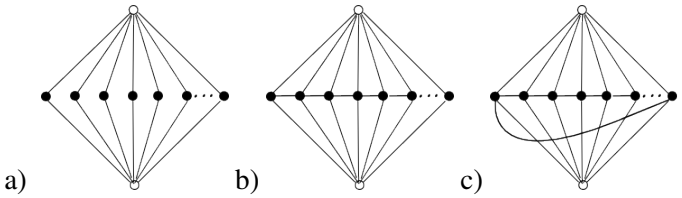


Figure 5.1: The three families: a) $\overline{K_2} + \overline{K_n}$, b) $\overline{K_2} + P_n$ and c) $\overline{K_2} + C_n$. The nodes highlighted with open dots indicate the initial and target nodes, due to the symmetry of the graphs, it does not matter which is which.

In this section I discuss three families of graphs depicted in Figure 5.1. These have the same number of nodes, and all include a join with the graph $\overline{K_2}$. The family $\overline{K_2} + P_n$ has no notable transport properties in the discrete time walk, but exhibits perfect state transfer in the continuous time walk for some choices of n.

5.2.1 Periodicity and perfect state transfer on the graph $\overline{K_2} + \overline{K_n}$

Due to the fact that the 2 dimensional Grover coin is simply a swap operator, it is obvious that $\overline{K_2} + \overline{K_n}$ exhibits perfect state transfer between the nodes of $\overline{K_2}$ in two steps when the Grover coin is used at all nodes and all amplitude is initially at one vertex of $\overline{K_2}$. For this state transfer, it does not matter how the amplitude is distributed between the coin states at the initial vertex, or whether any of this amplitude has an associated phase.

Periodicity can be achieved using less trivial operations, as outlined for various cases in Table 5.1 below. Perfect state transfer to the target vertex is not observed half way through the

period of the walk, except in special cases, usually with $n = 2$ where the graph is equal to C_4. If the DFT coin is used at the nodes of $\overline{K_2} + \overline{K_n}$, but with half of the Hadamards replaced by H_2, all the amplitude returns to the initial vertex when t is half the period, with the state being the DFT of the initial state.

The calculation to show periodicity in walks over $\overline{K_2} + \overline{K_n}$ with initial state $(a_1 \dots a_n)^T$ using the DFT coin is elementary. The state vectors at the initial, final, and intermediate ($\overline{K_n}$) nodes are denoted by $|I\rangle$, $|T\rangle$, and $|\overline{K_n}\rangle$ respectively:

After step 1: $|\overline{K_n}\rangle = \begin{pmatrix} b_i^* \\ 0 \end{pmatrix}$ After step 2: $|I\rangle = |T\rangle = \begin{pmatrix} \frac{b_1}{\sqrt{2}} \\ \vdots \\ \frac{b_n}{\sqrt{2}} \end{pmatrix}$

After step 3: $|\overline{K_n}\rangle = \begin{pmatrix} \frac{a_i}{\sqrt{2}} \\ \frac{a_i}{\sqrt{2}} \end{pmatrix}$ After step 4: Permutation of initial coin states at initial node

* b_i = component corresponding to a_i, where $1 \leq i \leq n$ after the DFT is performed. The steps go the same for Grover coins at nodes of $\overline{K_2}$, but the initial state rather than a permutation of it will arise. This will also be the case for any other unitaries where $UU = \mathbb{I}$.

5.2.2 Periodicity and perfect state transfer in $\overline{K_2} + C_n$

If the Grover coin is used at all vertices, the discrete time quantum walk over graph $\overline{K_2} + C_n$ is periodic, with a period of 12. The walk has perfect state transfer after $6 + 12m$ steps, where $m \in \mathbb{Z}_{\geq 0}$, from one vertex of $\overline{K_2}$ to the other, provided the initial state is an equal superposition of all coin states at the initial vertex.

The initial state is an eigenvector of the Grover coin, with eigenvalue 1, so is unchanged by the first coin operation. After the first step the state at each of the nodes of C_n is of the form:

Coins used at each vertex	Initial state used	*Period*
All DFT	Any	8
Any unitary at $\overline{K_2}$, G_2 in $\overline{K_n}$	Any	4
G_n at $\overline{K_2}$, H at $\overline{K_n}$	Any	4
G_n at $\overline{K_2}$, H, H_2 at half nodes each, $n > 2$, even	See below*	8

Table 5.1: The periods of walks with various coins over $\overline{K_2} + \overline{K_n}$ with all amplitude initially in one of the vertices of $\overline{K_2}$. *Equal superposition over all coin states and all amplitude initially in one coin state.

$$\frac{\gamma}{\sqrt{n|\gamma|}}\begin{pmatrix}1\\0\\0\\0\end{pmatrix} \text{where } \gamma \in \mathbb{C} \tag{5.1}$$

The perfect state transfer in the discrete time walk with the Grover coin on the graph $\overline{K_2} + C_n$ is achieved via the steps below. $|C_n\rangle$ denotes the state vector at the nodes of C_n.

After step 2: $|I\rangle = \begin{pmatrix}-\frac{1}{2}\alpha\\0\\\vdots\\0\end{pmatrix}$ $|T\rangle = \begin{pmatrix}\frac{1}{2}\alpha\\0\\\vdots\\0\end{pmatrix}$ $|C_n\rangle = \begin{pmatrix}0\\0\\\frac{1}{2}\alpha\\\frac{1}{2}\alpha\end{pmatrix}$

After step 3: $|I\rangle = \begin{pmatrix}\frac{1}{2}\alpha\\0\\\vdots\\0\end{pmatrix}$ $|T\rangle = \begin{pmatrix}\frac{1}{2}\alpha\\0\\\vdots\\0\end{pmatrix}$ $|C_n\rangle = \begin{pmatrix}-\frac{1}{2}\alpha\\\frac{1}{2}\alpha\\0\\0\end{pmatrix}$

After step 4: $|I\rangle = \begin{pmatrix}\frac{1}{2}\alpha\\0\\\vdots\\0\end{pmatrix}$ $|T\rangle = \begin{pmatrix}-\frac{1}{2}\alpha\\0\\\vdots\\0\end{pmatrix}$ $|C_n\rangle = \begin{pmatrix}\frac{1}{2}\alpha\\\frac{1}{2}\alpha\\0\\0\end{pmatrix}$

After step 5: $|I\rangle = \begin{pmatrix}0\\0\\\vdots\\0\end{pmatrix}$ $|T\rangle = \begin{pmatrix}0\\0\\\vdots\\0\end{pmatrix}$ $C_n = \begin{pmatrix}\frac{1}{2}\alpha\\-\frac{1}{2}\alpha\\0\\0\end{pmatrix}$

After step 6: $|I\rangle = \begin{pmatrix}0\\0\\\vdots\\0\end{pmatrix}$ $|T\rangle = \begin{pmatrix}\alpha\\\alpha\\\vdots\\\alpha\end{pmatrix}$ $|C_n\rangle = \begin{pmatrix}0\\0\\0\\0\end{pmatrix}$

As we see, after six steps of the walk perfect state transfer is achieved. From the symmetry of $\overline{K_2}+C_n$ it can be seen that the evolution is periodic, with period 12. The choice of coin operator is important in acheiving this perfect state transfer, using only a DFT coin for quantum walks over $\overline{K_2} + C_n$ does not result in any notable transport properties. For the walk over $\overline{K_2} + C_n$ starting with an equal superposition at one vertex of $\overline{K_2}$, the probabilities on the nodes of $\overline{K_2}$ do not depend on n when $n > 1$. The initial state is also important, as I now discuss.

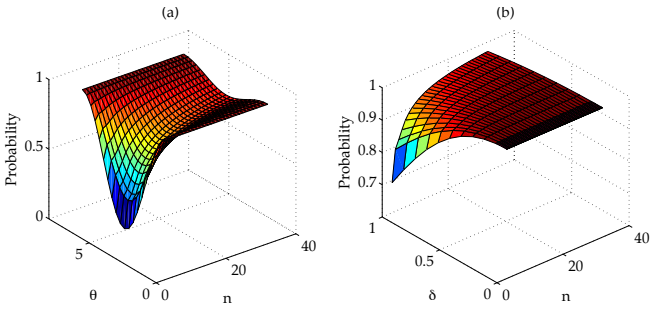


Figure 5.2: Robustness of perfect state transfer to variation in a) initial phase and b) amplitude for one coin state. The size of C_n is n, θ is the phase and δ is the defect in Equation 5.2.

5.2.3 Robustness of perfect state transfer in graphs from $\overline{K_2} + C_n$

Numerical investigations were carried out to test the robustness of the first instance of perfect state transfer, after 6 steps, to variation in initial conditions. The transport was shown to be very robust to variations in initial condition of the form:

$$|\psi_{int}\rangle = \frac{1}{\sqrt{(n - 2\delta + \delta^2)}} \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 - \delta \end{pmatrix} \tag{5.2}$$

As n increases, the robustness increases too, even in the case where $\delta = 1$, where the state is of a qualitatively different form to the standard initial state. Taking $\delta > 1$ adds a phase to the perturbed initial coin state. The robustness of the perfect state transfer to variation in phase of one of the initial coin states was also investigated, with similar findings, see Figure 5.2. Though the perfect state transfer is much less robust to variation in phase, with probability going down to 0.19 for $n = 5$ when we have a phase factor of π, we can again increase the robustness of the transfer by increasing n. For $n > 35$ the probability at the target vertex after 6 steps does not go below 0.9 for any phase. The robustness increases with n because an equal superposition amongst n coin states is used for the initial state, and hence perturbing one part of that superposition results in a smaller relative perturbation as n increases. If the initial state is instead varied by perturbing each coin state by a random $0 \leq \delta \leq 1$ then the robustness decreases slightly with n. For $n = 3$ the amplitude at the target node goes down to 0.82, averaged from 1000 runs, at the perfect state transfer time. As n increases the value tails off at 0.77.

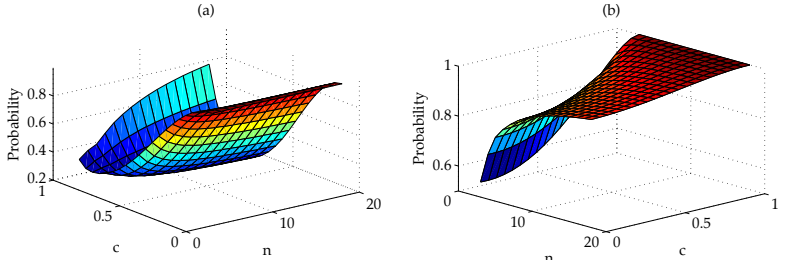


Figure 5.3: Variation of fidelity of state transfer during interpolation between a) Graphs $\overline{K_2} + \overline{K_n}$ and $\overline{K_2} + P_n$; and b) $\overline{K_2} + P_n$ and $\overline{K_2} + C_n$. n is the number of nodes in K_n, P_n or C_n and c is the weighting of the edges that these graphs differ by. The orientations of the graphs are selected for clearest viewing.

The robustness of the transport with respect to decoherence was also tested. The effects of decoherence in the coin state and decoherence in the position state are both independent of n, and probability at the target vertex decays smoothly as the rate increases, recovering the classical distribution for decoherence rate $p = 1$ as expected.

5.2.4 Interpolation between the three families

It is possible to interpolate between graphs with the same vertices but different edges by weighting the edges. To perform quantum walks on such graphs, coins reflecting this edge weighting are required. If we choose to use the Grover coin, the coin should be the Grover operator of dimension $(d - t)$ when no edge is present, and that of dimension d when the edge is fully present, so t is the number of edges being 'switched on' in the interpolation. Although walks using the Grover coin over both $\overline{K_2} + \overline{K_n}$ and $\overline{K_2} + C_n$ display perfect state transfer, adding the edges they differ by gradually by using a coin that interpolates between Grover operators of different dimensions in the way described in [41] destroys this perfect state transfer. The coin is specified by

$$\mathrm{G}_{d,t} = \begin{pmatrix} a & b & b & \dots & b & c & c & c & \dots & c \\ b & a & b & \dots & b & c & c & c & \dots & c \\ . & . & . & \dots & . & . & . & . & \dots & . \\ . & . & . & \dots & . & . & . & . & \dots & . \\ . & . & . & \dots & . & . & . & . & \dots & . \\ b & b & b & \dots & a & c & c & c & \dots & c \\ c & c & c & \dots & c & e & f & f & \dots & f \\ c & c & c & \dots & c & f & e & f & \dots & f \\ . & . & . & \dots & . & . & . & . & \dots & . \\ . & . & . & \dots & . & . & . & . & \dots & . \\ . & . & . & \dots & . & . & . & . & \dots & . \\ c & c & c & \dots & c & f & f & f & \dots & e \end{pmatrix} \quad (5.3)$$

Where $a \ldots f \in \mathbb{R}$ where there are $(d-t)$ 'normal' edges giving rise to blocks of size $(d-t)$ containing $a's$ and $b's$ and t 'tunneling' edges with transitions specified by the block containing $e's$ and $f's$. This coin enables an edge to be 'turned on' with strength c. In order to guarantee that this coin is unitary, certain relations between the parameters are required, details of these can be found in [41]. The effect of turning on the additional edges to go from $\overline{K_2} + \overline{K_n}$ to $\overline{K_2} + C_n$ does not depend on n. The amplitude at the target vertex at the perfect state transfer time decays quickly with c, and rises again very slowly as $c \to 1$. One can also interpolate between $\overline{K_2} + \overline{K_n}$ and $\overline{K_2} + C_n$ by going via $\overline{K_2} + P_n$. This interpolation does not improve the transport properties of the walk over $\overline{K_2} + P_n$, as can be seen in Figure 5.3.

5.2.5 The continuous time walk on $\overline{K_2} + \overline{K_n}$, $\overline{K_2} + C_n$ and $\overline{K_2} + P_n$

In general, a graph exhibiting periodicity, high probability, or perfect state transfer for a specific coin in the discrete time walk is no indication that this will occur for the continuous time walk over the same graph. However for the three families of graphs discussed in this chapter, the continuous time walks exhibit perfect or very high fidelity state transfer.

The continuous time walk on $\overline{K_2} + \overline{K_n}$ displays perfect state transfer for any n, and is hence periodic due to the symmetry of the graph. The perfect state transfer time can be tuned by adjusting n, as n increases, the period decreases. An analytic expression for the period in terms of n was deduced by inspection of the eigenvectors. The eigensystems of the graphs $\overline{K_2} + \overline{K_n}$ have only two nonzero eigenvalues of the form $\sqrt{2n}$, hence only two components of the analytic expression for the time evolution determine the period, regardless of the initial state used. The eigenvectors of the adjacency matrices of graphs from $\overline{K}_2 + \overline{K}_n$ are of the form:

$$|v_+\rangle = \frac{1}{\sqrt{2n}} \begin{pmatrix} 1 \\ \vdots \\ 1 \\ \frac{\sqrt{2n}}{2} \\ \frac{\sqrt{2n}}{2} \end{pmatrix} \qquad |v_n\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 0 \\ \vdots \\ 0 \\ -1 \\ 1 \end{pmatrix}$$

$$|v_-\rangle = \frac{1}{\sqrt{2n}} \begin{pmatrix} 1 \\ \vdots \\ 1 \\ -\frac{\sqrt{2n}}{2} \\ -\frac{\sqrt{2n}}{2} \end{pmatrix} \qquad |v_i\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} -1 \\ \vdots \\ 1 \\ \vdots \\ 0 \\ 0 \end{pmatrix}$$

There are $n-1$ eigenvectors, each having eigenvalue 0, of the form $|v_i\rangle$, with 1 occurring at a different $\overline{P}_n$ basis state for each $i \leq n-1$. The coefficient of these eigenvectors is 0 when solving for the time evolution with the appropriate initial conditions, so they do not contribute to the time evolution. The eigenvector $|v_n\rangle$ also has eigenvalue 0, hence the eigenvectors $|v_+\rangle, |v_-\rangle$, each with eigenvalue $\pm\sqrt{2n}$, are the only ones whose contribution to the evolution varies over time. The period of the graph is $2\pi/\sqrt{2n}$, giving a perfect state transfer time between vertices of $\overline{K_n}$ of $\pi/\sqrt{2n}$. The expression for the time evolution over $\overline{K}_2 + \overline{K}_n$ with adjacency matrix A with the amplitude initially at a vertex of $\overline{K}_2$ is given by:

$$
\begin{aligned}
|\psi(t)\rangle = e^{-itA}|n+1\rangle &= -\frac{1}{\sqrt{2}}|v_n\rangle + \sum_{\pm} \pm\frac{1}{2}e^{-it\sqrt{2n}}|v_\pm\rangle \\
&= \begin{pmatrix} \frac{1}{\sqrt{2n}}(-i\ sin(\sqrt{2n}\ t)) \\ \vdots \\ \frac{1}{\sqrt{2n}}(-i\ sin(\sqrt{2n}\ t)) \\ \frac{1}{2}(cos(\sqrt{2n}\ t)+1) \\ \frac{1}{2}(cos(\sqrt{2n}\ t)-1) \end{pmatrix} \qquad (5.4)
\end{aligned}
$$

where the first n entries are the vertices of $\overline{K_n}$, the $(n+1)^{th}$ entry is the initial vertex, and the $(n+2)^{th}$ entry is the final vertex. This expression applies for any n, including $n = 0$ where the vertices of $\overline{K_n}$ are not connected, so no evolution can occur. The cases for $n = 1$ and $n = 2$, giving rise to the graphs P_3 and C_4 respectively are already known [51].

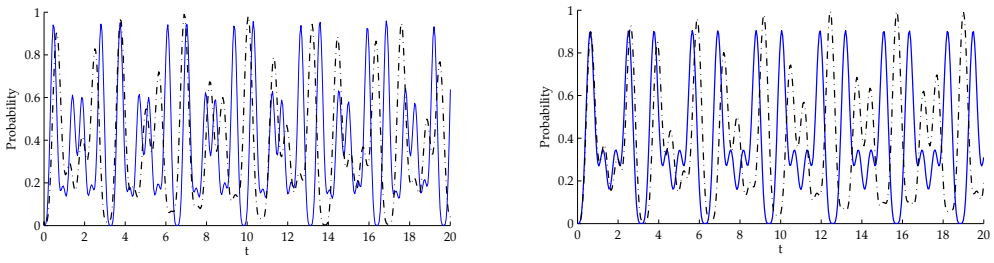


Figure 5.4: Time evolution of probability at the target node for a) graph $\overline{K_2} + P_n$ with $n = 11$ (solid) and $n = 10$ (dotted); b) graph $\overline{K_2} + P_n$ with $n = 21$ (solid) $n = 11$ (dotted).

There is little difference between the evolution over the graphs $\overline{K_2} + P_n$ and $\overline{K_2} + C_n$, as might be expected given that they differ by a single edge. Both exhibit oscillatory motion at the initial and target vertices, as depicted in Figure 5.4, with the oscillations peaking at very high probabilities or unity. Clearly from the choice of n's plotted, perfect state transfer does not occur for every n, so there is no simple relationship between the evolutions for different n's. The eigensystems of these families were too complex to enable a simple expression for the time evolution to be deduced even for a specific choice of n. As for $\overline{K_2} + \overline{K_n}$, the periods of the oscillations in these families get smaller as n increases. Decoherence in the walks over all three families quickly smears out the oscillatory behaviour, as can be seen in Figure 5.5 (a).

(a) (b)

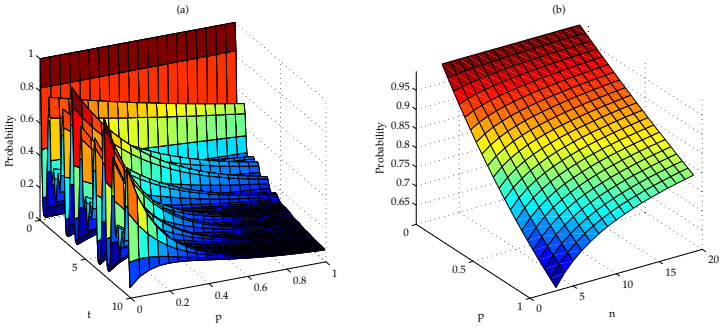


Figure 5.5: a) Time evolution of probability at the target vertex for graph $\overline{K_2} + C_n$ with decoherence rate p. b) Effect of decoherence on the probability at target vertex of graph $\overline{K_2} + \overline{K_n}$ at the first perfect state transfer time for increasing n

5.3 Conclusion

I have presented and analysed in some detail three families of graphs which have interesting transport properties in both the discrete and continuous time walk. Two of the families of graphs, for the right choices of coin operator in the discrete time walk, exhibit periodicity, and in some cases perfect state transfer. I found that these families also exhibit very high amplitude transfer, or perfect state transfer, in the continuous time case. In the case of the discrete time walk over $\overline{K_2}+C_n$ it was found that increasing n improved robustness of the perfect state transfer

to variations in one coin state of the initial state, but not to decoherence. In the continuous time case, n can be used to tune the perfect state transfer time.

Characterising the graphs which admit perfect state transfer and understanding which structural properties of those graphs give rise to perfect state transfer remains an ongoing project. I have found cases where adding an arbitrary number of nodes to a particular graph in the right way preserves its state transfer properties. Ideally general methods of determining whether a graph has perfect state transfer, such as that outlined in [81] for the continuous time case, are required. Due to the coin degrees of freedom, and the choice in coin operator, this is a far more difficult task in the discrete time case. Some analytical methods capable of dealing with the coin degrees of freedom have been developed, but are relatively limited in scope. For example, the methods used by Krovi and Brun in [46] only apply to regular graphs whose edges can be numbered in a certain way. The numerical methods I have used are not so restricted, and thus currently appear to be more suitable for investigating the discrete time quantum walk. Once a discrete time walk result is obtained numerically, it is often very easy to reproduce it analytically, however this will obviously not lead to such general analytical results as those in [46].

Chapter 6

Non-reversal Quantum Walks

6.1 Introduction

The random walks introduced in Chapter 2 concerned idealised walkers with no spatial extension. Whilst these have many uses in modelling physical and biological processes, in some cases we may want to consider walkers which do have spatial extension, and hence cannot move into space which they are already occupying. In this chapter I introduce a quantum version of such a walk, which effectively simulates a dimer whose tail always follows the head. The motivation for studying such a walk is much the same as that for studying the classical version: more realistic simulation of physical systems. There are also potential algorithmic applications, such as searching. Search algorithms based on random walks can be made more efficient by ensuring that they do not revisit positions already visited.

One process which has been modelled by a self-avoiding classical random walk is polymer folding. In this case, the walker is finite rather than infinite, so does not have to avoid every position it has ever visited, but does have to avoid positions which it currently occupies. In some cases, such a restricted model may not be suitable. Continuing the example, polymers can overlap to form loops, but they cannot occupy volume that they already occupy. A type of intermediary between the fully random walk and the self avoiding walk is known as the non-reversal walk. A non-reversing random walker is allowed to occupy spaces on a lattice that it has previously occupied, but is not allowed to return to the point which it has just come from.

In both the classical and quantum case, the self-avoiding or non-reversal walk on the line is trivial. This is because there are only two degrees of freedom in the movement, so if one of those is prohibited by the model, then unidirectional ballistic transport is obtained. The walks

studied in this chapter take place over a square lattice, in which case the dynamics are highly non-trivial.

The chapter proceeds as follows: In Section 6.1.1 classical self avoiding walks are introduced in more detail. Then for the sake of comparison the properties of the quantum walk on the square lattice are discussed. The non-reversal quantum walk is then defined in Section 6.2.1 and its properties are described in Section 6.2.2. I finish by indicating how one might obtain a self-avoiding walker of length three in Section 6.3 and briefly summarising the results and potential avenues for future work in the conclusion.

6.1.1 Classical self-avoiding random walks

The classical self-avoiding walk has proven difficult to treat analytically, hence the results concerning it have all so far been numerical [94] and there remain many open questions about it. Even enumerating the number of self avoiding walks, despite them being so rare that coming upon one by mistake when examining a random walk is highly improbable, has proven very difficult. Whilst some facts are clear, for example that $c_{n+m} \leq c_m + c_n$ where c_n is the number of walks with n steps. The set of walks of length n concatenated with the set of walks of length m contains not only the self avoiding walks of length $n + m$ but some which overlap, hence the inequality. However, determining the precise number of walks is difficult, though bounds have been established. The number of self avoiding walks must be less than the number of non-reversal walks, as these include the self avoiding walk as a subset. Additionally it is possible to construct subsets of self avoiding walks which grow as 2^n, so we know that there are between 2^n and 3^n self avoiding random walks. The best evidence so far suggests that there are 2.638^n self avoiding walks of length n, and this is provided as a non-rigorous estimate in [95]. The best evidence for this value was obtained by enumerating each such walk of length up to 51 and required a 1024 processor supercomputer [96]. Without new algorithms it is unlikely that we will be able to enumerate much further than this. Guttmann [96] obtained 14,059,415,980,606,050,644,844 walks using this method, however these walks make up less than 1 in 240 million of the possible random walks of length 51.

As even counting the walks has proved difficult, it is unsurprising that little is known regarding other properties. The property we are most interested in when comparing walks is the standard deviation. The mean squared displacement is conjectured to be $n^{3/2}$ though so far, even a proof that the exponent must be between 1 and 2 is elusive [94].

Another known fact about self avoiding walks demonstrates a key difference between the self avoiding walk and its standard and non-reversal counterparts. This is that the self avoiding walk does not necessarily continue indefinitely. This is because it is possible to reach a lattice site whose only adjacent lattice sites have previously been visited, hence the walker cannot continue. After the first few steps there is a small probability that the walk will end on any subsequent step.

The non-reversal walk is in some ways more tractable. For example, it is clear that on the square lattice there are 3^n such walks, where n is the number of steps traversed by the walker. Its mean squared displacement is $2n$, so it spreads twice as fast as the standard random walk, but slower than the completely self avoiding walk. There is very little literature on the non-reversal walk, and this tends to examine specific characteristics of the walk relevant to the study of polymer chains [97], rather than its general features.

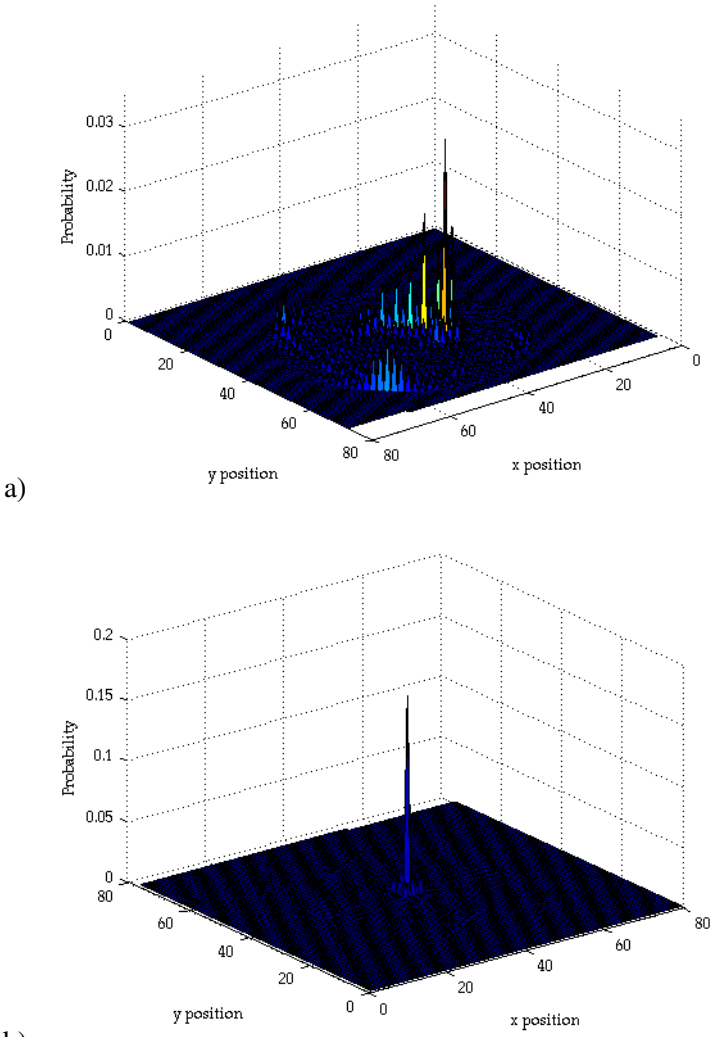


Figure 6.1: Probability distributions arising after 40 steps of a walk on a 2d lattice using a) the DFT coin and b) the Grover coin for the symmetric initial state shown in equation 6.3

6.1.2 Quantum walks on the square lattice

The properties of the discrete time quantum walk on the square lattice were extensively explored in [43] following initial investigations in [57]. In particular they examined the mean

distance at time t:

$$\langle r \rangle_t = \sum_{x,y} \sqrt{x^2 + y^2}\, p(x, y, t) \tag{6.1}$$

so r is the radial distance from the origin, and $p(x, y, t)$ is the probability of finding the walker at position (x, y) at time t. They also characterise the walks in terms of the standard deviation:

$$\sigma = \sqrt{\langle r^2 \rangle - \langle r \rangle^2} \tag{6.2}$$

They use three choices of coin operator. Their first choice behaves like the Hadamard operator for both the 'left/right' ($|l\rangle/|r\rangle$) coin states and the 'up/down' ($|u\rangle/|d\rangle$) coin states. This operator does not mix the two dimensions, so a two dimensional version of the distribution of the walk on the line is obtained. More interestingly, they consider the DFT and Grover operators for a number of initial conditions. These coins are both unbiased, in that they distribute amplitude equally between each coin state. In contrast to the walk on the line, they find that the dynamics of the walk depend strongly on the coin used. Additionally, the dynamics for a specific coin depend strongly on the choice of initial state. The lowest and highest standard deviations obtained for the position of the walker were found using the Grover operator. It was observed that the reason for this is that regardless of the initial state, the distribution forms a central spike, with a ring around it which propagates outwards. The choice of initial condition controls how much amplitude is situated in the central spike, and how much amplitude is situated in the ring. The initial state at the origin for both the distributions plotted in Figure 6.1 is:

$$\begin{aligned} |\psi\rangle &= \frac{1}{2}(|ld\rangle + i|lu\rangle + i|rd\rangle - |ru\rangle) \otimes |0\rangle \\ &= \frac{1}{2}(|l\rangle + i|r\rangle) \otimes (|d\rangle + i|u\rangle) \otimes |0\rangle \end{aligned} \tag{6.3}$$

The authors of [43] studied all unbiased four dimensional unitary operators with entries equal to either $\pm 1/2$ or $\pm i/2$, which when the leading diagonal is selected to be $1/2$ gives 640 unitary operators. These operators were found to fall into ten types, with the DFT, Hadamard and Grover all being of different type. Of all operators tested, the Grover operator was found to give rise to both the largest and smallest standard deviation, depending on the initial state.

6.2 Non-reversal quantum walks

6.2.1 Definition

I will now define our non-reversal quantum walk on the square lattice. Whilst in the treatment of walks on the square lattice in [43] the specification of the shift operation is implied, more care must be taken here to ensure that the coin operator really does what we want it to. Two variations of the shift, ones which shifts amplitude from, say, the 'moving-right' state of one node to the 'moving-right' state of the next, as is used in [57], and one which shifts the same amplitude to the 'moving-left' state of the next node, are considered. The non-reversal walk arises when the 'moving-right' to 'moving-left' permutation is used, this is the shift operation defined in [42].

The state of the walker, Ψ, at time $t \in \mathbb{N}$ is described by a 4-dimensional vector which is a function of each discrete lattice point $(x, y) \in \mathbb{Z}^2$. This is denoted

$$\Psi(x, y, t) = \begin{pmatrix} \psi_1(x, y, t) \\ \psi_2(x, y, t) \\ \psi_3(x, y, t) \\ \psi_4(x, y, t) \end{pmatrix} \tag{6.4}$$

where we have $\sum_{x,y,i} |\psi_i(x, y, t)|^2 = 1$ and each component is a complex function of discrete position and time. The coin operator will then be given by a unitary in $U(4)$. The first coin I consider, albeit briefly, will be called the non-repeating coin, as when using the choice of shift operator which makes the walk more amenable to Fourier space techniques, it never allows amplitude to move in the same direction in two consecutive steps. This coin is defined by:

$$C^{(NF)} = \begin{pmatrix} 0 & \lambda e^{i\alpha} & \gamma e^{i\beta} & f(\lambda,\gamma)e^{i\theta} \\ \lambda e^{-i(\phi+\delta+\alpha)} & 0 & -f(\lambda,\gamma)e^{i(\psi-\theta+\beta)} & \gamma e^{i\psi} \\ -\gamma e^{-i(\delta+\alpha+\psi)} & -f(\lambda,\gamma)e^{i(\phi-\theta+\alpha)} & 0 & \lambda e^{i\phi} \\ f(\lambda,\gamma)e^{i(\theta-\alpha-\psi-\phi-\beta)} & -\gamma e^{i(\delta+\alpha-\beta)} & \lambda e^{i\delta} & 0 \end{pmatrix} \tag{6.5}$$

where all of the variables are real, $0 \leq \gamma^2 + \lambda^2 \leq 1$, and $f(\lambda, \gamma) = \sqrt{1 - (\lambda^2 + \gamma^2)}$. This is the most general $SU(4)$ operator with zeros on the diagonal. A simple example of such a coin, used to produce the probability distribution shown in Figure 6.2, takes $\theta = \phi = \pi$ and $\lambda = \gamma = f(\lambda, \gamma) = \frac{1}{\sqrt{3}}$, with all other phase variables being zero. This leads to the following

coin:

$$C^1 = \frac{1}{\sqrt{3}} \begin{pmatrix} 0 & 1 & 1 & -1 \\ 1 & 0 & 1 & 1 \\ 1 & -1 & 0 & -1 \\ -1 & -1 & 1 & 0 \end{pmatrix} \tag{6.6}$$

We can see that this coin does what it is required to by examining its operation on an arbitrary state

$$|\psi\rangle = \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} \tag{6.7}$$

It is clear that

$$C^1|\psi\rangle = \frac{1}{\sqrt{3}} \begin{pmatrix} b+c-d \\ a+c+d \\ a-b-d \\ c-a-b \end{pmatrix} \tag{6.8}$$

so the initial amplitude in each coin state is redistributed between the other coin states, and none of it is transmitted to the node it came from. In the numerical work below I consider

$$C^{NR} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} C^{NF} \tag{6.9}$$

where clearly C^{NR} is a permutation of C^{NF}. Permuting the coin in this manner has the same effect as permuting the shift operation and gives rise to the non-reversal walk. The permutations arise naturally when one considers the freedom in choice of edge labelling on a 2d lattice. As the arising walk dynamics have notable properties in common I discuss both. In [98] there is an analytical proof that when using C^{NF} the joint (x and y direction) moments are independent of the initial condition. It was not possible to obtain a corresponding proof for the operator C^{NR}, so its properties were investigated numerically, as detailed below.

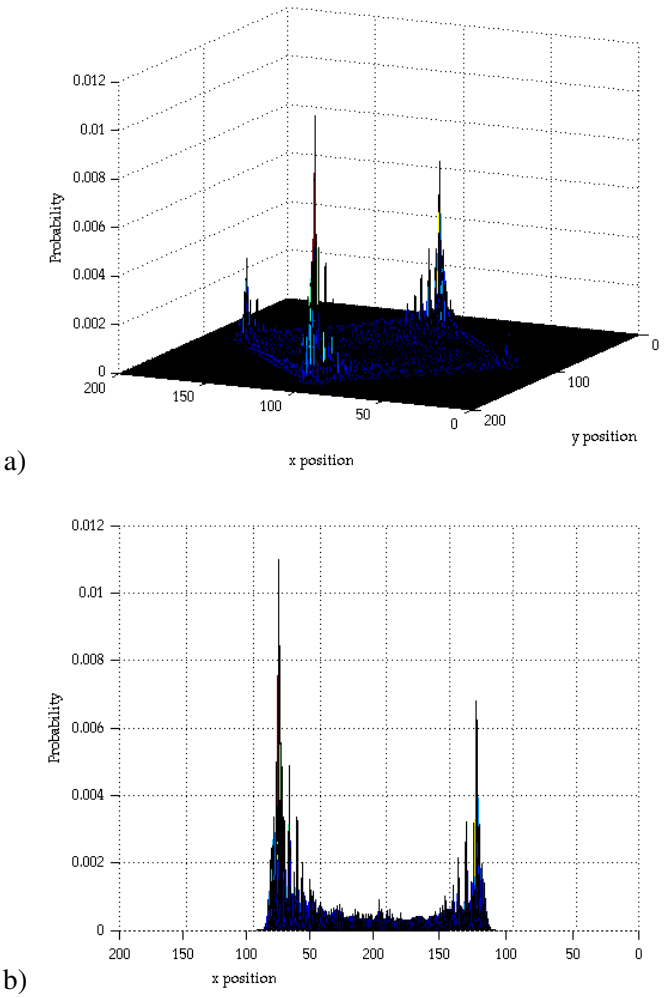


Figure 6.2: Probability distributions arising after 100 steps of a typical non-reversal quantum walk shown a) over the entire lattice and b) in profile 6.3

6.2.2 Properties

An example of a typical probability distribution arising from a non-reversal quantum walk is shown in Figure 6.2. The distribution is shown in 3d and in profile in order to show the similarity of the profile to that of the walk on the line. Regardless of initial condition, the dynamics are similar, tracing out a square with peaks at each corner. The initial condition determines the height and number of distinctive peaks at the corners of the square. The initial condition used to obtain the distribution shown in Figure 6.2 is the same as that used in equation

6.3.

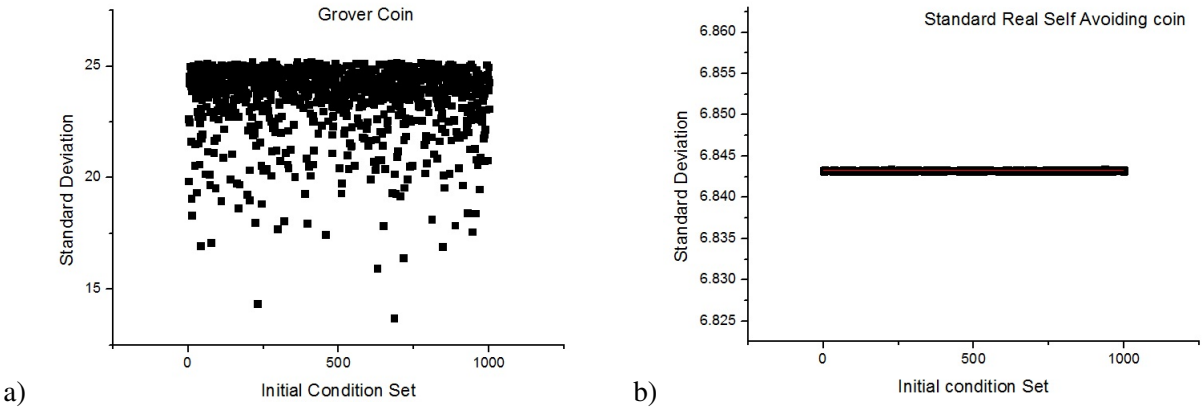


Figure 6.3: Comparison of standard deviation after 100 steps of the walk for various initial conditions using a) the Grover coin and b) a self-avoiding coin

As the analytical result concerning the independence of moments on initial condition indicated above applies only to walks using the operator U^{NR}, the non-reversal walks were investigated numerically. I conjecture that the same result holds of the non-reversal walk. These walks were investigated by varying the parameters in the coin as well as the initial condition. Random choices for each variable were used to generate 500 coins, and the walks arising from these operators were investigated using 1000 initial conditions. It was found numerically that for all choices of initial condition the mean position (Equation 6.1) and standard deviation (Equation 6.2) of the walker is constant at a given time t, as can be seen for the standard deviation in Figure 6.3. Further investigations suggest that all the moments of the distribution are independent of the initial condition. The parameters which determine the standard deviation are λ and γ, so these values determine some real constant k for each self-avoiding coin so that the standard deviation at time t is equal to kt. This contrasts strongly with standardly used coins, as mentioned in Section 6.1.2 above, where the initial condition can be used to control the standard deviation of the walk using the Grover coin operator. The mean and standard deviations as a function of time are shown for a variety of coins in Figure 6.4. Whilst the mean position increases faster than all other coins, the standard deviation increases more slowly, so in cases where we have freedom to choose our initial condition, the Grover operator is still the best to use if we want to achieve the fastest possible spreading of the amplitude across the lattice.

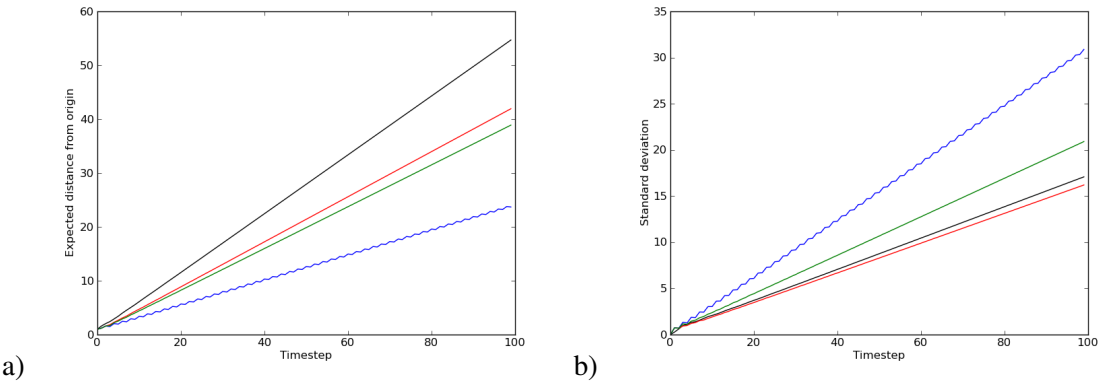


Figure 6.4: Comparison of the non-reversal (black), Hadamard (red), Grover (blue) and DFT (green) coin operators in terms of a) expected radial distance from the origin and b) standard deviation.

The analytics concerning the non-repeating coin deal with the joint moments, and numerical investigations of both coins showed that the individual x and y moments do depend on the initial condition. These were not investigated further as it appears that for any practical application of the model, the fully specified position of the walker will be of more interest.

6.3 Trimers

A self-avoiding walker of length three, which models a trimer, requires a more complex definition. This is because the trimer on a square lattice, presuming that the head and tail are distinguishable, can be in one of three configurations, as shown in Figure 6.5. These configurations can then have four possible orientations, facing in the up, down, left and right directions. This means that in order to represent the trimer, a qutrit tensored with a ququat is required, giving rise to a twelve dimensional coin. The head of the trimer can make the same moves as that of the dimer, but now as well as updating the position of the walker, the configuration, which will in general be in a superposition, is also updated.

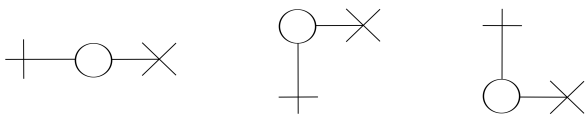


Figure 6.5: The three possible configurations of a self avoiding trimer

It may be possible to define a fully general self-avoiding quantum walk, however this will in practise be difficult. Even deciding when a site has been visited has some ambiguity, for instance, can one part of the superposition access sites which another part of the superposition has previously accessed?

6.4 Conclusion

I have introduced a new type of coin for the discrete time quantum walk and described analytically, for one choice of shift operator, and numerically for the other, that it has some notable properties, namely that the mean and standard deviation of the position of the walker is independent of the choice of initial condition. The standard deviation grows linearly with t for much the same reason as it does the walk on the line, as the coin operator always ensures that some amplitude moves away from the origin with each step.

A fully self avoiding quantum walk is yet to be developed and explored, and thus far no continuous time analogues have been proposed. In order to fully capture the folding of polymer chains, the quantum nature of their constituents may need to be accounted for, which would provide a potential application for such a model.

In addition to further investigating the properties of the self avoiding quantum walk and developing the concept, better comparisons with its classical counterpart may be useful. For example, only walks on square lattices have been considered here, but the self avoiding random walk has been shown to have macroscopic properties which are independent of the choice of lattice. In order to see if this property carries over into the quantum case, coins of varying dimension are required.

In short, there are many potential avenues through which to explore the self avoiding or non-reversal quantum walk, and the choices of which direction to take will depend on the desired applications of the walk. The purpose of the work here was simply to find out whether using a non-reversal coin operator gives rise to any properties which are not observed using other operators, hopefully provoking further investigation into the uses of such a coin. In the proceeding chapters, I turn to specific applications of various types of quantum walk.

Chapter 7

Quantum Finite Automata and Quantum Walks

7.1 Introduction

Whilst perfect state transfer is an important component in quantum computing, as it is potentially required for physically moving qubits around the computer, and in algorithms which rely on it to gain their speedup over classical algorithms [8; 27; 28], further ingredients are required [65; 99]. The required ingredients depend on the model of computation which we are using. For example, in Chapter 3 we saw that in order to perform Turing universal computation in the circuit model that, as well as wires, gates capable of, in combination, simulating any unitary operation to arbitrary precision are required. We also saw another model of quantum computation which is not universal, and which does not have perfect state transfer as a key ingredient in its construction, namely the QFA [72; 73; 74]. This model answers questions of the type: Is the input of the required form? Traditionally models of computation of this type are abstractions of an input tape, which is read by a tape head. The tape head can evolve into different computational states according to some transition function. These two models, circuits and QFAs, appear to be quite different. For instance the input is necessarily manipulated by the circuit model, and this generates the output, whereas an FSA does not manipulate its input and the output is determined by the state of the tape head. However, in their most general forms, in which case the FSA has extra structure added and can manipulate the input, becoming a Turing machine, the apparently different models are computationally equivalent. Computational equivalence entails that each model can simulate the other, and hence gives researchers

freedom to choose from a variety of computational models when approaching new questions.

There are a wide variety of mathematical models, all of which are Turing complete. For example, in addition to the Turing machine and circuit models, we have the cellular automaton [100; 101; 102; 103], the untyped lambda calculus [104; 105] and partial recursive functions [106; 107], all being equivalent. In the quantum case, the analogues of Turing machines [60; 108], cellular automata [109; 110] and circuits [76; 111] have also been shown to be equivalent, hence we have a variety of universal models to work with. This means that non universal restrictions of these models can also be explored in terms of each other as well if required. As we know that discrete time quantum walks are universal for quantum computation [41], we know that they can perform any computing task, regardless of the computing paradigm that task comes from. For example, the discrete time walk on the line has been shown in [112] to be equivalent to a one dimensional quantum cellular automaton [6; 109; 110; 113]. As these had already been proven to simulate any quantum Turing machine [109] it is implied that the discrete time walk on the line may also be able to simulate any quantum Turing machine. For this to be the case the update rules for the cellular automaton to which the quantum walk is mapped to must be of the form required for the second mapping onto the Turing machine. Whether this is the case is unclear from the constructions used to prove the equivalences and no subsequent research has examined this question. So whilst it has been known for a long time that there may be ways to construct quantum walks which accept languages, the work in this chapter is the first concrete example known to me.

In this chapter, we look at the relation between quantum walks and a model of quantum computing based on QFAs. There are many ways in which to modify QFAs, for examples see [73; 78; 114; 115; 116]. Unlike the case of their classical counterparts, these modifications do increase their computing power. Beyond accepting more languages than the standard QFA, the exact power of the different variations of QFAs is not generally known. Some progress, such as finding closure properties of languages accepted, has been made in this direction [72; 117; 118]. The specific modifications considered in this chapter involve adding open system dynamics to the QFA. We will also see in Section 7.2.2 that even much simpler modifications, such as allowing the tape head to move in both directions along the input tape, affect the language acceptance properties of some varieties of QFA.

The chapter proceeds as follows: In Section 7.2 I review the 'Measure-Many'-QFA, and discuss its language acceptance properties in order to demonstrate how adding an auxiliary system to the QFA increases its computational power. Then in Section 7.3 I introduce a much

newer modification of the QFA developed by Yakarylmaz *et al* in [119] which incorporates a memory, despite the transition function never depending on the contents of the memory, hence the term 'write-only memory.' To show how the write-only memory is useful, I provide an original algorithm showing a how this type of memory enables languages which cannot be accepted by standard QFAs. The final section shows that this modification of the QFA is equivalent to a quantum walk which uses a time dependent coin. In Section 7.5.1 I then show that by restricting the input alphabet to a unary alphabet, a time independent coin can be used, but in this case the number of steps required to accept any given language vastly increases.

7.2 'Measure-Many'- QFAs

An MM-QFA is a 6-tuple:

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

with Q being a finite set of states, and $\Sigma \cup \{ \, ¢, \$ \, \}$ being the tape alphabet where ¢ is the symbol directly before the input word and \$ denotes the end of the input word. The state vector evolves according to the unitary operator U_σ induced by the transition function $\delta : Q \times \Gamma \times Q \to \mathbb{C}$. The evolution operator is the same as for MO-QFAs:

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

The value of the transition function can be calculated if one knows the exact form of the operator induced by δ:

$$\delta(q, \sigma, q') = \langle q' \,|\, U_\sigma \,|\, q \rangle \tag{7.1}$$

for a 3-tuple (q, σ, q') with $a \in \Gamma$ and $q, q' \in Q$ the value of δ is the *amplitude* of the transition from configuration (q, σ) to state q. The probability of a given transition taking place is calculated in the normal way by squaring the absolute value of the amplitude of that transition.

The Hilbert space with basis set Q is partitioned by three subsets into three orthogonal subspaces: Q_{acc} is the basis for the subspace spanned by accepting state vectors; Q_{rej} is the basis for the subspace spanned by rejecting state vectors and $Q_{non} = Q \setminus (Q_{acc} \cup Q_{rej})$ is the

basis for the subspace spanned by all non-halting state vectors, so $Q = Q_{acc} \oplus Q_{rej} \oplus Q_{non}$. We also require, for fixed n, product spaces $C_{acc} = Q_{acc} \times \mathbb{Z}_n$, $C_{rej} = Q_{rej} \times \mathbb{Z}_n$ and $C_{non} = Q_{non} \times \mathbb{Z}_n$. After each transition the projection operator is applied to the product of some natural number less than n, the length of the input word, and the computational state. The computation halts when the system is measured to be in C_{acc} or C_{rej}. By operating on the product space rather than the Hilbert space itself, unwanted disturbance of the state vector of M is avoided.

During a computation an MM-QFA can accumulate accepting and rejecting probabilities before it halts. It can be useful to keep track of these by representing the computational state as a triple $v = (|\psi\rangle, p_{acc}, p_{rej})$ which evolves according to an operator T_σ, upon reading symbol σ

$$T_\sigma : (\psi, p_{acc}, p_{ref}) \rightarrow (P_{non} U_\sigma |\psi\rangle,\ p_{acc} + |P_{acc} U_\sigma |\psi\rangle|^2,\ p_{rej} + |P_{rej} U_\sigma |\psi\rangle|^2) \tag{7.2}$$

Once amplitude is transferred by the T_σ operator into a halting state it is simply a real number rather than a coefficient of a vector. Hence it no longer interacts with any non-halting states.

MM-QFAs are generalizations of MO-QFAs, hence the class of languages accepted by MO-QFAs is contained in the class of languages accepted by MM-QFAs. MO-QFAs simply have trivial projections between each computational step, those projections being identity matrices over the whole Hilbert Space. The additional subspaces and projections effectively increase the number of states the automaton can enter, and this increases its computational power.

7.2.1 Languages accepted by MM-QFAs

It was proven by Yakaryilmaz and Cem Say in [120] that the class of languages accepted with unbounded error by MM-QFAs with cut-point $1/2$ is precisely the stochastic languages. The proof is very long and technical, so is omitted here. If attention is restricted to acceptance with bounded error, then any advantage over classical FSAs is lost. This is because it has been shown by Kondacs and Watrous in [74] that MM-QFAs accept only regular languages, and there exists a regular language not accepted by MM-QFAs with bounded error. The language in their theorem is accepted with bounded error by the quantum walks discussed in this chapter and Chapter 8, so I include the proof that it is not accepted by MM-QFAs in order to give some insight into how they operate and indicate why, in a standard quantum information setting, this particular language can be problematic.

Theorem 7.2.1.1. *The regular language, L denoted by the regular expression $\{a,b\}^*a$ is not accepted with bounded error by an MM-QFA.*

Proof. The proof shows that given an MM-QFA, M, which accepts L, it cannot do so with bounded error. Let $x = \sigma_1 \dots \sigma_n \in \Gamma^*$, with Γ being the tape alphabet, so that

$$\psi_x = (P_{non}V_{a_n}) \dots (P_{non}V_{a_1})|q_0\rangle.$$

Define $\mu = inf\{\,|\psi_{¢w}\,||w \in \{a,b\}^*\}$, thus μ is the smallest probability that M has not yet halted, and the T operator for M will have placed the rest of the amplitude into halting states. If $\mu = 0$ then some computations of M will not depend on the symbol following w. However $wa \in L$ but $wb \notin L$, so in this case M cannot recognise L with bounded error.

Before considering the case where $\mu > 0$, note that for some set $A \subset B$ where $B = \{\, v \in V \,||v| \leq 1\}$, if there exists some $\xi > 0$ such that the norm of the distance between any $v, v' \in A$, $|v - v'|$ is greater than ξ then A has a finite number of elements. A must be finite in order to represent the configuration of an MM-QFA. One can also derive a constant c, such that $|T_x v - T_x v'| \leq c|v - v'|$.

Take w such that $|\psi_{¢w}| < \mu + \xi$, in which case $|\psi_{¢wy}| \in [\mu, \mu + \xi)$ for any $y \in \{a,b\}^*$. Therefore one can take any b^j with

$$|(P_{non}V_b)^j \psi_{¢wa}| \in [\mu, \mu + \xi)\,.$$

Thus the bounded sequence $\{(P_{non}V_b)^j\psi_{¢wa}\}$ has a limit point. There will also be another integer $k \geq 1$ such that $|(P_{non}V_b)^j(\psi_{¢wa} - (P_{non}V_b)^k\psi_{¢wa})| < \xi$. Combining this with the bounds established above for $|(P_{non}V_b)^j\psi_{¢a}|$ one can derive a constant, c' independent of the limit such that when $j = 0$, $|\psi_{¢wa} - (P_{non}V_b)^k\psi_{¢wa}| < c'\xi^{1/4}$. Hence there is another constant

$$|T_{¢wa\$}(|q_0\rangle, 0, 0) - T_{¢wab^k\$}(|q_0\rangle, 0, 0)| < c''\xi^{1/4}.$$

As ξ can be made arbitrarily small, the probability of accepting a word in L is arbitrarily close to that of accepting a word not in L, for example wab^k. In other words, M cannot accept L with bounded error.

□

7.2.2 2-way MM-QFAs

MM-QFAs can be modified to accept all regular languages with bounded error, and some non-regular languages (specifically the context free language $\mathcal{L}_{eq} = \{a^m b^m \,|\, m \in \mathbb{N}\}$), in a simple way. Namely by allowing the tape head to move in both directions down the tape. These machines are known as 2QFAs. In this case, the transition function must be modified to encompass

the probability of moving left, right, or not at all along the input tape. This makes the problem of ensuring that the transition function is well formed, in that it induces a unitary transition matrix, more complex. However necessary and sufficient conditions which the function must satisfy in order to guarantee being well formed were found by Kondacs and Watrous in [74]:

- $\sum_{q'} = \delta^*(q_1, \sigma, q', d)\delta(q_2, \sigma, q', d) = \begin{cases} 1 & \text{if } q_1 = q_2 \\ 0 & \text{otherwise} \end{cases}$
- $\sum_{q'}(\delta^*(q_1, \sigma_1, q', 1)\delta(q_2, \sigma_2, q', 0) + \delta^*(q_1, \sigma_1, q', 0)\delta(q_2, \sigma_2, q', -1)) = 0$
- $\sum_{q'} = \delta^*(q_1, \sigma_1, q', 1)\delta(q_2, \sigma_2, q', -1) = 0$

These conditions intuitively ensure logical reversibility, and that the rows and columns of the induced transition matrix are orthogonal.

Rather than allowing the tape head to move both ways down the tape, computational power can be gained simply by allowing the MM-QFA to pause on a tape square, becoming a PQFA. I proved that the language $\mathcal{L}_{eq}$ could be accepted by these automata in my MSc dissertation, by employing 'counting' states which were designed to only reach the last square of the input at the same time, thus allowing the QFT to transform them into the accepting state, if the number of a's was equal to the number of b's. The same technique was also used to show that there exists a PQFA accepting the context-sensitive language $\mathcal{L}_{abc} = \{a^m b^m c^m \,|\, m \in \mathbb{N}\}$. Clearly these automata require a large number of computational states, in the next section we see that the addition of 'write-only memory' enables QFAs to accept $\mathcal{L}_{eq}$ and $\mathcal{L}_{abc}$ with a limited number of computational states.

7.3 QFAs with write only memory

A Quantum Finite Automaton with Write Only Memory, $\mathcal{M}$, a QFA-WOM hereafter, is defined, in its simplest form, to be a 6-tuple:

$$(Q, \Sigma, \Gamma, \delta, q_0, F)$$

with Q being a finite set of states, and $\Sigma \cup \{\, \text{¢}, \$ \,\}$ being the tape alphabet as for MM-QFAs. The WOM-tape alphabet is Γ, which also includes the *empty string*, #, and the *empty symbol*, ϵ. The initial state of $\mathcal{M}$ is q_0 and $F \subset Q$ is the set of *accepting states*. During each computational step the tape head of $\mathcal{M}$ is restricted to move only right, but the WOM tape head is allowed to

move left, l, right, r, or remain stationary, s where the set $\{l, r, s\}$ is denoted D. The states Q form a basis for the computational Hilbert space, $\mathcal{H}_{\mathrm{Q}}$.

The possible states of the WOM tape are ordered pairs from the set $\Gamma^* \times \mathbb{N}_m$ where Γ^* is the Kleene closure of Γ. The set $\mathbb{N}_m$ is the set of natural numbers up to $m = 2k$, where k is the length of the input word plus two for the end markers. The length of the input word limits how far the WOM tape head can move from its initial position, as it is only allowed to move one square during each computational step. As the tape head can move in any direction, its current position must be indexed by $n \in \mathbb{N}$ in order to give the complete state of the WOM. A basis for the Hilbert space of the WOM, $\mathcal{H}_{\mathcal{W}}$, can be formed from the set $\Gamma^* \times \mathbb{N}_m$. As the computation is guaranteed to halt, because the QFA tape head always moves right and the computation ends when $\mathcal{M}$ reads the \$ symbol, $\mathcal{H}_W$ will be finite dimensional. The dimensions of $\mathcal{H}_W$ accessed by the QFA-WOM will depend on the precise instantiation of and input to $\mathcal{M}$ and has upper bound $2m|\Gamma|$, because during each step any $\gamma \in \Gamma$ can be written to the WOM tape and then the head can move in one of two directions or stays still. The space $\mathcal{H}_W$ can be interpreted physically as an environment which the QFA can interact with in a precisely controlled manner.

In [119] they introduce variations of WOM. For instance, one can allow the tape head to move in one direction only, in which case we have a QFA with a *push-only stack*, or a QFA-POS. In this case the dimension of the Hilbert Space will be equal to the length of the input string multiplied by $|\Gamma|$. One can simplify the model further by using an increment only counter rather than a tape, to make a QFA-IOC. Please note that the QFA-WOM and the variations introduced here are a simplification of the QFA-WOM introduced in [119] which also incorporates a finite register.

The transition function, δ when $\mathcal{M}$ is in state q reading symbol σ is specified by

$$\delta(q, \sigma) = \sum_{(q,\sigma,d)\in Q\times\Sigma\times D} (q, \sigma, q', \gamma, d)(q', \gamma, d).$$

where $(q, \sigma, q', \gamma, d) = a$ for $a \in \mathbb{C}$. This gives the amplitude of the transition from state q as the QFA tape head reads the symbol σ to q' whilst writing symbol γ to the WOM tape and moving the WOM tape head in direction d. To ensure that the evolution is unitary it must be the case that

$$\sum_{(q,\sigma,d)\in G\times\Sigma\times D} |\delta(q, \sigma, q', \gamma, d)|^2 = 1.$$

One can interpret δ as changing the state of $\mathcal{M}$ to q', writing $\gamma \in \Gamma$ to the WOM tape and moving the WOM tape head in direction d. After δ has been applied the QFA tape head moves one square to the right. In order for $\mathcal{M}$ to be *well-formed* the transition matrices, U_σ, induced by δ for each symbol of Σ, must be unitary. The entries of U_σ are specified from values of δ, but it is easier to consider the action of this operator as two separate operations. One, U_{Q_σ}, which operates only on the QFA states and the other U_{W_σ} controls the evolution of the WOM. The operator U_{W_σ} enables the QFA to control the WOM-tape, as the state of the QFA is updated before the WOM tape is written to, we have $U_\sigma = U_{W_\sigma} U_{Q_\sigma}$. The configuration of the QFA-WOM is its internal state, the position of the input tape head and the contents and position of the WOM tape head. If a configuration c_j can be reached in one step from c_i whilst $\mathcal{M}$ writes γ onto the WOM tape and moves in direction d then the (i, j)'th entry of U_σ is nonzero.

As the QFA and the WOM-tape are initially uncorrelated, we can assume without loss of generality that the initial state of the WOM tape is $|\#_k\rangle$, that is the empty string in the k'th tape square. Therefore the initial state of $\mathcal{M}$ is $|\psi_{int}\rangle = |q_i\rangle \otimes |\#_k\rangle$. After the operator U_σ is applied, the state of $\mathcal{M}$ is a vector in the Hilbert space $\mathcal{H}_\mathcal{Q} \otimes \mathcal{H}_\mathcal{W}$ of the form $|\psi_{QW}\rangle = \sum_{j,p} a_{j,p} |q_j, w_p\rangle$ where $a_{j,p} = \alpha_j \beta_p$ where α_j and β_p are the coefficients of the vectors $|q_j\rangle \in \mathcal{H}_Q$ and $|w_p\rangle \in \mathcal{H}_W$ respectively.

After the state transition from the \$ symbol had been made, the state of the QFA must be measured. The measurement is applied to reduced density operator $\rho_Q = Tr_W(\rho_{QW})$ where Tr_W is the partial trace over the WOM subsystem. $\mathcal{M}$ is said to have made an *accepting computation* if the state of $\mathcal{M}$ is measured to be in a state $q \in F$ at the end of its computation. Whilst the term 'write-only memory' may imply that we cannot measure or deduce the state of the WOM, this is not the case. In fact, as we will see in Section 7.4 below, knowing what state the WOM will be in, and therefore when interference between computational states can occur, is important in designing algorithms for the QFA-WOM. The term refers to the memory from the perspective of the QFA-WOMs transition function, which can write to the memory, but whose value never depends on the state of the WOM.

A schematic representation of a possible first transition of a QFA-WOM which accepts the language $L_{rev} = \{wcw^r | w \in \{a, b\}^*\}$, [119], omitting the part using the register, can be seen in Figure 7.1. This QFA-WOM operates in a very similar way to the one in the example given in Section 7.4 below, with a suitably increased input and WOM-tape alphabet.

QFA-WOMs and MM-QFAs could appear to be two sides of the same coin, as they involve two different methods of describing environmental interactions. However, due to the fact that

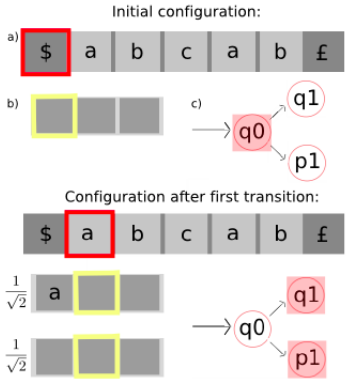


Figure 7.1: Schematic diagram of a QFA-WOM transition. Part a) shows the QFA-tape, where the input word is read, the red square shows the tape head position. b) shows the initially empty WOM-tape. After this transition it splits into an equal superposition of the tape head having written 'a' and moved one square right, and having moved without writing a symbol. c) shows part of the state diagram, the arrow going into state $|q_0\rangle$ indicates that this is the initial QFA state; arrows going from it are possible transitions. In this case the state goes to $\frac{1}{\sqrt{2}}(|q_1\rangle + |p_1\rangle)$.

the measurements performed during the computation of an MM-QFA are on space $\mathcal{H}_Q \otimes \mathbb{Z}_l$, not the computational basis space, this is not the case. Additionally, the entire evolution of the QFA-WOM is unitary, whereas due to the projections, every computational step of an MM-QFA may be non unitary. Only those steps where the projection operator is the identity will give rise to unitary evolution. A direct comparison between MM-QFAs and QFA-WOMs would shed light on what type of open system dynamics may be most beneficial in a quantum computer, but due to the subtleties in the models such a comparison would be difficult.

7.4 Example QFA-WOMs

To complete the introduction to QFA-WOMs, I now show that QFA-WOMs are more powerful than standard QFAs, by proving they accept the context-free language $\mathcal{L}_{eq}$ with bounded error. Whilst the conclusion that WOM adds computational power is not original, as it is shown in [119], the algorithm described here is, though it is based on the same techniques used in [119]. Additionally, the formulation of QFA-WOMs used here is not quite identical to the original definition, as mentioned above, that also includes a finite register, so we prove that not all of the structure added to the QFA in [119] is required in order to gain computational power.

Theorem 7.4.0.1. *There exists a QFA-WOM which accepts the language $\mathcal{L}_{eq}$ with bounded error.*

Proof. Call the QFA-WOM we are considering $\mathcal{M}$ and let $Q = \{q_0, q_1, q_2, q_3, q_4, q_5\}$, $F = \{q_0\}$ and $\Gamma = \{a\}$. Define δ in the following way:

$$\delta(q_0, ¢) = \frac{1}{\sqrt{2}}(q_0, \epsilon, 0) + \frac{1}{\sqrt{2}}(q_1, \epsilon, 0)$$

$$\delta(q_0, a) = (q_0, a, +1) \qquad \delta(q_1, a) = (q_1, \epsilon, 0)$$
$$\delta(q_0, b) = (q_3, \epsilon, 0) \qquad \delta(q_1, b) = (q_2, a, +1)$$
$$\delta(q_3, b) = (q_3, \epsilon, 0) \qquad \delta(q_2, b) = (q_2, a, +1)$$
$$\delta(q_3, a) = (q_4, \epsilon, 0) \qquad \delta(q_2, a) = (q_5, \epsilon, 0)$$
$$\delta(q_4, a) = \delta(q_4, b) = (q_4, \epsilon, 0) \quad \delta(q_5, a) = \delta(q_5, b) = (q_5, \epsilon, 0)$$
$$\delta(q_3, \$) = \tfrac{1}{\sqrt{2}}(q_0, \epsilon, 0) + \tfrac{1}{\sqrt{2}}(q_3, \epsilon, 0) \quad \delta(q_2, \$) = \tfrac{1}{\sqrt{2}}(q_0, \epsilon, 0) - \tfrac{1}{\sqrt{2}}(q_3, \epsilon, 0)$$

To show the result, first consider inputs of the required form. Upon reading the beginning marker symbol the state splits into a superposition. The q_0 state counts the number of $a's$ appearing before the first b in the input by writing each occurrence of an a to the WOM tape. The q_1 state does nothing until it reaches a b. Upon reaching the first b the q_0 state transitions into state q_3 and writes nothing further to the WOM tape. The q_1 state transitions into state q_2 and writes an a to the WOM tape. For the rest of the input, q_2 writes another a each time, and does not make a state transition. Therefore upon reading the end marker symbol, both parts of the superposition will have the same WOM tape contents, a string of a's which is half the length of the input, and hence constructive interference can occur, resulting in the computational state being q_0 with certainty, as required.

For inputs not of the form $a^m b^n$, the two parts of the superposition will have different associated WOM-tape contents, so cannot constructively interfere. Hence the input will be accepted with probability $1/2$. In the case where we have inputs where an a occurs after the first b the QFA transitions into rejecting states, and the probability of accepting is zero. □

Corollary 7.4.0.1. *There exist QFA-WOMs accepting languages of the form* $\mathcal{L} = \{a^m b^m c^m ... | m \in \mathbb{N}\}$.

Proof. The construction above has states which indicate when transitions to new symbols have been made, which use the WOM to count the number of new symbols and enter into rejecting states if symbols which should not be occur at a given position of the input word are read. Simply augment the set of computational states to allow for equivalent such states for each additional symbol in the language. □

7.5 Quantum walks as QFA-WOMs

I will now show that the QFA-WOM as formulated above is simply a discrete time quantum walk and show one of the algorithms from [119] as quantum walk in order to elucidate the mapping. First, I specify what conditions, in general, must be fulfilled in order to map a QFA onto a quantum walk:

1. An encoding of the computational states, which will automatically encode q_i and q_f.

2. An encoding of the input.

3. An operation which replicates the final probability distribution of states in $\mathcal{H}_C \otimes \mathcal{H}_V$ as is produced in $\mathcal{H}_Q$, the space whose basis is the set of computational states of the QFA, for a specific input.

To model the QFA-WOM, in addition to the above conditions we must find:

1. An encoding of the WOM-tape.

2. An operation which replicates the final probability distribution of states in $\mathcal{H}_C \otimes \mathcal{H}_V$ as is produced in $\mathcal{H}_Q \otimes \mathcal{H}_W$ for a specific input.

If we have a time dependent coin, then the mapping is direct. We can identify $\mathcal{H}_Q$ with $\mathcal{H}_C$ and $\mathcal{H}_W$ with $\mathcal{H}_V$, so the computational states are the coin states, and the WOM state is the position state. In this case, the encoding of the input is in the sequence of coin operators applied. Acceptance of the input is determined by measurement of the final coin state. This formulation of the QFA-WOM helps to elucidate the role of the WOM, controlling when interaction between computational states can occur by only allowing them to interact when they are both at the same position of the graph structure which represents the possible configurations of the WOM and the relations between them. Whilst the examples of the operation of the WOM given here are all based on an interferometer, as there are more ways for different parts of the superposition of the quantum walker to end up at the same node of a graph, it is possible that this formulation will lead to more sophisticated algorithms for the QFA-WOM.

A more concrete example of how this mapping works is shown schematically in Figure 7.2 for a walk accepting the language $\mathcal{L}_{rev} = \{ wcw^r \,|\, w \in \{a, b\}\}$ where w^r denotes the word w reversed. For clarity, only edges and coin states required at each step of the computation are shown. In this example, an accepting computation is simply a quantum walk between antipodal nodes of a cycle. The walker traverses each side simultaneously. If the input is in the language accepted, then the walkers end up at the same node. If the input is not in the language accepted, then the walk will not take place on a cycle. Instead, it is simply a walk on the line, with the two parts of the superposition moving deterministically further away from each other at each step.

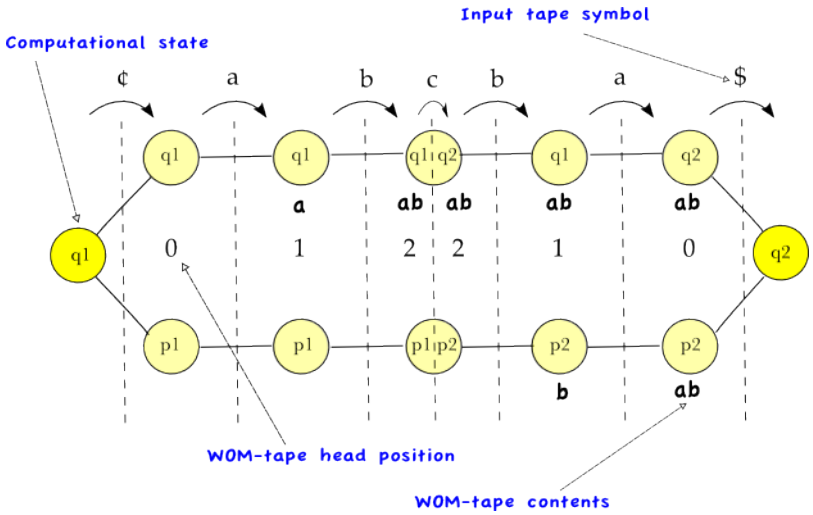


Figure 7.2: The computation of a QFA-WOM accepting $\mathcal{L}_{rev}$ on word *abcba*

The graph structure induced by the WOM will be, in the most general case, multiply connected. The number of connections between each node will be equal to $|Q|$. In practise, some of the coin states at each node will never contain amplitude so they can be omitted. The overall structure of the induced graphs is restricted by its interpretation as possible configurations of symbols on a tape. We cannot, for instance, jump from a vertex which represents the initial configuration of the WOM to one representing a string of length greater than one. As every time the coin state splits into a superposition, the WOM-tape splits too, the graph will generally have a branching structure, with branches meeting when separate parts of the QFA superposition have the same WOM-tape contents and head position. If the coin state evolves without updating the WOM-tape, then the graph will have self loops. As we are identifying the computational states with coin states, a QFA would just be a single node with many self loops.

7.5.1 Walks with a time independent coin

The mapping between the QFA-WOM and the quantum walk requires a time dependent coin in order to account for the input. A time independent coin can be used if we restrict inputs to a unary alphabet. In this case, the input simply determines the number of steps that the walk is run for, though the beginning and end marker symbols must now be incorporated into the coin operation or the graph structure. As crucial state transitions are made from both end marker symbols, they cannot be omitted even though we know how many steps to run the walk for, and therefore when the end of the input word has been reached. In the example below, the end

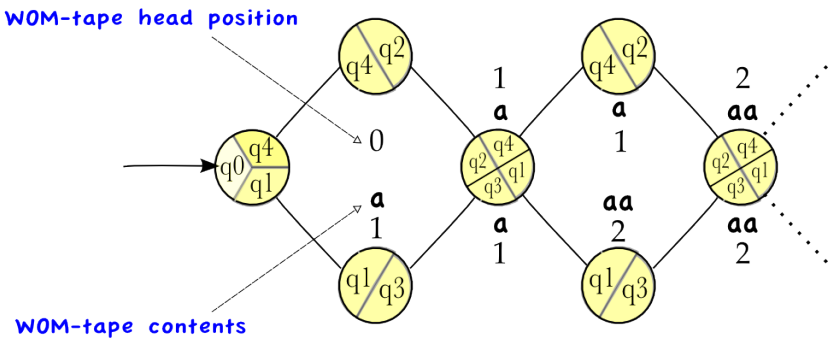


Figure 7.3: The quantum walk and QFA-WOM accepting the unary version of the language $\mathcal{L}_{even}$

marker symbols are not required so this problem is not tackled. The way to specify the coin and graph structure must be independent of the length of input otherwise the problems may be solved by this specification, rather than the quantum walk itself.

Moving to a unary language may seem to heavily restrict the types of problems a computer can solve, however this is not necessarily the case as there are simple algorithms to convert between binary and unary representations of strings [121]. The choice of the number of symbols affects the complexity of subsequent computations rather than their inherent computability. It is not always the case that languages expressed in a unary alphabet take longer to recognise than their binary representations, but if the model of computation being used to evaluate strings for membership in the language is a one way model, and strings in the unary representation are longer than their binary counterparts, this will be the case. In general, unary languages are more often used to derive complexity results rather than computability results for specific computational paradigms [122; 123].

A simple example of a unary language is $\mathcal{L}_{even} = \{ 0^k \,|\, \mathrm{k} \text{ is even} \}$. In binary this language is represented by the regular expression $\{a, b\}^* a$, which, as we saw in Theorem 7.2.1.1, cannot be accepted with unbounded error by an MM-QFA.

Theorem 7.5.1.1. *There exists a QFA-WOM accepting* $\mathcal{L}_{even}$

Proof. Take the QFA-WOM with $Q = \{q_0, q_1, q_2, q_3, q_4\}$, q_0 being, as usual, the initial state and $F = \{q_0, q_2, q_3\}$. The transition function is specified as follows:

$$\delta(q_0) = \tfrac{1}{\sqrt{2}}(q_4, \epsilon, 0) + \tfrac{1}{\sqrt{2}}(q_1, a, +1)$$
$$\delta(q_4) = (q_2, a, +1) \qquad \delta(q_1) = (q_3, \epsilon, 0)$$
$$\delta(q_2) = (q_4, \epsilon, 0) \qquad \delta(q_3) = (q_1, a, +1)$$

As the input alphabet is unary, the input symbol argument has been removed from δ. As the states q_2 and q_3 are obtained after the first two applications of δ, and every subsequent second application of δ, the language is clearly accepted with certainty. The states q_2 and q_3 cannot be reached after an odd number of steps, so inputs of odd length will be rejected with certainty. □

Whilst it is not necessary for the proof that both accepting computational states occur at the same position, this reduces the number of nodes required in the graph structure. The quantum walk version of this QFA-WOM is a diamond chain. All of the coins but the first are permutation operators. A schematic diagram of the walk, along with its translation into the QFA-WOM is shown in Figure 7.3.

7.6 Conclusion

We have seen how the addition of different types of heavily controlled environments increase the computational power of QFAs. In the case of MM-QFAs, projections are applied meaning that the evolution is no longer purely unitary. The advantage of these projection operators could be clearly seen in the use of the T operator (see equation 7.2), as portions of the amplitude are effectively disregarded in the computation. Accumulated accepting and rejecting probabilities cannot be reintroduced into the non-halting subspaces, so different properties of the input can be tested simultaneously without interference. However, whilst MM-QFAs are more computationally powerful than QFAs, their one way incarnations are still very limited in the types of problems they can solve, even if we allow for unbounded errors. They are, however, potentially more physically relevant, as environmental interaction, which is in practise very difficult to avoid, is built into the model.

The evolution of the QFA-WOM is purely quantum, though the QFA itself enters into a mixed state. QFA-WOMs have been shown to can accept, with certainty, context-free and context-sensitive languages. The advantage of the WOM is that it enables control over when

interference between computational states can occur. In the examples both here and in [119] this feature is only used in the last step of the computation so the full capabilities of this type of memory may not yet have been realised. As they are a new model of quantum computation, there are many open questions regarding the QFA-WOM. For example, thus far all the QFA-WOMs studied only allow the input tape head to move in one direction along the tape. As this modification increases the power of other QFAs, it may increase the power of the QFA-WOM. Determining precisely which languages are accepted with bounded and unbounded error will enable better comparison with other quantum computers. Establishing lower bounds for the number of states, and the size of the WOM alphabet would indicate whether the WOM gives any advantage beyond the ability to accept more languages.

By noting the equivalence between QFA-WOMs and quantum walks, I have found alternative frameworks for either to be investigated in and elucidated the role of the WOM. This provides a new way of viewing quantum walks as quantum computers, by expressing them in terms of the language acceptance model of computation rather than the circuit model. By restricting attention to unary languages, a quantum walk can simulate a QFA-WOM with a time independent coin. In the following chapter, I will give examples of quantum walks solving language acceptance problems with a time independent coin and no such restrictions on the size of the alphabet.

The motivations for examining both variations of QFAs and quantum walks is not purely intellectual. Existing experimental implementations of quantum computers are far more similar to QFAs than to quantum Turing machines, so by studying them we can learn more about which problems it is currently practically feasible to solve with a quantum computer. Equally, discrete time quantum walks have been experimentally realised in a variety of ways [124; 125], some of which are capable of performing over 100 timesteps [124]. So far, these are typically walks on the line, or on a two dimensional lattice. However as a high degree of control over the coin, and protection from decoherence, can be obtained there is no reason not to be confident that walks over more complex structures are feasible.

Chapter 8

Language recognition

8.1 Introduction

In the previous chapter we saw that quantum walks can solve language acceptance problems via their equivalence to the QFA-WOM. In this case, we must either use a time dependent coin or a unary alphabet. Due to the fact that the discrete time quantum walk using a time independent coin is a computational primitive, we know that it must be possible to use it to solve arbitrary language acceptance problems. In this chapter I show two ways of using the quantum walk to solve precisely these sorts of problems. This provides a new way in which to view quantum walks as performing quantum computations. As mentioned in the previous chapter, the fact that quantum walks have been experimentally realised to a large number of steps [125] motivates the investigation of quantum walks as quantum computers.

Long before the proofs that the continuous and discrete time quantum walks are universal for quantum computation, it was known that they had algorithmic applications. For example, the quantum walk search algorithm has been extensively studied [11; 12; 41], and the faster hitting times have been exploited [26; 28; 126]. The algorithmic applications have been used to prove complexity bounds, for example, the time taken to evaluate an AND-OR formula [127]. The purpose of the work presented in this chapter is to extend the applications of the discrete time quantum walk to novel computational problems. As mentioned in Chapter 7 it is possible that there is an indirect mapping from quantum Turing machines to quantum walks via the quantum cellular automaton, but this has not been investigated. Also one can potentially use the known equivalence between the quantum Turing machine and the quantum circuit model to apply the results from [23] to language acceptance problems, finding a direct mapping pro-

vides a more intuitive understanding of this application of quantum walks. Furthermore, using either possible route to equivalence with the Turing machine would require clarification and extensions of the proofs. For example, the proof of equivalence between the quantum Turing machine and the circuit model from [76] applies to circuit families, and makes no reference to the specific unitaries used for the gates. Therefore to use this equivalence the proof would need to be reformulated in terms of specific unitaries, and the notion of the circuit family would need to be formulated in terms of the quantum walk model.

This chapter introduces two ways in which quantum walks can solve language acceptance problems by constructing walks which accept specific languages. These walks are shown numerically to have the required properties, and in two cases alternative walks which admit simple analytic proofs are given. First, in Section 8.2 the requirements which a walk would have to fulfil for it to be considered to accept a language are stated. Then in Section 8.3 the first method, which processes the entire input simultaneously, and the techniques used to analyse all following walks, are described. The walks induced using this type of method are shown to accept languages from different places in the Chomsky hierarchy. These walks are particularly suited to accepting languages with at most one word of each length, but they are also shown to accept languages with more than one word of each length in Section 8.3.1. Instead of processing each input symbol simultaneously, it is possible to process each symbol in turn, and this case is dealt with in Section 8.4. These walks can be used to accept specific words using only permutation operators for the coin, so the graph structure itself is used to determine whether the input is of the required form. Entire languages are then considered, and I show by example that this can be more efficient than using the permutation scheme for words in the language.

8.2 Requirements of language acceptors

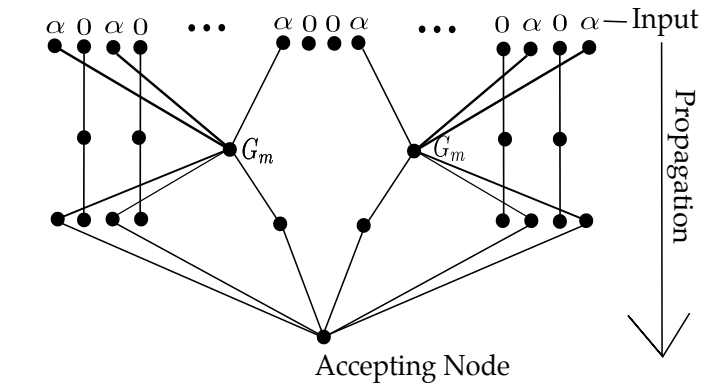
In order for anything to be considered to perform a computation, an input and an output are required, as well as a way to manipulate the input in order to generate the output. As we are specifically looking to apply the quantum walk to language recognition problems, the input must represent a string, and the output must allow us to determine whether or not the string was in the language we are recognising. Whilst there may be some freedom in choosing the encoding of the input, it is difficult to see how the output can be determined by anything but a measurement of the state of the walker. Once the input and output have been formulated a graph structure and coin operations which generates an appropriate output from the input

must be found. There are a variety of ways that this can be done and in this chapter I use the state of the walker itself as an input, starting in a superposition. First I distribute the input along different nodes so that the entire word is operated on simultaneously. Then I show that the input symbols can be fed into the graph structure one after another. These ways of initialising the walk could potentially be implemented using a multiport interferometer, modulated to enable differentiation between a and b inputs. Most of the work presented in this chapter deals with languages from a binary alphabet, however the models developed are not restricted to using only two input symbols, and using larger alphabets enables me to indicate how the walks can accept context-sensitive languages.

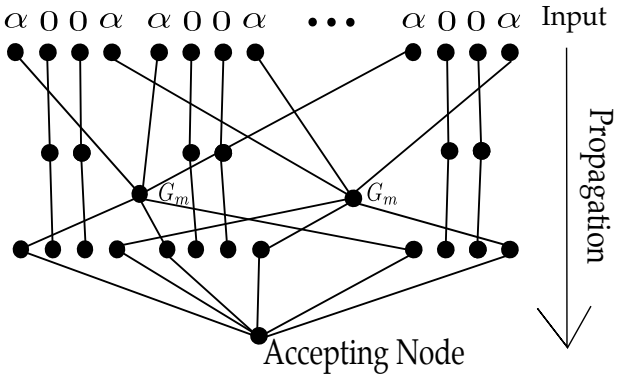
The simplest way to deal with the output appears to be to designate an 'accepting node,' which all accepting amplitude should be directed to. This also offers potential practical advantages, as determining acceptance then equates to measuring the position of the walker. The problem then is to find graph structures and sets of coin operations which, upon a string from the language the walk is designed to accept, transports a high proportion of amplitude to the accepting node. If the remaining amplitude can be redirected to another, rejecting, node then the accepting and rejecting conditions can be inverted to allow the same walk to accept both its language and its complement- the set of words not in that language. Choosing these methods of input and acceptance relates the language acceptance problem to perfect state transfer problems, see, for example [29; 30; 84].

8.3 Spatially distributed input

In this type of walk the input will initially be distributed along different nodes of the graph which are all operated on simultaneously. There are two nodes for each input symbol with alternate nodes representing a or b. The length n of the input must be known. For each input symbol σ at position j in the input string, the node $2j$ is populated with amplitude $1/\sqrt{n}$ if σ is an a and $2j+1$ contains amplitude if it's a b. The structure of the graph and coin for a walk testing whether a word is in a given language must be specifiable in terms of n only. This ensures that small computational overheads are required to initialise the walk. Representing the input like this is reminiscent of the techniques used in quantum fingerprinting [128], as the size of the HIlbert space used to represent a word from a binary alphabet is $2n$ where n is the



a)



b)

Figure 8.1: Graphs required to accept a) $\mathcal{L}_{eq}$ and b) $\mathcal{L}_{ab}$ with each input symbol being operated on simultaneously. Grover coins of the appropriate dimensions are used at each vertex. Amplitude encodes a word which will be accepted by the walk. Edges join only at nodes indicated by black circles

length of the word, rather than 2^n. Quantum fingerprinting is used to determine whether two strings are equal using their 'fingerprints' alone. The fingerprints are exponentially smaller than the original words, but this gain results in a small probability of error when determining if strings are equal. It is possible that the probability of error in determining whether a word is in the language accepted by a given quantum walk occurs in a manner analogous to the case of quantum fingerprinting, and is hence a result of representing the input in a Hilbert sapce which is exponentially smaller than the Hilbert space of all possible inputs.

Once the input is intialised the walk is then run for a specified number of steps. When the initial state encodes a word in the language accepted by the walk, the amplitude should be

directed to the designated 'accepting node.' If a word not in the language accepted is encoded in the initial state then less of the amplitude will be directed towards the accepting node, so we are looking for acceptance with bounded error. The modulus squared of the final amplitude at the accepting node yields the probability of acceptance.

Acceptance of the empty set, empty string, and singleton symbols is trivial. For the empty set and string, specify a graph with a single, accepting, node. For the singleton symbols, simply connect only that symbol to the accepting node and use permutation operators for the coin. Less trivially, these walks are most easily used to swiftly accept languages which contain at most one word of each length, so the graph tests for that specific word. Examples of such languages are the context free language $\mathcal{L}_{eq} = \{a^m b^m | m \in \mathbb{N}\}$ and the regular language $\mathcal{L}_{ab} = \{(ab)^m | m \in \mathbb{N}\} = \{ab\}^*$ and graphs accepting them can be seen in Figure 8.1. The design of these graphs was informed by the previous work in this thesis concerning perfect state transfer, particularly using the Grover operator.

The input is accepted with certainty if it is in the language accepted by the walk. The probability of accepting words not in the language depends precisely on the string, but will for strings of length $n > 1$ be less than or equal to $2/n^2$. Acceptance with certainty can also be achieved without using the Grover operator, by simply connecting the input nodes where there should be amplitude if the input is in the language accepted by the walk to the accepting node. However, in this case there is no way for amplitude to be transmitted away from the accepting node if the input is not in the language accepted by the walk, hence the probability of accepting these inputs is higher. For a word of the form $a^m b^{m-1} a$ the probability of accepting will be $1 - \frac{1}{\sqrt{n}}$, which is in general higher than the probability of accepting words not in the language in the walk using the Grover operator.

In both of these walks the input is processed in three full steps of the walk, regardless of length. However the gains in time complexity are at the expense of spatial complexity. For inputs of length n the walks accepting $\mathcal{L}_{eq}$ and $\mathcal{L}_{ab}$ require $4n+3$ nodes. The graph from Figure 8.1 a) can be extended to accept the archetypal context-sensitive language $\mathcal{L} = \{a^m b^m c^m | m \in \mathbb{N}\}$ (see Chapter 9 for further details) and b) to accept $\mathcal{L} = \{(abc)^m | m \in \mathbb{N}\} = \{abc\}^*$. In this case we must extend the model to deal with more than two input symbols which will involve a corresponding increase in the number of nodes required.

The properties of the walks over the graphs indicated in Figure 8.1 were investigated by simulating them in Python for all possible inputs up to a given length. For both languages the results were very similar so I limit the discussion to $\mathcal{L}_{eq}$ here.

For the analysis of the walk accepting $\mathcal{L}_{eq}$ and all subsequent walks the Jaro distance [129; 130] between the input and the closest word to a word in the language under consideration for that length was also calculated. The Jaro distance of strings w_1 and w_2:

$$d_j = \begin{cases} 0 & \text{if } m = 0 \\ \frac{1}{3}\left(\frac{m}{|w_1|} + \frac{m}{|w_2|} + \frac{m-t}{m}\right) & \text{otherwise} \end{cases} \tag{8.1}$$

with m being the number of matching characters, characters which occur in both strings, in the same order, within a certain distance determined by the length of the strings. The value of t, the number of transpositions, is obtained by dividing the number of characters which differ by sequence order by 2. The three parts to the equation calculate the ratios of the number of matching characters to the lengths of w_1 and w_2 and then the ratio of non-transpositions to matching characters.

The Jaro distance was selected as it always has values between 0 and 1, with 1 indicating that two words are equal, hence it was easy to compare to the probability of acceptance. In the case of even length inputs, the Jaro distance between that input and the word from $\mathcal{L}_{eq}$ of that length was calculated. For odd inputs with length n the word was compared to the word in $\mathcal{L}_{eq}$ of length $n-1$. The results for the first 200 strings are illustrated in Figure 8.2 (a). The points at which both curves peak are at the position of the words *ab*, *aabb*, *aaabbb*. As the languages considered do not have words of each possible length, in order to test inputs of any possible length a distance metric which can compare strings of different lengths was required, hence the use of the Jaro distance, rather than, say, the Hamming distance.

The disparity between the Jaro distance and the probabilities of acceptance for words not in $\mathcal{L}_{eq}$ arises from the design of the algorithm, which requires a low probability of acceptance for any word not in the language, regardless of how close that word is to a word in $\mathcal{L}_{eq}$. Hence the probability of acceptance cannot be used as a good measure of how close the input word is to one in in that language in cases where it is not equal to 1.

The constructions used here are based on the fact that a d dimensional Grover operator:

$$G_d = \begin{pmatrix} \frac{2-d}{d} & \frac{2}{d} & \cdots & \frac{2}{d} \\ \frac{2}{d} & \frac{2-d}{d} & \cdots & \frac{2}{d} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{2}{d} & \frac{2}{d} & \cdots & \frac{2-d}{d} \end{pmatrix} \tag{8.2}$$

with d even, transmits all amplitude to the 'leaving' coin states of a vertex if the amplitude is initially evenly (with respect to both magnitude and phase) distributed between $n/2$ 'entering' coin states. In other words:

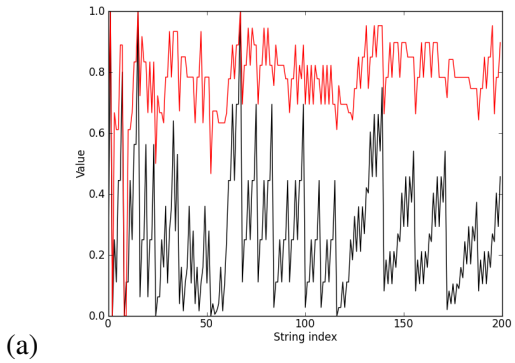
$$G_d \begin{pmatrix} \alpha \\ \alpha \\ \vdots \\ \alpha \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ \alpha \\ \vdots \\ \alpha \\ \alpha \end{pmatrix} \tag{8.3}$$

The deterministic evolution that the Grover operator can produce was used to design the 'wires' which transmitted the amplitude between gates in [23]. This evolution can also be exploited to generate walks accepting with certainty other languages such as $\mathcal{L}_{twin} = \{ww|w \in \{a,b\}^*\}$ and $\mathcal{L}_{rev} = \{ww^r|w \in \{a,b\}^*\}$ where w^r denotes the symbols of w in reverse order.

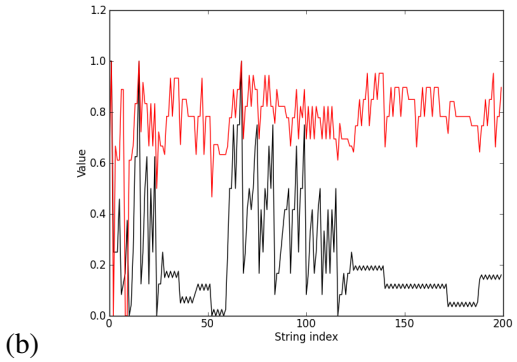
It is simple to prove that the discrete time quantum walk accepts the languages $\mathcal{L}_{eq}$ and $\mathcal{L}_{ab}$ with certainty if we use a slightly different graph, which requires an increase in the number of nodes required by the graph accepting the walk. The graphs are shown in Figure 8.3. For the walk accepting $\mathcal{L}_{eq}$ the probability of acceptance for the first 200 strings using both graph structures is plotted in Figure 8.2. Whilst both walks shown to accept $\mathcal{L}_{eq}$ peak at the index indicating a string in the language, and go to zero for words sharing no symbols with the word in $\mathcal{L}_{eq}$ of that length, their behaviour on intermediate words is quite different. This is due to the fact that on the walk over the graph depicted in Figure 8.2 (b), if a single coin state is populated at one of the four dimensional nodes, which will not happen for inputs in $\mathcal{L}_{eq}$, half of the amplitude is always directed back towards the input nodes. In the case of the walk over the graph shown in Figure 8.1 the proportion of amplitude directed back to the input nodes depends on how many coin states are populated, which is determined by the number of symbols the input is wrong by.

Proposition 8.3.0.1. *The language $\mathcal{L}_{eq}$ is accepted with certainty by the graph and choice of operators shown in Figure 8.3 (a). Words not in the language are accepted with bounded error.*

Proof. Words in $\mathcal{L}_{eq}$: The result follows by induction on m. For the base step, simple calculation and Equation 8.3 shows that ab will be accepted with certainty. For the induction step,



(a)

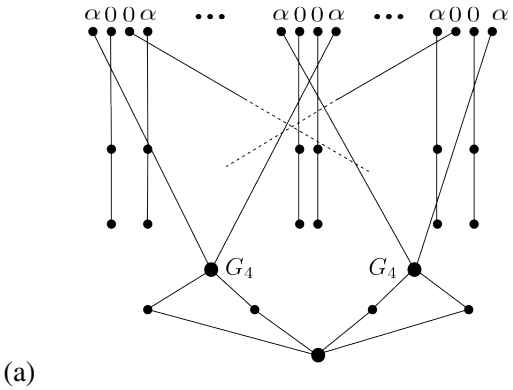


(b)

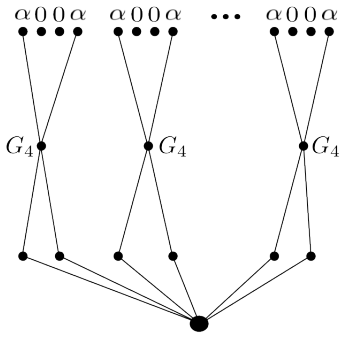
Figure 8.2: Probability of acceptance for (a) first walk detecting word $\mathcal{L}_{eq}$ for the first 200 strings (black) and (b) the second walk accepting the same language. The Jaro distance between the input word and an appropriately sized word from $\mathcal{L}_{eq}$ is indicated in red.

suppose that a word $a^m b^m$ is accepted with certainty, and consider the word $a^{m+1}b^{m+1}$. Due to the construction of the graph, every pair $a_i b_{i+m}$ where $i \leq m$ is treated independently, so by the induction hypothesis all amplitude from the first m a's and b's is transmitted to the accepting node. The calculation to show that all amplitude from the subsequent pair of a's and b's is identical to the base step.

Words not in $\mathcal{L}_{eq}$: As there is no path between the input nodes representing symbols which do not occur in words from $\mathcal{L}_{eq}$, their amplitudes cannot contribute to the accepting probability. To prove the acceptance is with bounded error, consider a word which differs from a word in $\mathcal{L}_{eq}$ by one symbol, for example $a^m b^{m-1} a$. This will be accepted with the maximum possible probability for a word not in $\mathcal{L}_{eq}$. The amplitude from the final a cannot be transmitted to the accepting node, so the word cannot be accepted with probability $p_{acc} > 1 - \frac{1}{2m}$. Additionally, after step one of the walk the amplitude from the m'th a goes to the Grover operator, and now



(a)



(b)

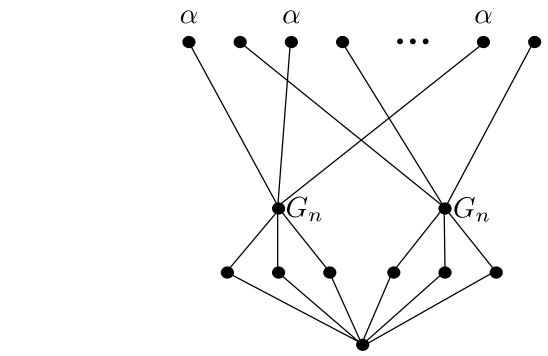
Figure 8.3: Graphs accepting (a) $\mathcal{L}_{eq}$ and (b) $\mathcal{L}_{ab}$ used in proof of Proposition 8.3.0.1 and Corollary 8.3.0.1

some of this amplitude will be transmitted back to the input node:

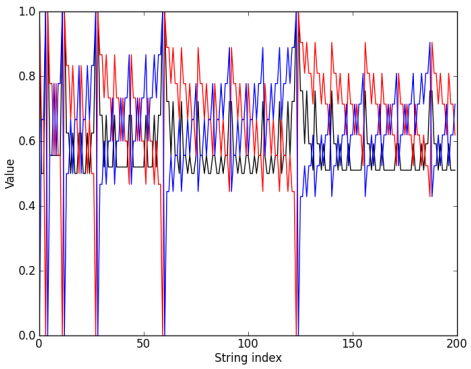
$$\begin{pmatrix} -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2m}} \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} -\frac{1}{2} \\ \frac{1}{2\sqrt{2m}} \\ \frac{1}{2\sqrt{2m}} \\ \frac{1}{2\sqrt{2m}} \end{pmatrix} \tag{8.4}$$

Hence the total probability of accepting $a^m b^{m-1} a$ is $1 - \frac{1}{2m} - \frac{1}{4m}$. □

Corollary 8.3.0.1. *The language $\mathcal{L}_{ab}$ is accepted with certainty by the graph and choice of operators shown in Figure 8.2 (b).*



(a)



(b)

Figure 8.4: (a) Graph accepting $\mathcal{L}_{single}$ and (b) the probability of accepting the first 200 strings (black) along with the Jaro distance between each string and a string of the same length entirely composed of a's (red) and entirely composed of b's (blue)

8.3.1 Accepting languages which contain more than one string of each length

The walks outlined above take advantage of the fact that the languages they accept are highly structured, to the extent that it is only possible to have one word of a given length in these languages. Whilst other languages which contain more than one word of each length such as ($\mathcal{L}_{rev}$ and $\mathcal{L}_{twin}$) were mentioned, as their structure can be exploited in a similar way, it is also possible to use quantum walks to accept languages with far less structure. In this section I present two such examples, a walk accepting the language $\mathcal{L}_{single}$ denoted by the regular expression $\{a\}^* \cup \{b\}^*$, which has two words of each possible length, and a walk accepting $\mathcal{L}_{even} = \{a, b\}^* a$.

The walk accepting $\mathcal{L}_{single}$ exploits the Grover operator again, this time to ensure that if all the input symbols are a or all the input symbols are b then all input amplitude will be directed to the accepting node. The graph over which the walk takes place is shown in Figure 8.4 (a) and the probability of accepting each string is shown in part (b). The walk accepts words in the language with certainty. The acceptance is with bounded error. The strings not in $\mathcal{L}_{single}$ with the highest probability of being accepted by the walk contain $n-1$ a's and a single b, or vice-versa. The order of the symbols does not matter. In order to calculate the total probability of a word of this form being accepted, we just add the contributions from the a and b symbols. The contribution from the a symbols is $(n^2-2n+1)/n^3$ and from the b symbol we have $1/n^2$ so the total probability of accepting words of this form is $(1/n^2)(2+n^2-2n)$.

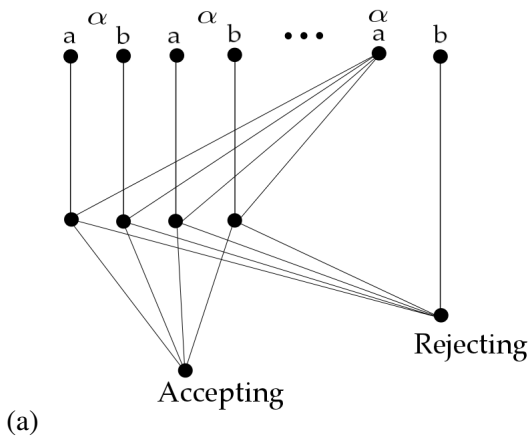
The graph over which the walk accepting $\mathcal{L}_{even}$ takes place is shown in Figure 8.5 (a) and the probability of accepting each string is shown in Figure 8.5 (b). This walk uses a biased Hadamard operator at the central nodes:

$$H^{bias} = \begin{pmatrix} \sqrt{\frac{1}{2n}} & \sqrt{1-\frac{1}{2n}} \\ \sqrt{1-\frac{1}{2n}} & -\sqrt{\frac{1}{2n}} \end{pmatrix} \tag{8.5}$$

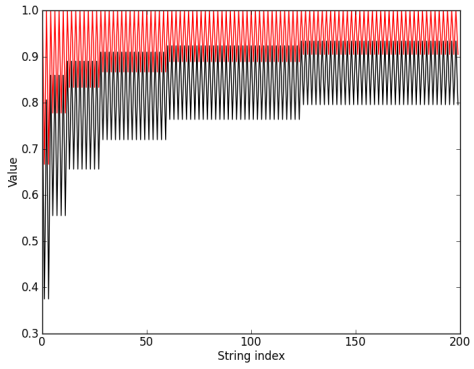
where n is the length of the input word. The acceptance is not with certainty, but it is with bounded error. Words of length n in the language are accepted with probability $(n-1)(\frac{1}{2n^2(n-1)}+\frac{1}{n}(1-\frac{1}{2n}))$, and words not in the language are accepted with probability $\frac{n-1}{n}(1-\frac{1}{2n})$.

Whilst the walks presented in this section both accept with bounded error, they generally have far greater probabilities of accepting words not in the language accepted by the walk than the examples presented in the previous section. The walk accepting $\mathcal{L}_{single}$ exploits the Grover operator to ensure that when the input is of the correct form, all amplitude is transmitted to the accepting node, but even when words are not in the language, this operator will transmit some of the amplitude to the accepting node. Unlike the previous cases where this operator is used, every node representing a possible input symbol is connected to the accepting node via nodes at which the Grover operator acts, so amplitude from any input node can influence the accepting probability.

In the case of the language $\mathcal{L}_{even}$ the high probability of accepting words not in the language occurs because the acceptance is determined by a single input symbol, the amplitude from which is used to control the proportion of amplitude from other symbols that is directed to the accepting node, but the coin operator always sends some amplitude to the accepting node regardless.



(a)



(b)

Figure 8.5: (a) Graph accepting $\mathcal{L}_{even}$, with the operator at the central nodes given by Equation 8.5 and (b) the probability of accepting the first 200 strings (black) along with the Jaro distance between each string and the closest string from $\mathcal{L}_{even}$ (red)

Using a spatially distributed input allows for long words to be accepted with the same number of operations as short words. However, the number of nodes required in the graph structure grows, albeit linearly in the examples shown, with the length of the word. The number of nodes required to accept a given language can be held constant regardless of input length if each input symbol is fed into the structure in turn, so we now turn to this case.

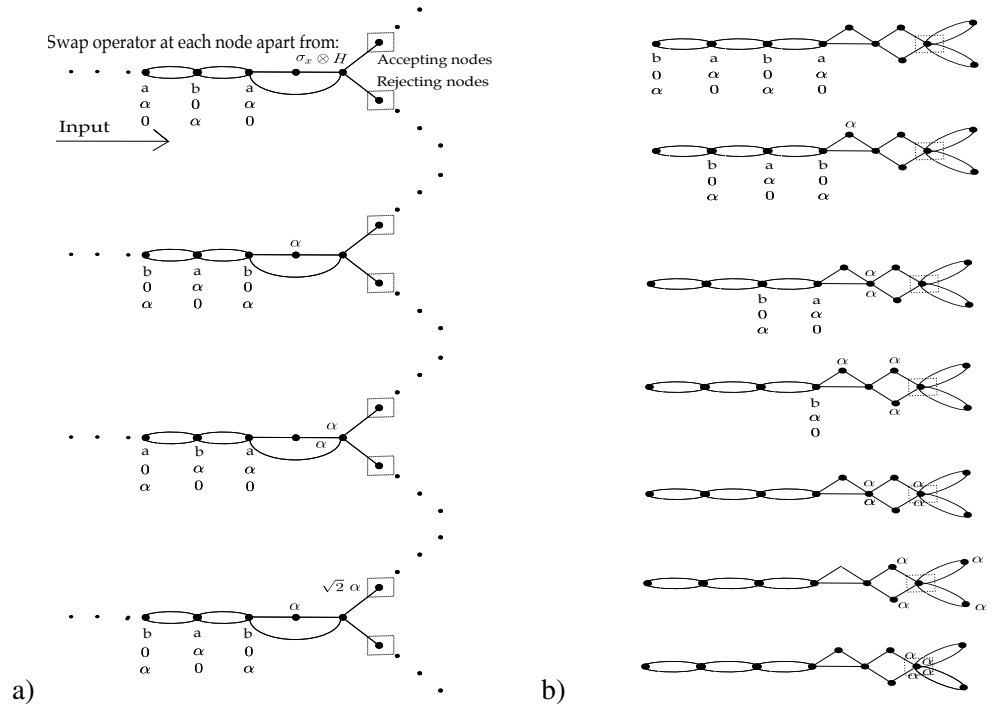


Figure 8.6: a) Graph structure and coins accepting language $\mathcal{L}_{ab}$ b) Graph specifically accepting the word of length 4 from that language

8.4 Sequentially distributed input

The input can be treated sequentially if we start with it along a chain with two links between each node. The two symbols are represented:

$$a = \begin{pmatrix} \alpha \\ 0 \\ 0 \\ 0 \end{pmatrix} \qquad b = \begin{pmatrix} 0 \\ \alpha \\ 0 \\ 0 \end{pmatrix} \tag{8.6}$$

The coin on this part of the graph is $\sigma_x \otimes \mathbb{I}_2$, which simply swaps amplitude between the 'leaving' nodes of the current node to the 'arriving' nodes of the next node. These are then fed into a graph which has either 'accepting paths' and 'rejecting paths' or 'accepting nodes' and 'rejecting nodes.' The total square of the modulus of the amplitude on the accepting node/path gives the probability of accepting the input. The shape of the graph and the coins at each node determine which words will be accepted.

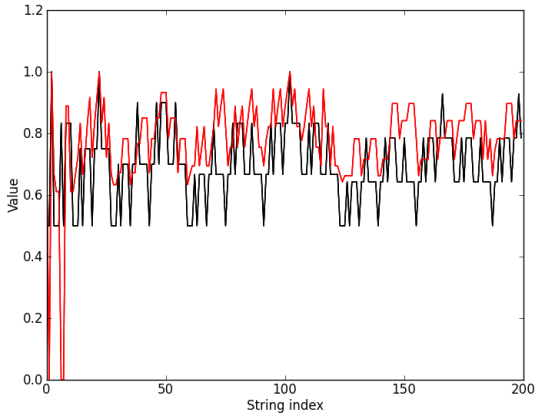


Figure 8.7: Probability of acceptance for walk detecting word from $\mathcal{L}_{ab}$ for the first 200 strings (black). The Jaro distance between the input word and an appropriately sized word from the language is indicated in red.

The empty set and empty string are accepted trivially, as we can distinguish between no walk occurring and all amplitude being rejected. Singleton symbols can be accepted by paths of length 3 with the amplitude initially in the appropriate coin state of the central node and swap operators between each node. In some cases, such as walks accepting specific strings of known length, the only coins required are trivial swap operators, as in Figure 8.6 (b). More complex languages require more complex graphs, such as that in Figure 8.6 (a), the graph accepting the language $\mathcal{L}_{ab}$. This graph uses the Hadamard operator:

$$H = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{pmatrix} \tag{8.7}$$

to determine whether each pair of symbols fed to it is of the form ab and moves the amplitude from both symbols into the accepting path if they are of that form. Words in the language are accepted with certainty and those not in the language are accepted with probability at least $1/2$. By adding m nodes rather than one to the path which amplitude from b's goes into, this walk can be modified to accept $\mathcal{L}_{eq}$.

Here we have another example of multiple graphs performing the same function, in this case accepting a specific word. Either a specific graph or the graph accepting $\mathcal{L}_{ab}$ can be used to accept $abab$, as shown in Figure 8.6 (a) and (b). In the case tested for comparing whether different graphs accepting the same word, all words of length four, give rise to different probability distributions, the probability of acceptance did not depend on the graph, however this

property is not generally expected to hold. Here we can see that exploiting quantum properties, using the Hadamard operator to control interference between different parts of the amplitude rather than swapping amplitude around so that it all arrives at the right place eventually, gains us efficiency. The simple swapping version of the graph accepting *abab* has 8 nodes (discounting the input nodes) and takes 6 steps to accept the word. Whereas the more general graph not only accepts more words, but accepts *abab* in 5 steps. To accept longer words from $\mathcal{L}_{ab}$ using a permutation scheme rather than the graph using the Hadamard operator, more cycles are required and the size of the graph increases accordingly.

I demonstrate the algorithms correctness enumeratively, but similarly to those in Section 8.4 this can be proven easily by induction on m, where m is the number of times the string *ab* is repeated. Again the probability of accepting each input string was computed and was plotted alongside the Jaro distance from the input word to a string of an appropriate length in the language as can be seen in Figure 8.7. The points at which both values go to unity indicate the positions of the word *ab*, *abab*, *ababab*.

8.5 Conclusion

In this chapter, two models of language recognising quantum walks have been discussed. Specific examples of such walks have been given, and shown numerically, and in some cases analytically, to have the required acceptance properties. We have seen that there can be more than one walk accepting a given language and that these have different effects on strings not in the language accepted by the walk. In order to specify walks accepting, for instance, an arbitrary regular expression, some way of choosing which walks to use to accept a given expression must be found. This would also be required for stating any definite complexity results. In order to compare the walks with other standard models of computation in terms of language acceptance, relations between the efficiency measures used here and other standard efficiency measures such as the minimal number of computational states, or tape squares traversed, must be found.

In addition to enabling comparison with other models of computation in terms of computability and complexity, I hope that further development of the work initiated in this chapter may shed light on formal languages in the context of quantum computation. In classical computation there is a well defined hierarchy of formal languages. As mentioned in Chapter 3 this hierarchy is called the Chomsky hierarchy. There are well understood mappings between the

languages of the hierarchy and the computational models which accept them [70]. Currently there is no quantum analogue of this hierarchy, and the fact that modifying QFAs can increase the number of languages they can accept suggests that the quantum case will not be so clear cut. The fact that very similar quantum walks can accept regular, context-free, and context-sensitive languages highlights this. I hope that if it is possible to come up with quantum walks accepting any formal language, comparison of the walks may clarify whether there is a quantum analogue to the Chomsky hierarchy and how the languages might be grouped together in this case.

The two approaches discussed in this chapter may not be the only ways to specify quantum walks such that they can be interpreted as accepting formal languages. It is not yet clear what the most fruitful approach will be. A better understanding of the relative merits of using spatially versus sequentially distributed inputs will be informative, and gaining insights into their limitations may suggest further ways of specifying walks to recognise languages. Whilst the examples in this chapter use the state of the walker to specify the input, there may be ways of using the graph structure or coin operator to encode the input. There is less freedom in the choice of the output, as the state of the walker is the only thing which would change during the walk, so this must be used to determine whether the input is accepted, even if it did not initially encode the input.

The choice of approach will depend on the intended applications of the walks. In the next chapter we see that this work can be applied to the problem of quantum state discrimination. In this case the spatially distributed input has a clear advantage: it is able to perfectly discriminate orthogonal states, whereas the example given for the time sequential input accepts every input with probability $p_{acc} \geq 1/2$ and hence cannot perform the task as well as a simple projective measurement.

Chapter 9

Quantum inputs and quantum state discrimination

9.1 Introduction

The models of computation we have seen so far take classical inputs. In the circuit model, inputs are either state $|0\rangle$ or $|1\rangle$, in the language acceptance model, the input is a specific word. As the initial state of a quantum walk can be any quantum state, the language recognition models from the previous chapter can be used to examine quantum inputs. Testing the walks for different types of inputs should increase our understanding of these computational models and may suggest ways to increase their power. Furthermore, this is the first instance known to me of a quantum computational model which can take quantum inputs, so exploring them may provide new insights into the field of quantum computation as a whole.

In this chapter I outline a preliminary investigation into the use of quantum inputs. First in Section 9.2 I give examples of the effects of quantum inputs on the acceptance properties of specific language accepting walks and note that the walks can then be interpreted as performing a type of state discrimination. Then in Section 9.3 I note that with a binary alphabet, it is not possible to have superpositions of more than two words. In Section 9.3.1 I give an example of a walk which accepts a language from an alphabet of three symbols, and can take inputs which are a superposition of three words. The results from these preliminary investigations suggest a link to quantum state discrimination, and in Section 9.4 I revisit the example from Sections 9.2 and 9.3 to compare the state discrimination performed by the walk to the standard state discrimination scenario.

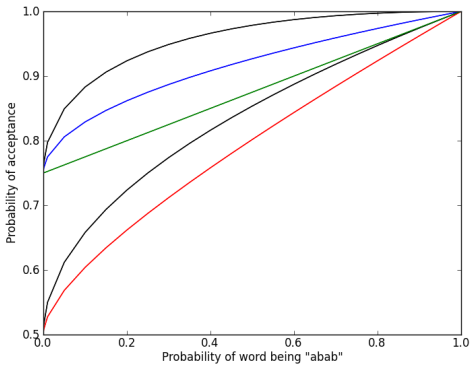


Figure 9.1: Probability of accepting state for walk detecting word from $\mathcal{L}_{ab}$ using a superposition of input word 'abab' with every other binary string of length 4 using the sequentially distributed input. The results for the words 'baaa' and 'bbba' are shown by the red curve; the lower black curve shows 'aaaa,' 'bbbb' and 'bbaa'; the green curve depicts 'baba'; blue shows 'aabb,' 'baab', 'abba','babb' and 'aaba' and the upper black curve shows 'abbb', 'abaa', 'aaab', and 'bbab.'

9.2 Superpositions of two words

The way the language accepting walks have been set up allows us to give them a quantum input. Each symbol in the word can be in a superposition of a or b, $xa+yb$ such that $|x|^2+|y|^2 = \alpha^2$. Superpositions of words, for example $abab$ and $bbbb$ can then be created by using the appropriate superposition for each symbol in the word. Where symbols match, the amplitude is allocated to that symbol as for the classical encoding. Where symbols do not match, the amplitude is distributed between the a and b states accordingly.

As a preliminary investigation into using quantum inputs I tested the effects of using a quantum input on the acceptance probability for the walks accepting $\mathcal{L}_{ab}$ using both the spatial and sequential input versions. I then compared the two versions of the walk accepting $\mathcal{L}_{eq}$ using the spatially distributed input. The results from spatially distributed inputs were very similar, so again I limit discussion to $\mathcal{L}_{eq}$ in this case. For the potential application of quantum inputs developed in this chapter it is preferable to use spatially distributed inputs, as we can see from Figure 9.1 showing results for the sequential input accepting $\mathcal{L}_{ab}$, all states are accepted with probability $p_{acc} \geq 1/2$, so focus the discussion on the case of the spatially distributed input. I tested each possible binary string of length four in superposition with the word of length four from the languages accepted by the walks. As the weighting of the word from the language accepted by the walk increases, the probability of acceptance increases.

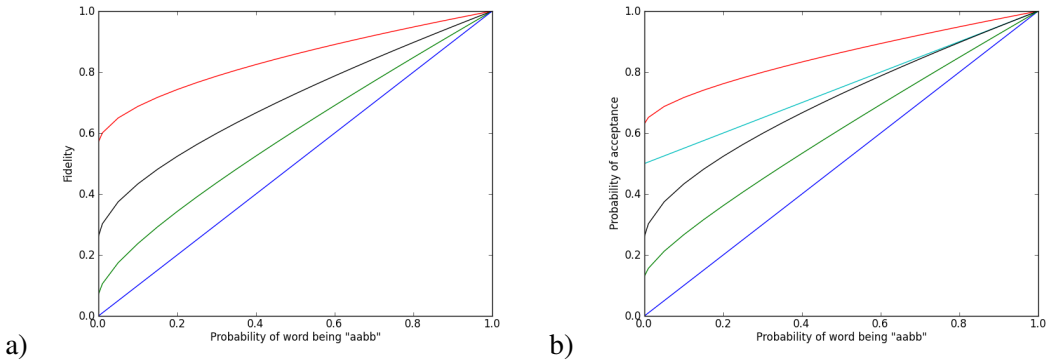


Figure 9.2: Probability of accepting quantum inputs for walk detecting words from $\mathcal{L}_{eq}$ using a superposition of input word 'aabb' and every other binary string of length four using a) the walk using an operator G_m and b) the walk using operator G_2

For all the examples tested, the fifteen possible superpositions fall into four or five distinct cases see Figures 9.2. Whether we get four or five cases, four cases always correspond to the number of matching symbols between the words in the superposition. For example, in Figure 9.2 (a), the blue curve represents the probability of accepting a superposition between 'aabb' and the 'orthogonal' word 'bbaa;' the red curve represents the probability of accepting superpositions of 'aabb' and 'abaa,' 'baaa,' 'bbba' or 'bbab' and so on for the other two curves. Where we get an additional curve, the ordering of the symbols affects the acceptance probability. In the walk accepting $\mathcal{L}_{eq}$ using the G_4 operators, if there is an 'a' at position i and a 'b' at position $i+m$ then all amplitude from these symbols will be directed to the accepting node, giving rise to the cyan curve of Figure 9.2 (b). In the other walk accepting this language, the positions of 'a's relative to 'b's does not affect the accepting probability so we get the four curves based on the number of matching symbols. The case for the sequential input walk accepting $\mathcal{L}_{ab}$ is more subtle, as both the ordering of words and the positioning of specific pairs in the work affects the acceptance probability. Words such as 'baba' are accepted with higher probability despite sharing no symbols with the word of length four from $\mathcal{L}_{ab}$, because the walk does not take into account the position in the word that a pair 'ab' occurs, so all the amplitude from this portion of the word becomes accepting amplitude.

Using a quantum input frames the question of language acceptance in terms of quantum state discrimination. If we have one of two states $|\psi\rangle$ and $|\phi\rangle$, one of which encodes a word in the language accepted by the walk, and the other does not, then we can gain information about which state we had by seeing whether or not it is accepted by the walk. Another setup for using

discrete time quantum walks to perform state discrimination by measuring at specific positions is outlined in [131].

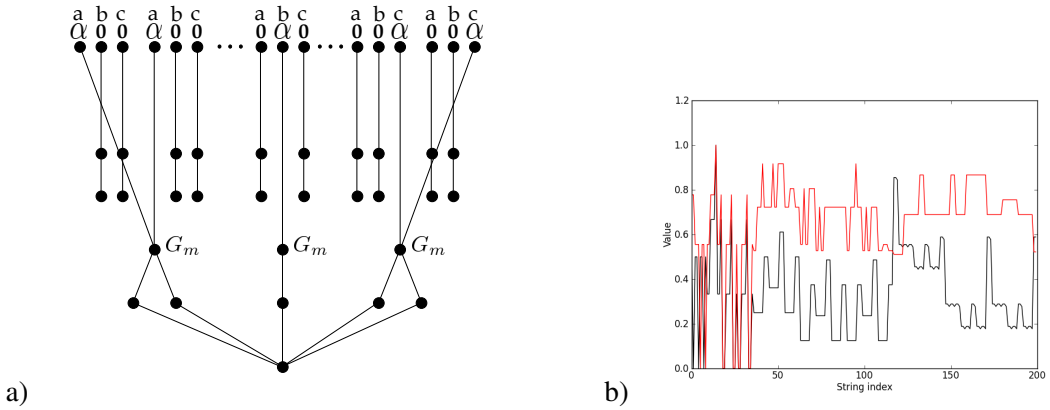
9.3 Superpositions of more than two words

In general, one may wish to distinguish between more than two states. In this case, a larger alphabet is required, as superpositions of more than two words are always equal to one of two words. This is because, when written in vector form, supepositions of three words do not give rise to three linearly independent vectors, they can always be rewritten as two linearly independent vectors. The words in the final superposition will have no overlap, and hence be orthogonal states when representing the input to walks, as long as for each position in the words, at least one word in the set considered has a different symbol to the rest. For example:

$$\begin{aligned} &\frac{1}{\sqrt{3}}|aabb\rangle + \frac{1}{\sqrt{3}}|bbaa\rangle + \frac{1}{\sqrt{3}}|abab\rangle = \\ &\begin{pmatrix} \sqrt{\frac{1+1}{3}}|a\rangle \\ \frac{1}{\sqrt{3}}|b\rangle \end{pmatrix} + \begin{pmatrix} \frac{1}{\sqrt{3}}|a\rangle \\ \sqrt{\frac{1+1}{3}}|b\rangle \end{pmatrix} + \begin{pmatrix} \sqrt{\frac{1+1}{3}}|a\rangle \\ \frac{1}{\sqrt{3}}|b\rangle \end{pmatrix} + \begin{pmatrix} \frac{1}{\sqrt{3}}|a\rangle \\ \sqrt{\frac{1+1}{3}}|b\rangle \end{pmatrix} = \\ &\sqrt{\frac{2}{3}}|abab\rangle + \frac{1}{\sqrt{3}}|baba\rangle \end{aligned} \tag{9.1}$$

The words in the rewritten superposition can be perfectly distinguished by the walk accepting $\mathcal{L}_{ab}$. When the words in the superposition do all share a symbol at a given position, it is still the case that the superposition is equal to one of two words, but in this case they will not be represented by orthogonal initial state vectors. Both parts of the superposition will have that symbol at that position, for instance:

$$\begin{aligned} &\frac{1}{\sqrt{3}}|aaaa\rangle + \frac{1}{\sqrt{3}}|bbaa\rangle + \frac{1}{\sqrt{3}}|abab\rangle = \\ &\begin{pmatrix} \sqrt{\frac{1+1}{3}}|a\rangle \\ \frac{1}{\sqrt{3}}|b\rangle \end{pmatrix} + \begin{pmatrix} \frac{1}{\sqrt{3}}|a\rangle \\ \sqrt{\frac{1+1}{3}}|b\rangle \end{pmatrix} + \begin{pmatrix} \sqrt{\frac{1+1+1}{3}}|a\rangle \\ 0|b\rangle \end{pmatrix} + \begin{pmatrix} \frac{1+1}{\sqrt{3}}|a\rangle \\ \sqrt{\frac{1}{3}}|b\rangle \end{pmatrix} = \\ &\sqrt{\frac{2}{3}}|abaa\rangle + \frac{1}{\sqrt{3}}|baab\rangle \end{aligned} \tag{9.2}$$



a) b)

Figure 9.3: a) The graph accepting $\mathcal{L}_{abc}$, which uses the Grover operator at each node and b) the probability of acceptance for the first 200 strings (red) along with the Jaro distance between each string and the string of the closest length from the language accepted

9.3.1 Expanding the alphabet

The walks developed in this thesis can be applied to state discrimination of sets of more than two states. From a computer science perspective, there is no need to increase the alphabet, as any language can be encoded in binary, but this will be required if we wish to develop a general state discrimination scheme. Here I show that it is possible to use walks accepting languages with a larger alphabet to discriminate between sets of more than two states. In particular I take a walk which can discriminate between three states and show it has the desired properties. The walk used accepts the context sensitive language $\mathcal{L}_{abc} = \{a^m b^m c^m | m \in \mathbb{N}\}$. The graph required to accept this walk is shown in Figure 9.3 a) and the probability of accepting for the first 200 strings of three symbols is shown in part b). Due to the fact that only strings of lengths which are multiples of three can be accepted, and the large number of combinations of three symbols, the only accepting peak shown on the graph is for the string 'abc'. It is possible to prove that this walk has the desired properties via induction on m, just as we saw for $\mathcal{L}_{eq}$ and $\mathcal{L}_{ab}$ in Chapter 8. As seen in Figure 9.4, which shows the probability of accepting superpositions of varying weights of the words 'abc,' 'cab' and 'ccc,' the properties of the walk in terms of state discrimination are similar to those discussed in Section 9.2. The 'orthogonal' word is discriminated perfectly, and the ability of the walk to discriminate between words depends on the number of symbols they share. Hence 'ccc,' sharing one symbol with 'abc,' is accepted with probability $1/3$. In contrast to the case of superpositions of two words, if the walk rejects the input, we cannot know with certainty whether we had 'ccc' or 'cab.'

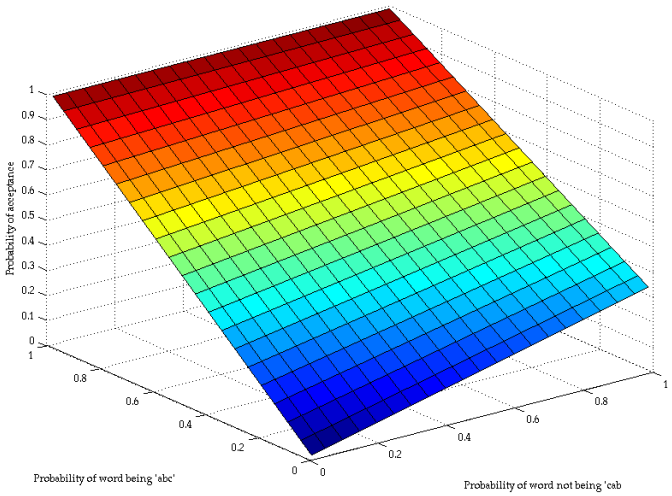


Figure 9.4: The probability of accepting superpositions of ‘abc,’ ‘cab’ and ‘ccc,’ the profile of the ‘abc’ axis gives the probability of accepting a superposition of just ‘abc’ and ‘cab’ and the profile of the ‘cab’ axis gives the probability of accepting superpositions of just ‘cab’ and ‘ccc.’

9.4 Comparison of language acceptance and quantum state discrimination

I now compare the outcomes of walks using quantum inputs to quantum state discrimination by calculating the probability of correctly identifying what input word we had with the theoretical bound for correctly identifying which quantum state we had in the case of two specific examples.

9.4.1 Distinguishing two states using the walk accepting $\mathcal{L}_{eq}$

Suppose that we are trying to distinguish whether a state encoded the word ‘aabb’ or ‘aaab.’ In the case of the $\mathcal{L}_{eq}$ example in Section 9.2 acceptance of a word means that it could have been ‘aabb,’ but it could also have been ‘aaab.’ If the word is rejected, then the word had to be ‘aaab’. Without running the walk multiple times to gain the exact probability of acceptance, in which case the states can be distinguished unambiguously, a firm answer to the question of which state we had cannot be obtained. It is possible, however, to calculate the probability that we had ‘aabb’ or ‘aaab.’ If we know the prior probabilities of each word occurring, then the

probability of the state having been ‘aaab’ can be calculated using Bayes theorem:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{9.3}$$

The probability of the walk accepting ‘aaab’ can be calculated directly, to simplify formulae I round it to 0.6 in the following calculations. It is also assumed that the prior probabilities are equal for ‘aabb’ and ‘aaab,’ but this does not have to be the case. In the case where the walk rejects the input, we have

$$P(`aaab'|reject) = \frac{P(reject|`aaab')P(`aaab')}{P(reject)} = \frac{0.4 \times 0.5}{0.2} = 1 \tag{9.4}$$

as required. The probability of the walk rejecting is simply $1 - P(accept)$ where $P(accept)$ is the total probability that the walk accepts, given equal prior probabilities of either word, this is $1/2 + 0.6/2 = 0.8$. In the case where the walk accepts, it is not possible to know exactly which word we had, but the probability that we had either word can easily be calculated:

$$P(`aaab'|accept) = \frac{0.6 \times 0.5}{0.8} = 0.375 \tag{9.5}$$

$$P(`aabb'|accept) = \frac{1 \times 0.5}{0.8} = 0.625 \tag{9.6}$$

Hence even though it is not possible to know with certainty which word we had, information has been gained. As the probability that the word was ‘aabb’ is greater than that of it being ‘aaab,’ we conclude that the word was ‘aabb’ in this case. As the probability that the word was accepted was 0.8 and the probability that the word was ‘aaab’ is 0.375, the total probability of making the wrong decision is 0.3. This is a much higher probability of error (P_E) than that established by the Helstrom bound [132]:

$$P_E \geq \frac{1 - (1 - |\langle\psi_2|\psi_1\rangle|^2)^{0.5}}{2} \tag{9.7}$$

which comes to 0.169... when ψ_1 and ψ_2 are the appropriate vector representations of the words ‘aabb’ and ‘aaab.’

There is also another type of quantum state discrimination called ‘error-free’ or unambiguous state discrimination. As the name suggests, in this case there is no probability of error, but it is possible to have an inconclusive result. In our example, using the walk, the probability of unambiguously distinguishing the state representing ‘aabb’ is 0, as if the computation accepts,

the input could have been ‘aabb’ or ‘aaab.’ In the case of the rejection, we know unambiguously that the state was ‘aaab.’ The probability of rejection in this case is $1/4$. With equal prior probabilities, the probability that the result is inconclusive is $0.5\times1+0.5\times0.75 = 0.875$. Using the unambiguous state discrimination scheme, the probability of an inconclusive result is

$$P_? = 1 - \sum j = 1, 2\eta_j P_j \tag{9.8}$$

where the projection operator which unambiguously measures the state representing‘aabb’ is indexed 1 and is represented by the matrix

$$P_1 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{9.9}$$

and P_2, which unambiguously measures the state representing ‘aaab’ is represented by

$$P_1 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}. \tag{9.10}$$

As the probability that the state can be discriminated unambiguously is one in both cases, the probability of an inconclusive result is zero. Hence, in this case our scheme is not optimal either. These are examples of very specific types of quantum state discrimination, applied to states with a very specific form. Quantum state discrimination is a broad subject area, which I have by no means covered exhaustively, for more detail please see [133].

9.4.2 Distinguishing three states using the walk accepting $\mathcal{L}_{abc}$

Similar calculations to those used above can be used for the example which took a superposition of three words. In this case we wish to distinguish whether we had ‘abc,’ ‘cab’ or ‘ccc’ assuming equal prior probabilities. In this case the total probability of accepting is equal to

that of rejecting, namely 0.5. The following values are obtained for each word given a certain outcome of the computation:

$$P(`abc'|accept) = \frac{1\times\frac{1}{3}}{0.5} = \frac{2}{3} \tag{9.11}$$

$$P(`ccc'|accept) = \frac{\frac{1}{3}\times\frac{1}{3}}{0.5} = \frac{4}{9} \tag{9.12}$$

$$P(`ccc'|reject) = \frac{(1-\frac{1}{3})\times\frac{1}{3}}{0.5} = \frac{4}{9} \tag{9.13}$$

$$P(`cab'|reject) = \frac{1\times\frac{1}{3}}{0.5} = \frac{2}{3} \tag{9.14}$$

In the case where the input is accepted, the probability of the input word having been 'abc' is higher, so we assume that this was the state we had, with probability of error is $2/9$. In the case where the input is rejected, we conclude that we had 'cab,' with probability of error $1/3$. This makes the total probability of error very high, $5/9$, but given that the computation will always reject or accept, I use the specific probabilities in each of these cases to compare to the results obtainable using standard state discrimination. To calculate the probability of error using projective measurements we must evaluate

$$P_E = 1 - \sum_j \eta_j Tr\Pi_j\rho_j \tag{9.15}$$

and as we have pure states we have that the probability of obtaining measurement result ω_j given initial state $|\psi\rangle$ it is the case that

$$P(\omega_j|\psi) = Tr\Pi_j\rho_j = \langle\psi|\Pi_j\psi\rangle. \tag{9.16}$$

Taking the following vector representations of each word, and projection operators Π_j:

$$|\psi_{abc}\rangle = \frac{1}{\sqrt{3}}\begin{pmatrix}1\\0\\0\\0\\1\\0\\0\\0\\1\end{pmatrix} \quad \Pi_{abc} = \begin{pmatrix}1&0&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&1&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\end{pmatrix} \tag{9.17}$$

$$|\psi_{ccc}\rangle = \frac{1}{\sqrt{3}}\begin{pmatrix}0\\0\\1\\0\\0\\1\\0\\0\\1\end{pmatrix} \quad \Pi_{abc} = \begin{pmatrix}0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&1&0&0&0\\0&0&0&0&0&0&1&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&1\end{pmatrix} \tag{9.18}$$

$$|\psi_{cab}\rangle = \frac{1}{\sqrt{3}}\begin{pmatrix}0\\0\\1\\1\\0\\0\\0\\1\\0\end{pmatrix} \quad \Pi_{abc} = \begin{pmatrix}0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&0\\0&0&0&1&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&1&0\\0&0&0&0&0&0&0&0&0\end{pmatrix} \tag{9.19}$$

we get $\langle\psi_{abc}|\Pi_{abc}|\psi_{abc}\rangle = 2/3$, $\langle\psi_{ccc}|\Pi_{ccc}|\psi_{ccc}\rangle = 2/3$ and $\langle\psi_{cab}|\Pi_{cab}|\psi_{cab}\rangle = 1$. Plugging these values into Equation 9.16 gives $1 - 2/9 - 2/9 - 1/3 = 2/9$. Hence in this case, when the computation accepts, the probability of error is the same as when we use the projective measurements indicated. When the computation rejects, the probability of error using the language recognising walks is slightly higher than using the projective measurements. So whilst the overall probability of making the wrong decision regarding which input word we had is very large, the probabilities for specific computational outcomes are equal to, or only slightly larger than, the probability of error using a natural choice of projection operators. There is however, one obvious flaw to this scheme, in that there is no option to say that the word was 'ccc,' which according to the prior probabilities, it will be one third of the time. This

is due to the fact that there are only two outcomes from the computation. So from this example we conclude that whilst the walks developed may be able to perform state discrimination tasks with a relatively low probability of error, they must be modified to deal with cases where we have more than two states.

If the prior probabilities are adjusted, the resulting scenarios can be worse still. Take, for example, the probability of having 'ccc' being 0.8, and dividing the rest of the probability equally between the other two words. In this case, if the walk accepts the input, we have actually lost information about whether we had 'ccc,' as the probability of it being 'ccc' given the walk accepted is 0.7272.... Information has been gained about whether we have 'abc,' but we still conclude that we had 'ccc.' In the case where the walk rejects, the probability that we had 'ccc' is 0.842..., so again we conclude that this was the input. In other words, in both cases we get the same outcome. Analysing the walks in this way will never lead us to conclude that we had any word but 'ccc,' but there was only a 0.8 prior probability of this being the case. This suggests that great care must be taken if we attempt to modify these walks to distinguish between more than two states, as in order to be useful the different outcomes from the walk must lead to different conclusions about which state we had.

9.5 Conclusion

In this chapter I have performed a preliminary investigation of the effects of quantum inputs on the language acceptance properties of the quantum walks developed in the previous chapter. We have seen that in order to have superpositions of more than two words, we need to have an alphabet which contains more than two symbols. Though there are potentially various directions in which research into different types of inputs to the walks developed can be taken, I have concentrated on using them as a potentially novel quantum state discrimination scheme. In Section 9.4 I calculated the probability of correctly identifying which state we had in two cases. In the first case, this was not optimal, whereas in the second case if we had an accepting computation then the computation was able to distinguish one of the states as well as the projection operators selected for the calculation could. This example illustrated the importance of having more than two outcomes from the walk if we wish to discriminate between more than two states, and guaranteeing that the different outcomes of the walk will lead to different conclusions about which state we had.

The first way in which I would like to develop this work is to see if there is any way to reduce the accepting probability of words not in the language accepted by the walk, and hence increase the probability of correctly identifying which state we had. I tried one method to reduce the accepting probability of words not in the language, by attaching a mirror image of the graph accepting $\mathcal{L}_{eq}$ and using another Grover operator at the accepting node. Concatenating the resulting graph structures does not affect the probability of accepting words in the language. The effect of the Grover coin means that for words not in the language, amplitude gets distributed evenly between the new coin states, and so is also deterministically transmitted along the graph, thus the probability of accepting is not reduced. Whilst I have other potential strategies: dividing the input amplitude between various copies of the same graph, or modifying the accepting node, I have not yet had time to test these.

In order to fully develop the proposed novel state discrimination scheme, there is much more work to be done. There are some large differences between language acceptance models and state discrimination schemes which may need to be addressed. For example, clearly there are only two possible outcomes to a computer solving a language acceptance problem: 'accept' and 'reject.' State discrimination schemes typically have more outcomes [133; 134], because the number of possible outcomes must equal the number of possible states. For instance the tetrad state discrimination scheme in [135] has four possible outcomes, reflecting the four possible states.

Beyond looking into the apparent link between language acceptance and state discrimination, there may be applications which make more direct use of quantum inputs. Current computability results concern classical inputs. Given that the input to a quantum computer must be a quantum state, it seems natural to ask how the computer will be affected if that state is in a superposition. The work presented here gives a concrete way of investigating this. Ideally more general language accepting walks would be used for this investigation, as this may enable more general results. When developing walks with a view to devising new state discrimination schemes, the choice of language accepted by the walk may be important. To perfectly distinguish orthogonal states, words which induce initial state vectors which are orthogonal to those of words in the language must be rejected with certainty. In languages such as $\mathcal{L}_{even} = \{a, b\}^* a$ this will not be the case, whereas, as we have seen in this chapter, it is for $\mathcal{L}_{eq}$.

Classical inputs can be grouped together into languages in a natural way. It is difficult to see how quantum inputs could be naturally grouped together, but generalising from a quantum 'word' to a quantum 'language' would be interesting from a theoretical perspective. From

the preliminary investigation the quantum inputs separate themselves into distinct cases based on how many symbols are shared between the words in the superposition. This may provide one possible avenue to developing a quantum language. Naively, it appears that if we wish to fully understand the implications of quantum inputs, we must be able to generalise them to languages. This is because in the classical case, results concerning computational power are formulated in terms of language acceptance, rather than the acceptance of specific words. A wider and more detailed investigation of quantum inputs should inform us as to whether we do indeed require a generalisation of the notion of quantum inputs.

Chapter 10

Conclusion

I now summarise the main results from this thesis and indicate directions in which to take future work. There are three main topics in this thesis: perfect state transfer in quantum walks over graphs based on cycles, self avoiding quantum walks, and formal language recognition, and I describe each of these in turn.

10.1 Perfect state transfer

In Chapter 4 I outlined the results of a systematic study of discrete time quantum walks on graphs based on cycles containing four, six and eight nodes. The graphs examined included up to four extra nodes. I was looking for perfect state transfer between antipodal nodes of the cycle, and tested three coin operators. Of the choices of coin operator, the one which used a permutation operator at nodes of degree two gave rise to most of the cases of perfect state transfer. In general perfect state transfer was found to be very rare, and most of the instances found involved cycles where only the antipodal nodes were modified. Some of these cases worked by having an initial state such that the coin operator ensured that amplitude was only transmitted to the nodes of the cycle after the first step of the walk, rather than the extra nodes.

Some of the modifications of C_4 found to lead to quantum walks which exhibit perfect state transfer could be generalised into families of graphs, and these were studied in detail in Chapter 5. The continuous time walks over these structures were also found to exhibit perfect, or very high amplitude state transfer. In one case it was possible to write down a simple analytic expression for the time evolution of the walk, but the eigensystems of the two other families did not admit easy analytic solutions. Some conditions for when perfect state transfer can take

place have been discovered analytically for both the continuous and discrete time walk [46] but necessary and sufficient conditions for when this transport can happen still need to be found.

10.2 Non-reversal quantum walks

In Chapter 6 a quantum walk using a new type of coin operator was introduced. The dynamics of the walk using two versions of the shift operator, one where a particular coin state is a 'going right' (left, up, down) state, and one which labels coin states by the edge which amplitude came in from. I found that in contrast to other choices of coin operator, the expected position of the walker and the standard deviation were independent of the initial coin states of an initially localised walker. The moments depended only on the parameters used for the coin operator, and in one case it was possible to prove this analytically. The self-avoiding coin used models a dimer, modelling longer self-avoiding walkers is much more difficult, but the eventual goal would be to come up with a fully self-avoiding walk. There remain questions about the type of coin used. For instance, if we use a different geometry, in which case a coin of the appropriate dimensions would be used, are the moments still independent of the initial condition?

10.3 Formal language recognition

Chapters 7 and 8 concerned the applications of quantum walks to computer science. In Chapter 7 I showed two ways in which quantum walks can accept a range of formal languages by directing amplitude from the input nodes to an accepting node. I showed that particular examples of regular, context free and context sensitive languages can be processed with a small number of steps using walks with a spatially distributed input. The walks with a sequentially distributed input can be used to accept specific words using only permutation operators for the coin, but in the example given smaller structures can be used to accept entire languages. There is a lot of future work in this area, of particular interest to me is the development of walks which can accept arbitrary formal languages. I would also like to explore other ways in which quantum walks can be used to solve language acceptance problems.

The observation that the walks developed can take inputs which represent a superposition of words, rather than specific words, lead me to investigate quantum inputs. It was found that using these we can interpret language recognition as a type of quantum state discrimination. Whilst the specific examples used did not perform optimal state discrimination, in one case

due to the probability of error and in another due to the restriction imposed by only having two outcomes from a computation, it may be possible to devise walks which can do this. This would provide an exciting novel approach to this problem, as well as suggesting theoretical links between formal language theory and quantum state discrimination.

10.4 Summary

I have presented work relating to apparently disparate aspects of the quantum walk, but they have an underlying theme, namely they are all key ingredients to designing a quantum computer. We must understand quantum transport if qubits are to move around during a computation. Clearly we must also have a good command of the underlying theoretical computational model before we can implement a quantum computer based on that model. Finally, in order to get any information from a computation, we must know the result of the computation, in which case we must be able to distinguish between different quantum states. The work presented in this thesis contributes to all these aspects of quantum computation, but leaves much scope for future work. Studying the full capabilities of the language recognising walks and developing the potential state discrimination scheme in more detail would lead to genuinely novel approaches to these problems.

Appendix A

Code example

Example code used to generate the shift and coin operators for an arbitrary graph, then run the walk and calculate the probability of being at each node of the graph.

```
import numpy as np
from math import *
from numpy import dot
from math import sqrt
import pickle
from numpy.fft import fft

# function to find degree of node
def finddegree(adj):
   # number of nodes is just length of adj
   degrees = [0 for i in range(len(adj))]
   for i in range(len(adj)):
      number = 0
      for j in range(len(adj[i])):
         number += adj[i][j]
      degrees[i] = number
   return degrees

# function to list coin states for each edge on each vertex of the graph
def finddegrees(adj):
```

```
    # number of nodes is just length of adj
    degrees = [0 for i in range(len(adj))]
    for i in range(len(adj)):
      number = 0
      for j in range(len(adj[i])):
           number += adj[i][j]
      list = [0 for k in range(number)]
      number2 = 0
      for j in range(len(adj[i])):
            if adj[i][j] == 1:
                list[number2] = j
                number2 += 1
      degrees[i] = list
    return degrees


# function to list the first coin state of each node in the graph
def firstnode(degree):
    number = 0
    array = [0 for i in range(len(degree))]
    for i in range(len(degree)):
      array[i] = number
      number += len(degree[i])
    return array

# function to create a DFT coin of appropriate dimension acting on each node
of the graph
def createcoin1(adj):
    degrees = finddegree(adj)
    # number of coin states is just sum of degrees.
    number = 0
    for i in range(len(degrees)):
      number += degrees[i]
    bigcoin = [[0 for i in range(number)] for j in range(number)]
    # number to keep hold of index of parts of coin we have already populated
    number = 0
```

```
   for i in range(len(degrees)):
     deg = degrees[i]
     mat = [[0 for j in range(deg)] for k in range(deg)]
     for j in range(len(mat)):
          mat[j][j] = 1
     coin = fft(mat)
     for j in range(len(coin)):
          for k in range(len(coin[j])):
               coin[j][k] = (1./sqrt(deg))*coin[j][k]
     for j in range(len(coin)):
          for k in range(len(coin)):
               bigcoin[number+j][number + k] = coin[j][k]
     number += deg
   return bigcoin


 # function to create a suitable shift operator from the adjacency matrix
of a graph
 def createshift(adj):
   degree = finddegrees(adj)
   nodeindex = firstnode(degree)
   # create matrix of correct size from degree, size is last index of firstnode
plus degree of that node
   size = nodeindex[len(adj)-1] + len(degree[len(adj)-1])
   array = [[0 for i in range(size)] for j in range(size)]
    # go through each node in turn to assign correct indices to links between
coin states of each node
   for i in range(len(degree)):
     # index of first coin state of node we're on
     index1 = nodeindex[i]
     # scroll through nodes that node were on now is joined to
      for j in range(len(degree[i])):
          coinstate1 = index1 + j
          # index of node coin state index1 + j joins to.  find correct coin
state there to link to by finding which index of degree[nextnode] gives the
```

```
node we're on
            node = degree[i][j]
            for k in range(len(degree[node])):
                 if degree[node][k] == i:
                      coinstate2 = nodeindex[node] + k
            array[coinstate1][coinstate2] = 1
    return array

 # sample adjacency matrix
 adj = [[0,1,0],[1,0,1],[0,1,0]]

 # create appropriate shift operators
 shift = createshift(adj)
 coin = createcoin1(adj)

 # required to calculate the probability of the walker being at each node
 degree = finddegrees(adj)
 firstnodes = firstnode(degree)

 # initialise state
state = [1,0,0,0,0,0]

 # run walk for ten steps
 for t in range(10):
    state = dot(shift, dot(coin, state))

 # calculate probability of the walker being at each node
 probs = [0,0,0]
 for i in range(len(probs)):
    prob = 0
    first = firstnodes[i]
    for k in range(len(degree[i])):
       prob += state[first + k]*state[first+k].conjugate()
    probs[i] = prob
```

References

[1] T. Schoning, "A probabilistic algorithm for k-sat and constraint satisfaction problems," in *Foundations of Computer Science, 1999. 40th Annual Symposium on*, pp. 410–414, IEEE, 1999. 5

[2] C. H. Papadimitriou, "On selecting a satisfying truth assignment," in *Foundations of Computer Science, 1991. Proceedings., 32nd Annual Symposium on*, pp. 163–169, IEEE, 1991. 5

[3] Feynman, R. P., Leighton, R. B., and Sands, M., *Feynman Lectures on Physics*. Addison Wesley, 1964. 5

[4] S. Gudder, *Quantum Probability*. Academic Press Inc., CA, USA, 1988. 5

[5] Aharonov, Y,, Davidovich, L., and Zagury, N., "Quantum random walks," *Physical Review A*, vol. 2, p. 16871690, 1992. 5

[6] G. Grossing and A. Zeilinger, "Quantum cellular automata," *Complex Systems*, vol. 2, p. 197 208, 1988. 5, 80

[7] A. Ambainis, E. Bach, A. Nayak, A. Vishwanath, and J. Watrous, "One-dimensional quantum walks," in *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pp. 37–49, ACM, 2001. 5, 11, 19, 46

[8] E. Farhi and S. Gutmann, "Quantum computation and decision trees," *Physical Review A*, vol. 58, no. 2, p. 915, 1998. 5, 6, 14, 15, 79

[9] A. Ambainis, "Quantum walks and their algorithmic applications," *International Journal of Quantum Information*, vol. 1, no. 04, pp. 507–518, 2003. 5, 6

[10] E. Dantsin, V. Kreinovich, and A. Wolpert, "On quantum versions of record-breaking algorithms for sat," *ACM SIGACT News*, vol. 36, no. 4, pp. 103–108, 2005. 5

[11] A. M. Childs and J. Goldstone, "Spatial search by quantum walk," *Physical Review A*, vol. 70, no. 2, p. 022314, 2004. 5, 27, 56, 95

[12] N. Shenvi, J. Kempe, and K. Whaley, "Quantum random-walk search algorithm," *Physical Review A*, vol. 67, 2003. 5, 12, 95

[13] A. Ambainis, "Quantum walk algorithm for element distinctness," in *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, pp. 22–31, IEEE Computer Society, 2004. 6

[14] F. Magniez, A. Nayak, J. Roland, and M. Santha, "Search via quantum walk," *SIAM Journal on Computing*, vol. 40, no. 1, pp. 142–164, 2011. 6

[15] B. Hein and G. Tanner, "Quantum search algorithms on the hypercube," *Journal of Physics A: Mathematical and Theoretical*, vol. 42, no. 8, p. 085303, 2009. 6

[16] A. Tulsi, "Faster quantum-walk algorithm for the two-dimensional spatial search," *Physical Review A*, vol. 78, no. 1, p. 012310, 2008. 6

[17] B. Hein and G. Tanner, "Quantum search algorithms on a regular lattice," *Physical Review A*, vol. 82, no. 1, p. 012326, 2010. 6

[18] E. Agliari, A. Blumen, and O. Muelken, "Quantum-walk approach to searching on fractal structures," *Physical Review A*, vol. 82, no. 1, p. 012305, 2010. 6

[19] S. D. Berry and J. B. Wang, "Quantum-walk-based search and centrality," *Physical Review A*, vol. 82, no. 4, p. 042333, 2010. 6

[20] H. Krovi, F. Magniez, M. Ozols, and J. Roland, "Finding is as easy as detecting for quantum walks," *Automata, Languages and Programming*, pp. 540–551, 2010. 6

[21] A. M. Childs, "Universal computation by quantum walk," *Physical review letters*, vol. 102, no. 18, p. 180501, 2009. 6, 30, 45

[22] A. M. Childs, D. Gosset, and Z. Webb, "Universal computation by multiparticle quantum walk," *Science*, vol. 339, no. 6121, pp. 791–794, 2013. 6, 33

[23] N. B. Lovett, S. Cooper, M. Everitt, M. Trevers, and V. Kendon, "Universal quantum computation using the discrete-time quantum walk," *Physical Review A*, vol. 81, no. 4, p. 042330, 2010. 6, 20, 45, 95, 101

[24] V. Kendon, "A random walk approach to quantum algorithms," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 364, no. 1849, pp. 3407–3422, 2006. 6

[25] S. E. Venegas-Andraca, "Quantum walks for computer scientists," *Synthesis Lectures on Quantum Computing*, vol. 1, no. 1, pp. 1–119, 2008. 6

[26] J. Kempe, "Discrete quantum walks hit exponentially faster," *Probability theory and related fields*, vol. 133, no. 2, pp. 215–235, 2005. 6, 14, 19, 56, 95

[27] A. M. Childs, E. Farhi, and S. Gutmann, "An example of the difference between quantum and classical random walks," *Quantum Information Processing*, vol. 1, no. 1, pp. 35–43, 2002. 6, 79

[28] A. Childs, R. Cleve, E. Deotto, E. Farhi, S. Gutmann, and D. Spielman, "Exponential algorithmic speedup by quantum walk," *Proc. 35th annual ACM STOC*, pp. 59–68, 2003. 6, 79, 95

[29] S. Bose, "Quantum communication through an unmodulated spin chain," *Physical review letters*, vol. 91, no. 20, p. 207901, 2003. 6, 20, 45, 46, 48, 97

[30] V. Kendon and C. Tamon, "Perfect state transfer in quantum walks on graphs," *Journal of Computational and Theoretical Nanoscience*, vol. 8, no. 3, pp. 422–433, 2011. 6, 14, 20, 45, 47, 48, 49, 50, 53, 97

[31] S. Bose, "Quantum communication through spin chain dynamics: an introductory overview," *Contemporary Physics*, vol. 48, no. 1, pp. 13–30, 2007. 6

[32] A. Kay, "Perfect, efficient, state transfer and its application as a constructive tool," *International Journal of Quantum Information*, vol. 8, no. 04, pp. 641–676, 2010. 6, 48

[33] V. Kendon and B. Tregenna, "Decoherence can be useful in quantum walks," *Physical Review A*, vol. 67, no. 4, p. 042315, 2003. 6, 20

[34] M. Mohseni, P. Rebentrost, S. Lloyd, and A. Aspuru-Guzik. Environment-assisted quantum walks in photosynthetic energy transfer. *Journal of Chemical Physics*, 2008. 6, 45

[35] F. Caruso, W. Chin, A, A. Datta, S. F. Huelga, and M. B. Plenio, "Highly efficient energy excitation transfer in light-harvesting complexes: The fundamental role of noise-assisted transport," *The Journal of chemical physics*, vol. 131, no. 10, 2009. 6

[36] P. Rebentrost, M. Mohseni, I. Kassal, S. Lloyd, and A. Aspuru-Guzik, "Environment-assisted quantum transport," *New Journal of Physics*, vol. 11, no. 3, p. 033003, 2009. 6

[37] A. Montanaro, "Quantum walks on directed graphs," *arXiv preprint quant-ph/0504116*, 2005. 7, 14

[38] J. Watrous, "Quantum simulations of classical random walks and undirected graph connectivity," in *Computational Complexity, 1999. Proceedings. Fourteenth Annual IEEE Conference on*, pp. 180–187, IEEE, 1999. 7, 14

[39] E. W. Montroll and G. H. Weiss, "Random walks on lattices. ii," *Journal of Mathematical Physics*, vol. 6, p. 167, 1965. 9

[40] D. A. Meyer, "From quantum cellular automata to quantum lattice gases," *Journal of Statistical Physics*, vol. 85, no. 5, pp. 551–574, 1996. 10

[41] N. B. Lovett, M. Everitt, R. M. Heath, and V. Kendon, "The quantum walk search algorithm: Factors affecting efficiency," *arXiv preprint arXiv:1110.4366*, 2011. 12, 62, 63, 80, 95

[42] V. Kendon, "Quantum walks on general graphs," *International Journal of Quantum Information*, vol. 4, no. 05, pp. 791–805, 2006. 13, 72

[43] B. Tregenna, W. Flanagan, R. Maile, and V. Kendon, "Controlling discrete quantum walks: coins and initial states," *New Journal of Physics*, vol. 5, no. 1, p. 83, 2003. 13, 20, 46, 50, 56, 70, 71, 72

[44] D. Aharonov, A. Ambainis, J. Kempe, and U. Vazirani, "Quantum walks on graphs," in *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pp. 50–59, ACM, 2001. 13, 47

[45] H. Krovi and T. A. Brun, "Hitting time for quantum walks on the hypercube," *Physical Review A*, vol. 73, no. 3, p. 032341, 2006. 14, 50, 56

[46] H. Krovi and T. A. Brun, "Quantum walks with infinite hitting times," *Physical Review A*, vol. 74, no. 4, p. 042334, 2006. 14, 15, 50, 56, 66, 126

[47] S. Severini, "On the digraph of a unitary matrix," *SIAM Journal on Matrix Analysis and Applications*, vol. 25, no. 1, pp. 295–300, 2003. 14

[48] S. Severini, "Graphs of unitary matrices," *Ars Combinatoria*, vol. 90, pp. 99–117, 2009. 14

[49] V. Kendon, "Decoherence in quantum walks-a review," *Mathematical Structures in Computer Science*, vol. 17, no. 6, pp. 1169–1220, 2007. 14, 18, 56

[50] S. Hoyer and D. A. Meyer, "Faster transport with a directed quantum walk," *Physical Review A*, vol. 79, no. 2, p. 024307, 2009. 14

[51] M. Christandl, N. Datta, A. Ekert, and A. J. Landahl, "Perfect state transfer in quantum spin networks," *Physical Review Letters*, vol. 92, no. 18, p. 187902, 2004. 14, 46, 47, 48, 57, 64

[52] N. Saxena, S. Severini, and I. E. Shparlinski, "Parameters of integral circulant graphs and periodic quantum dynamics," *International Journal of Quantum Information*, vol. 5, no. 03, pp. 417–430, 2007. 14, 20, 47, 48

[53] O. Mülken and A. Blumen, "Slow transport by continuous time quantum walks," *Physical Review E*, vol. 71, no. 1, p. 016101, 2005. 15

[54] C. Moore and A. Russell, "Quantum walks on the hypercube," in *Randomization and Approximation Techniques in Computer Science*, pp. 164–178, Springer, 2002. 19

[55] A. D. Gottlieb, S. Janson, and P. F. Scudo, "Convergence of coined quantum walks on $\mathbb{R}_d$," *Infinite Dimensional Analysis, Quantum Probability and Related Topics*, vol. 8, no. 01, pp. 129–140, 2005. 19

[56] V. Kendon, "Where to quantum walk," *arXiv preprint:1107.3795*, 2011. 19, 24

[57] T. D. Mackay, S. D. Bartlett, L. T. Stephenson, and B. C. Sanders, "Quantum walks in higher dimensions," *Journal of Physics A: Mathematical and Genera*, vol. 35, 2002. 20, 70, 72

[58] M. Bašić, M. D. Petković, and D. Stevanović, "Perfect state transfer in integral circulant graphs," *Applied Mathematics Letters*, vol. 22, no. 7, pp. 1117–1121, 2009. 20, 45, 47, 48

[59] R. P. Feynman, "Simulating physics with computers," *International journal of theoretical physics*, vol. 21, no. 6, pp. 467–488, 1982. 27

[60] D. Deutsch, "Quantum computational networks," *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, vol. 425, no. 1868, pp. 73–90, 1989. 27, 30, 80

[61] P. W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on*, pp. 124–134, IEEE, 1994. 27

[62] D. Deutsch and R. Jozsa, "Rapid solution of problems by quantum computation," *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, vol. 439, no. 1907, pp. 553–558, 1992. 27

[63] T. Toffoli, "Reversible computing," *Automata, Languages and Programming*, pp. 632–644, 1980. 28

[64] R. Landauer, "Irreversibility and heat generation in the computing process," *IBM journal of research and development*, vol. 5, no. 3, pp. 183–191, 1961. 28

[65] M. A. Nielsen and I. L. Chuang, *Quantum computation and quantum information*. Cambridge university press, 2010. 30, 37, 79

[66] S. Lloyd, "Almost any quantum logic gate is universal," *Physical Review Letters*, vol. 75, no. 2, pp. 346–349, 1995. 30

[67] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter, "Elementary gates for quantum computation," *Physical Review A*, vol. 52, no. 5, p. 3457, 1995. 30

[68] D. P. DiVincenzo, "Two-bit gates are universal for quantum computation," *Physical Review A*, vol. 51, no. 2, p. 1015, 1995. 30

[69] M. J. Bremner, C. M. Dawson, J. L. Dodd, A. Gilchrist, A. W. Harrow, D. Mortimer, M. A. Nielsen, and T. J. Osborne, "Practical scheme for quantum computation with any two-qubit entangling gate," *Physical review letters*, vol. 89, no. 24, p. 247902, 2002. 30

[70] J. Hopcroft and J. Ullman, *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979. 35, 110

[71] M. Sipser, *Introduction to the Theory of Computation*. Cengage Learning, 2012. 35

[72] A. Brodsky and B. Pippenger, "Characterizations of quantum finite automata," *SIAM Journal on Computing*, vol. 31, pp. 1456–1478, 2000. 37, 39, 42, 79, 80

[73] C. Moore and J. P. Crutchfield, "Quantum automata and quantum grammars," *Theoretical Computer Science*, pp. 275–306, 2000. 37, 79, 80

[74] A. Kondacs and J. Watrous, "On the power of quantum finite state automata," in *Foundations of Computer Science, 1997. Proceedings., 38th Annual Symposium on*, pp. 66–75, IEEE, 1997. 41, 79, 82, 84

[75] A. Chi-Chih Yao, "Quantum circuit complexity," in *Foundations of Computer Science, 1993. Proceedings., 34th Annual Symposium on*, pp. 352–361, IEEE, 1993. 43

[76] H. Nishimura and M. Ozawa, "Perfect computational equivalence between quantum turing machines and finitely generated uniform quantum circuit families," *Quantum Information Processing*, vol. 8, no. 1, pp. 13–24, 2009. 43, 80, 96

[77] L. Vandersypen, M. Steffen, G. Breyta, C. Yannoni, M. Sherwood, and I. Chuang, "Experimental realization of shor's quantum factoring algorithm using nuclear magnetic resonance," *Nature*, vol. 414, no. 6866, pp. 883–887, 2001. 43

[78] A. Ambainis, M. Beaudry, M. Golovkins, A. Kikusts, M. Mercer, and D. Thérien, "Algebraic results on quantum automata," *Theory of Computing Systems*, vol. 39, no. 1, pp. 165–188, 2006. 43, 80

[79] O. Mülken, V. Bierbaum, and A. Blumen, "Coherent exciton transport in dendrimers and continuous-time quantum walks," *The Journal of chemical physics*, vol. 124, p. 124905, 2006. 45

[80] O. Mülken and A. Blumen, "Continuous-time quantum walks: Models for coherent transport on complex networks," *Physics Reports*, vol. 502, no. 2, pp. 37–87, 2011. 45

[81] C. Godsil, "When can perfect state transfer occur?," *arXiv preprint arXiv:1011.0231*, 2010. 45, 46, 66

[82] W. So, "Integral circulant graphs," *Discrete Mathematics*, vol. 306, no. 1, pp. 153–158, 2006. 45, 48

[83] B. C. Travaglione and G. J. Milburn, "Implementing the quantum random walk," *Physical Review A*, vol. 65, no. 3, p. 032310, 2002. 45, 47

[84] M. Christandl, N. Datta, T. C. Dorlas, A. Ekert, A. Kay, and A. J. Landahl, "Perfect transfer of arbitrary states in quantum spin networks," *Physical Review A*, vol. 71, no. 3, p. 032312, 2005. 46, 97

[85] S. Bose, A. Casaccino, S. Mancini, and S. Severini, "Communication in xyz all-to-all quantum networks with a missing link," *International Journal of Quantum Information*, vol. 7, no. 04, pp. 713–723, 2009. 46

[86] E. Bach, S. Coppersmith, M. P. Goldschen, R. Joynt, and J. Watrous, "One-dimensional quantum walks with absorbing boundaries," *Journal of Computer and System Sciences*, vol. 69, no. 4, pp. 562–592, 2004. 46

[87] P. C. Richter, "Quantum speedup of classical mixing processes," *Physical Review A*, vol. 76, no. 4, p. 042306, 2007. 47

[88] A. Ahmadi, R. Belk, C. Tamon, and C. Wendler, "On mixing in continuous-time quantum walks on some circulant graphs," *Quantum Information and Computation*, vol. 3, no. 6, pp. 611–618, 2003. 47

[89] W. Adamczak, K. Andrew, P. Hernberg, and C. Tamon, "A note on graphs resistant to quantum uniform mixing," *arXiv preprint quant-ph/0308073*, 2003. 47

[90] W. Carlson, E. Harris, J. Rosen, C. Tamon, K. Wrobel, *et al.*, "Universal mixing of quantum walk on graphs," *Quantum Information & Computation*, vol. 7, no. 8, pp. 738–751, 2007. 47

[91] P. Lo, S. Rajaram, D. Schepens, D. Sullivan, C. Tamon, and J. Ward, "Mixing of quantum walk on circulant bunkbeds," *Quantum Information & Computation*, vol. 6, no. 4, pp. 370–381, 2006. 47

[92] K. Nemoto, "Generalized coherent states for su (n) systems," *Journal of Physics A: Mathematical and General*, vol. 33, no. 17, p. 3493, 2000. 48

[93] R. J. Angeles- Canul, R. Norton, M. Opperman, C. Paribello, and C. Tamon, "On quantum perfect state transfer on weighted join graphs," *International Journal of Quantum Information*, vol. 7, p. 14291445, 2009. 57

[94] B. Hayes, "Computing science: How to avoid yourself," *American Scientist*, vol. 86, no. 4, pp. 314–319, 1998. 68

[95] I. Jensen, "A parallel algorithm for the enumeration of self-avoiding polygons on the square lattice," *Journal of Physics A: Mathematical and General*, vol. 36, pp. 5731–5745, 2003. 68

[96] A. J. Guttmann and A. R. Conway, "Square lattice self-avoiding walks and polygons," *Annals of Combinatorics*, vol. 5, pp. 319–345, 2001. 68

[97] A. Skrilos and S. Chirikjian, "Position and orientation distributions for non-reversal random walks using space-group fourier transforms," *J Algebr Stat.*, vol. 1, pp. 27–46, 2010. 69

[98] K. Barr, T. Proctor, B. Hanson, S. Martiel, V. Pavlovic, A. Bullivant, and V. Kendon, "Self-avoiding quantum walks," *arXiv preprint arXiv:1303.1966*, 2013. 73

[99] M. H. Yung and S. Bose, "Perfect state transfer, effective gates, and entanglement generation in engineered bosonic and fermionic networks," *Physical Review A*, vol. 71, no. 3, p. 032310, 2005. 79

[100] S. Wolfram, "Cellular automata as models of complexity," *Nonlinear Physics for Beginners: Fractals, Chaos, Solitons, Pattern Formation, Cellular Automata, Complex Systems*, vol. 311, p. 197, 1998. 80

[101] M. Cook, "Universality in elementary cellular automata," *Complex Systems*, vol. 15, no. 1, pp. 1–40, 2004. 80

[102] S. Wolfram, *A new kind of science*, vol. 5. Wolfram media Champaign, 2002. 80

[103] K. Morita and M. Harao, "Computation universality of one-dimensional reversible (injective) cellular automata," *IEICE TRANSACTIONS (1976-1990)*, vol. 72, no. 6, pp. 758–762, 1989. 80

[104] A. Church, "A set of postulates for the foundation of logic," *The Annals of Mathematics*, vol. 33, no. 2, pp. 346–366, 1932. 80

[105] A. M. Turing, "Computability and λ-definability," *The Journal of Symbolic Logic*, vol. 2, no. 4, pp. 153–163, 1937. 80

[106] M. L. Minsky, "Computation: finite and infinite machines," *Englewood Cliffs, N.J.: Prentice-Hall*, 1967. 80

[107] H. Rogers, *Theory of recursive functions and effective computation*. McGrawHill, 1967. 80

[108] D. Deutsch, "Quantum theory, the church-turing principle and the universal quantum computer," *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, vol. 400, no. 1818, pp. 97–117, 1985. 80

[109] J. Watrous, "On one-dimensional quantum cellular automata," in *Foundations of Computer Science, 1995. Proceedings., 36th Annual Symposium on*, pp. 528–537, IEEE, 1995. 80

[110] K. Wiesner, "Quantum cellular automata," *arXiv preprint arXiv:0808.0679*, 2008. 80

[111] A. Chi-Chih Yao, "Quantum circuit complexity," in *Foundations of Computer Science, 1993. Proceedings., 34th Annual Symposium on*, pp. 352–361, IEEE, 1993. 80

[112] M. Hamada, N. Konno, and E. Segawa, "Relation between coined quantum walks and quantum cellular automata," *arXiv preprint quant-ph/0408100*, 2004. 80

[113] P. Arrighi, V. Nesme, and R. Werner, "One-dimensional quantum cellular automata," *International Journal of Unconventional Computing*, vol. 7, no. 4, 2011. 80

[114] A. Ambainis and J. Watrous, "Two-way finite automata with quantum and classical states," *Theoretical Computer Science*, vol. 287, no. 1, pp. 299–311, 2002. 80

[115] M. Amano and K. Iwama, "Undecidability on quantum finite automata," in *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pp. 368–375, ACM, 1999. 80

[116] A. Nayak, "Optimal lower bounds for quantum automata and random access codes," in *Foundations of Computer Science, 1999. 40th Annual Symposium on*, pp. 369–376, IEEE, 1999. 80

[117] A. Ambainis, A. Ķikusts, and M. Valdats, "On the class of languages recognizable by 1-way quantum finite automata," in *STACS 2001*, pp. 75–86, Springer, 2001. 80

[118] M. Macko, *On closure properties of quantum finite automata*. PhD thesis, Comenius University, 2006. 80

[119] A. Yakaryılmaz, R. Freivalds, A. Say, and R. Agadzanyan, "Quantum computation with devices whose contents are never read," *Unconventional Computation*, pp. 164–174, 2010. 81, 85, 86, 87, 88, 93

[120] A. Yakaryilmaz and A. C. Cem Say, "Languages recognized with unbounded error by quantum finite automata," *arXiv:0809.0073v2*, 2008. 82

[121] J. Rothe and J. Rothe, *Complexity theory and cryptology: an introduction to cryptocomplexity*. Springer, 2005. 91

[122] P. Berman, "Relationship between density and deterministic complexity of mp-complete languages," in *Automata, Languages and Programming*, pp. 63–71, Springer, 1978. 91

[123] S. R. Mahaney, "Sparse complete sets for np: Solution of a conjecture of berman and hartmanis," *Journal of Computer and System Sciences*, vol. 25, no. 2, pp. 130–143, 1982. 91

[124] H. B. Perets, Y. Lahini, F. Pozzi, M. Sorel, R. Morandotti, and Y. Silberberg, "Realization of quantum walks with negligible decoherence in waveguide lattices," *Physical review letters*, vol. 100, no. 17, p. 170506, 2008. 93

[125] M. Karski, L. Förster, J.-M. Choi, A. Steffen, W. Alt, D. Meschede, and A. Widera, "Quantum walk in position space with single optically trapped atoms," *Science*, vol. 325, no. 5937, pp. 174–177, 2009. 93, 95

[126] A. Ambainis, J. Kempe, and A. Rivosh, "Coins make quantum walks faster," in *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pp. 1099–1108, Society for Industrial and Applied Mathematics, 2005. 95

[127] A. Ambainis, A. M. Childs, B. W. Reichardt, R. Špalek, and S. Zhang, "Any and-or formula of size n can be evaluated in time nˆ1/2+o(1) on a quantum computer," *SIAM Journal on Computing*, vol. 39, no. 6, pp. 2513–2530, 2010. 95

[128] H. Buhrman, R. Cleve, J. Watrous, and R. de Wolf, "Quantum fingerprinting," *Phys. Rev. Lett.*, vol. 87, p. 167902, Sep 2001. 97

[129] M. A. Jaro, "Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida," *Journal of the American Statistical Association*, vol. 84, no. 406, pp. 414–420, 1989. 100

[130] M. A. Jaro, "Probabilistic linkage of large public health data files," *Statistics in medicine*, vol. 14, no. 5-7, pp. 491–498, 1995. 100

[131] P. Kurzynski and A. Wojcik, "Quantum walk as a generalized measuring device," *arXiv preprint arXiv:1208.1800*, 2012. 114

[132] C. W. Helstrom, "Quantum detection and estimation theory," *Journal of Statistical Physics*, vol. 1, no. 2, pp. 231–252, 1969. 117

[133] A. Chefles, "Quantum state discrimination," *Contemporary Physics*, vol. 41, no. 6, pp. 401–424, 2000. 118, 122

[134] A. Chefles, "Unambiguous discrimination between linearly independent quantum states," *Physics Letters A*, vol. 239, no. 6, pp. 339–347, 1998. 122

[135] R. B. M. Clarke, V. M. Kendon, A. Chefles, S. M. Barnett, E. Riis, and M. Sasaki, "Experimental realization of optimal detection strategies for overcomplete states," *Physical Review A*, vol. 64, p. 012303, May 2001. 122