Università degli Studi di Camerino

Scuola di Scienze e Tecnologie

Corso di Laurea in Informatica (Classe L-31)

Systematic Taxonomy of Quantum Finite Automata: Bridging Classical and Quantum Computational Models

Candidate
Marta Musso

Student ID 122360

Supervisor
Supervisor Name

Co-supervisor
Co-supervisor Name

A.A. 2024/2025

Abstract

Quantum automata theory investigates how principles of quantum mechanics can be integrated with classical models of computation to reveal new limits and possibilities in computational power. Although the theory offers deep insights, progress has been slowed by inconsistent notation, unclear model definitions, and fragmented comparisons across different approaches. This thesis establishes a unified framework that standardises definitions and systematically compares classical finite automata with their quantum counterparts, focusing on state complexity and language recognition capabilities.

The work begins with a comprehensive review of classical finite automata, including deterministic, nondeterministic, probabilistic, and two-way models, also specifying the formal language theory that underpins these models. This review lays the necessary theoretical foundation for the subsequent discussion of quantum finite automata. The thesis then introduces the foundational principles of quantum mechanics and explains how these underpin the definition of quantum automata. Subsequently, the work reviews various quantum automata models, analyzing their formal definitions, computational dynamics and language recognition properties. Lastly, the thesis presents a unified taxonomy that provides a systematic comparison between classical and quantum automata models, highlighting the computational capabilities and limitations of quantum automata. The unified framework presented here offers clear insights into the computational capabilities and limitations of quantum automata, and it provides a systematic basis for further research in quantum computational models.

Keywords: Quantum automata, finite automata taxonomy, computational complexity, hybrid quantum-classical models, formal language theory

Contents

1. Introduction

The intersection of quantum mechanics and theoretical computer science has given rise to quantum computing, a field that reimagines computational paradigms through the lens of quantum phenomena such as superposition and entanglement. At its core lies quantum automata theory, which seeks to understand how these principles redefine the boundaries of classical computation. Classical Finite Automata (CFAs)—Deterministic Finite Automaton (DFA), Nondeterministic Finite Automaton (NFA), Probabilistic Finite Automaton (PFA), and two-way variants—have long served as the bedrock of formal language theory, offering mathematically rigorous frameworks for analyzing computational complexity and decidability. In contrast, Quantum Finite Automata (QFAs) exhibit probabilistic and non-deterministic behaviors that transcend classical limits, necessitating a coherent framework to classify and analyse their capabilities. This thesis emerges from the recognition that the current landscape of quantum automata theory is fragmented: definitions vary across papers, notations lack standardization, and comparisons between classical and quantum models remain scattered across disjointed works [21]. By systematically unifying these elements, this thesis aims to bridge the conceptual gap between classical and quantum computational models, offering a structured lens through which their interaction can be rigorously studied [5].

The motivation for this work is two-fold: theoretical exploration and practical application. Theoretically, quantum automata represent the simplest quantum computational models, providing a sandbox to explore the interplay between quantum mechanics and computation. They challenge classical intuitions—for instance, quantum parallelism enables certain QFAs, such as the Measure-Many Quantum Finite Automaton (MMQFA), to recognise languages with exponentially fewer states than their classical counterparts [3]. Practically, as quantum hardware advances, understanding the minimal resources required to implement QFAs becomes critical for designing efficient algorithms and robust error-correcting schemes [39]. Yet, progress in the field has been hindered by ambiguities in model definitions. For example, early quantum automata models like the MMQFA and the Measure-Once Quantum Finite Automaton (MOQFA) were defined with differing acceptance criteria, leading to confusion about their relative computational power [29]. Similarly, hybrid models such as the One-Way Quantum Finite Automaton with Classical States (1QFAC) introduce classical memory components, complicating direct comparisons to purely quantum or classical automata [31]. These inconsistencies obscure the true capabilities of quantum models and hinder cross-disciplinary collaboration.

A central observation motivating this thesis is that no single document currently catalogs quantum automata models alongside their classical counterparts. Existing surveys, while valuable, often focus on specific subsets of models or lack the granularity needed to resolve nuanced differences in computational power, closure properties, or decidability [21]. For instance, although the expressive power of Two-Way Quantum Finite Automata (2QFAs) surpasses that of classical Two-Way Classical Finite Automata (2CFAs), the conditions under which this advantage manifests—such as the role of quantum interference in recognizing non-regular languages—remain underexplored in a unified context [59]. In contrast, this work adopts a taxonomic approach, dissecting each model's formal definition, acceptance criteria, and operational dynamics while contextualizing its position within the broader hierarchy of automata. Moreover, the literature review in this thesis is dedicated to clearly identifying the specific classes of languages each automaton model accepts. This approach not only clarifies existing results but also identifies gaps where further research is needed, such as the decidability of equivalence problems for QFAs with mixed states or the precise trade-offs between quantum entanglement and space efficiency [24].

The research challenges addressed in this thesis are multifaceted. First, reconciling disparate notation and definitions requires a meticulous synthesis of foundational and contemporary literature. For example, the transition from unitary operations in MO-QFA to superoperator-based transitions in open quantum systems, as seen in Open Time Evolution Quantum Finite Automata (OTQFAs), calls for a unified formalism to compare their computational behaviors [9, 11]. Second, characterizing the relationships between classical and quantum models necessitates a framework that accounts for both their similarities (e.g., the ability of 1QFACs to simulate DFAs) and their divergences (e.g., the exponential state advantage of 2QFAs over Two-Way Probabilistic Finite Automata (2PFAs)). Third, the absence of standardised pumping lemmas or minimization algorithms for QFAs complicates efforts to classify their language recognition capabilities, a challenge that this thesis tackles through a comparative analysis of closure properties and equivalence criteria [56].

To address these challenges, this thesis employs a structured methodology that unfolds in several stages. It begins by grounding the discussion in classical automata theory, revisiting DFA, NFA, PFA, and two-way variants to establish foundational concepts. Building on this, the thesis introduces the foundational principles of quantum mechanics—such as superposition, entanglement and measurement—which provide the basis for defining QFAs [39]. With this dual background in place, the work systematically explores various quantum models, ranging from early variants like the MOQFA [33] and the MMQFA [29] to advanced hybrids such as the 1QFAC and enhanced models (e.g., the Enhanced Quantum Finite Automaton (EQFA)). Each model is rigorously analysed along several dimensions: its formal definition is standardised, its acceptance criteria are scrutinised, and its computational dynamics—such as the role of measurement timing and the interplay between quantum and classical states—are carefully dissected, with particular emphasis on identifying the exact classes of formal languages recognised by each model.

A significant contribution of this work is the development of a hierarchical taxonomy of automata models, which organises both classical and quantum automata into a coherent structure based on their computational features and complexity classes. For example, the analysis reveals that 2QFAs occupy a higher complexity class than their one-way counterparts, while hybrid models such as the 1QFAC serve as an intermediate bridge between purely quantum and purely classical models [59]. The taxonomy further highlights open research questions, such as the precise relationships between models that employ different quantum operational frameworks (e.g., Ancilla-Based Quantum Finite Automata (A-QFAs) versus Generalised Quantum Finite Automata (gQFAs)) and the conditions under which quantum automata outperform probabilistic models in language recognition tasks [24].

The thesis is organised to guide the reader through these progressively complex layers of analysis. Following this introduction, Chapter 2 consolidates foundational concepts from both classical automata theory and quantum mechanics, providing a unified background for the discussions that follow. Chapter **??** presents a comprehensive catalog of quantum automata models; each model is formally defined, its computational dynamics are analysed, and its language recognition capabilities are explicitly detailed. Chapter **??** synthesises these findings by evaluating expressive power, closure properties, and decidability issues across different models. Finally, Chapter 5 concludes by reflecting on the thesis's contributions and outlining directions for future research, such as solving open questions in equivalence checking for QFAs [31], extending pumping lemmas to quantum models [56], developing minimization algorithms for hybrid automata and compilation techniques for quantum circuits.

In essence, this thesis seeks to transform quantum automata theory from a collection of isolated results into a cohesive and systematic framework. By standardizing definitions, clarifying the relationships between different models, and identifying critical open challenges, the work provides both a valuable reference for researchers and a solid methodological basis for future advancements in quantum computational models. As quantum computing transitions from theory to practice, such systematic foundations will be essential for harnessing the full potential of quantum-enhanced computation.

2. Background

The study of quantum automata theory necessitates a thorough grounding in both classical computational models and the quantum mechanical principles that redefine their capabilities. This chapter systematically establishes the conceptual foundation for analyzing quantum automata by first revisiting classical finite automata—the cornerstone of formal language theory—and then introducing the quantum mechanical framework that underpins novel computational paradigms.

We begin with an in-depth exploration of classical finite automata, which serve as the theoretical bedrock for understanding computational limits and language recognition. DFAs, NFAs, PFAs, and their two-way variants are analysed through their formal definitions, operational dynamics, and closure properties. These models collectively define the boundaries of classical computation, particularly in recognizing regular languages and in delineating limitations when handling context-free or stochastic languages. Foundational works such as Hopcroft et al. [25, 50]—which formalised the equivalence between DFAs and NFAs—and Rabin's seminal work on probabilistic automata [43] have played a pivotal role in shaping this understanding.

The discussion then transitions to the essential principles of quantum mechanics required for quantum computation. Key concepts such as qubit representation, quantum superposition, entanglement, and the measurement postulate are introduced and contextualised within computational frameworks. These principles fundamentally depart from classical bit-based processing by allowing phenomena such as state interference and probabilistic state collapse. The mathematical formalism provided by Nielsen and Chuang [39] serves as the backbone for these quantum operations.

2.1 Classical Finite Automata and Languages

Finite automata form the cornerstone of formal language theory, providing mathematical frameworks for analyzing computational limits and language recognition capabilities. This section systematically examines DFAs, NFAs, PFAs, and two-way variants such as Two-Way Deterministic Finite Automata (2DFAs), Two-Way Nondeterministic Finite Automata (2NFAs), and 2PFAs, emphasizing their structural relationships, operational dynamics, and computational boundaries. These classical models not only define the limits of traditional computation but also serve as a benchmark for more advanced paradigms, including quantum automata. Importantly, a primary goal of this review is to clearly identify the exact classes of languages each automaton model accepts.

2.1.1 Formal Languages and Grammars

The study of automata begins with the fundamental concepts of formal language theory, a field that emerged from the pioneering work of Stephen Kleene, Noam Chomsky, Alan Turing, and Michael Rabin. Kleene's early work on the representation of events in nerve nets and finite automata [27] laid the groundwork for understanding the algebraic structure of languages. Chomsky's introduction of language hierarchies [14] further clarified how different classes of languages can be recognised by increasingly powerful computational models [51]. Turing's conceptualization of computation [53] provided a model for algorithmic processes, while Rabin's introduction of probabilistic automata [43] expanded the framework to include models where transitions are governed by probability [15].

These seminal contributions collectively established the mathematical scaffolding for the study of automata and formal languages. Their rigorous definitions and operations are not only abstract mathematical constructs; they serve as the basis for practical applications. For example, regular expressions—rooted in the theory of regular languages—are widely used in text processing and programming language design [26, 45]. Similarly, the limitations of context-free languages have led to the development of more advanced parsing techniques in compilers [14].

Notation 2.1.1 (Symbols)*.* In this thesis, the following notations are used:

- Σ: an alphabet, i.e., a non-empty finite set of symbols.
- Σ^*: the Kleene closure of Σ, the set of all finite strings over Σ.
- ϵ: the empty string (with $\|\epsilon\| = 0$).
- $\|w\|$: the length of a string w.

Alphabets and Strings

Definition 2.1.1 (Alphabet)**.** An *alphabet* Σ is a non-empty, finite set of symbols that serve as the basic elements for constructing strings and, consequently, languages.

Example 2.1.1. Binary Alphabet: $\Sigma = \{0, 1\}$ is central to digital computing and coding theory [51].

Example 2.1.2. ASCII Alphabet: Σ_{ASCII}, which contains 128 distinct characters used for text encoding [13].

Definition 2.1.2 (String)**.** A *string* (or *word*) w over Σ is a finite sequence of symbols $a_1 a_2 \ldots a_n$, where each $a_i \in \Sigma$. The **length** of w, denoted by $\|w\|$, is the total number of symbols in the string. The special string ϵ, known as the **empty string**, has a length of zero ($\|\epsilon\| = 0$) [51].

Example 2.1.3. For $\Sigma = \{a, b\}$, consider the string $w = aba$. Then, $\|w\| = 3$. Conversely, $w = \epsilon$ represents the absence of input.

Remark. The concepts of Σ, Σ^*, and ϵ form the foundation for all language constructions and are pivotal when defining operations such as concatenation and the Kleene star.

In addition to defining strings, several operations are essential for manipulating them:

- **Reversal**: The operation w^R produces the string obtained by reversing the order of symbols in w (e.g., $(abc)^R = cba$) [51].
- **Substring**: A string v is a substring of w if there exist (possibly empty) strings x and y such that $w = xvy$ [51].

Languages and Operations

Definition 2.1.3 (Kleene Closure)**.** The *Kleene closure* [27] of an alphabet Σ is the set of all finite strings over Σ:

$$\Sigma^* = \bigcup_{n=0}^{\infty} \Sigma^n, \quad \text{where } \Sigma^0 = \{\epsilon\}.$$

Definition 2.1.4 (Language)**.** A *language* L is a subset of Σ^*, where Σ^* denotes the Kleene closure of Σ:

$$L \subseteq \Sigma^*.$$

Languages are constructed and manipulated using various operations. These operations are central to proofs of language properties and decidability:

1. **Concatenation**: For two languages L_1 and L_2, their concatenation is defined by

   $$L_1 \cdot L_2 = \{xy \mid x \in L_1,\ y \in L_2\}.$$

   Example 2.1.4. Let $L_1 = \{a, ab\}$ and $L_2 = \{b, ba\}$. Then,

   $$L_1 \cdot L_2 = \{ab, aba, abb, abba\},$$

   which demonstrates the formation of new languages by joining strings from different languages [51].

2. **Union/Intersection**: These operations are defined as:

   $$L_1 \cup L_2 = \{w \mid w \in L_1 \text{ or } w \in L_2\},$$
   $$L_1 \cap L_2 = \{w \mid w \in L_1 \text{ and } w \in L_2\}.$$

   They allow the combination or filtering of languages based on shared elements [51].

3. **Kleene Star**: The Kleene star operation generates the set of all possible concatenations (including the empty string) of elements from a language:

   $$L^* = \bigcup_{i=0}^{\infty} L^i, \quad \text{where } L^0 = \{\epsilon\} \text{ and } L^i = L \cdot L^{i-1}.$$

   Example 2.1.5. For $L = \{0, 1\}$, the set L^* comprises all binary strings, including ϵ [51].

4. **Complement**: The complement of a language L with respect to Σ^* is defined as

   $$\overline{L} = \Sigma^* \backslash L.$$

   This operation is useful for expressing languages indirectly [51].

5. **Homomorphism**: A homomorphism is a function $h : \Sigma^* \to \Gamma^*$ that maps each symbol in Σ to a string in Γ^*. For example, if $h(a) = 01$, then every occurrence of a in a string is replaced by 01 [51].

6. **Inverse Homomorphism**: Given a homomorphism h, the inverse homomorphism is defined by
$$h^{-1}(L) = \{w \mid h(w) \in L\},$$
which retrieves the pre-images from the target language [51].

Language Categories

Languages are classified according to the type of automata that recognise them and their inherent structural complexity. The main categories are as follows:

1. **Regular Languages (REGs)**: These languages are recognised by DFAs and NFAs and can be generated by regular expressions [51].

   Example 2.1.6. $L = \{w \in \{a, b\}^* \mid w \text{ contains } aba\}$ is a regular language [51].

2. **Reversible Regular Languages (REVs)**: A specialised subclass of regular languages, *reversible regular languages*, are those recognised by deterministic finite automata (DFAs) that are reversible—i.e., for every state q and symbol σ, there is at most one predecessor state q' such that $\delta(q', \sigma) = q$. These languages play a critical role in quantum automata theory, as models like the Measure-Once Quantum Finite Automaton (MO-1QFA) recognise exactly this class [29].

3. **Context-Free Languages (CFLs)**: These languages are recognised by Deterministic Pushdown Automata (PDAs) and generated by context-free grammars [14, 51].

   Example 2.1.7. $L_{\text{pal}} = \{ww^R \mid w \in \{a, b\}^*\}$, the language of palindromes, is a classic example [14].

4. **Context-Sensitive Languages (CSLs)**: Recognised by Linear-Bounded Automata (LBAs), these languages have structural constraints that extend beyond context-free languages [14, 51].

   Example 2.1.8. $L = \{a^n b^n c^n \mid n \geqslant 1\}$ is an example of a context-sensitive language [14].

5. **Recursively Enumerable Languages (Type-0)**: Recognised by Turing Machines (TMs), these languages embody the notion of algorithmic computability [51, 53].

   Example 2.1.9. The language corresponding to the Halting Problem is recursively enumerable [51].

6. **Stochastic Languages**: This class of languages strictly contains REGs and is recognised by PFAs with bounded error, allowing for probabilistic acceptance criteria [43, 15].

   Example 2.1.10. $L_{\text{maj}} = \{w \in \{a, b\}^* \mid |w|_a > |w|_b\}$ is a stochastic language. A PFA can accept it with probability $\geqslant \frac{2}{3}$ for $w \in L_{\text{maj}}$ [43].

Definition 2.1.5 (Regular Language)**.** A language $L \subseteq \Sigma^*$ is *regular* if there exists a DFA (or an equivalent NFA) that accepts exactly the strings in L.

Definition 2.1.6 (Reversible Regular Language)**.** A language $L \subseteq \Sigma^*$ is *reversible regular* if there exists a reversible DFA that accepts L, where reversibility ensures the transition function is injective for each symbol.

Example 2.1.11. The language $L = \{w \in \{0,1\}^* \mid \text{number of 0s in } w \text{ is even}\}$ is reversible regular. A reversible DFA for L has transitions that form a bijection between states for each input symbol [41].

Theorem 2.1.1 (Pumping Lemma for Regular Languages)**.** *Let $L \subseteq \Sigma^*$ be a regular language. Then there exists an integer $p \geqslant 1$, known as the pumping length, such that every string $s \in L$ with $\|s\| \geqslant p$ can be decomposed into three parts $s = xyz$, satisfying:*

1. $|xy| \leqslant p$,
2. $|y| \geqslant 1$, *and*
3. *For all $i \geqslant 0$, the string $xy^iz \in L$.*

Corollary 2.1.1. *If a language L fails to satisfy the conditions of Theorem 2.1.1 for any possible pumping length p, then L is not regular.*

Closure Properties

Closure properties determine how language classes behave under various operations—a critical aspect in proving decidability and constructing new languages:

- **REGs**: closed under union, intersection, complement, concatenation, and Kleene star [51].
- **CFLs**: closed under union and Kleene star, but not under intersection or complement [14, 51].
- **CSLs**: closed under union, intersection, and complement [14, 51].
- **Stochastic Languages**: closed under union, intersection, and concatenation, but not under complementation or Kleene star [43, 15].

Observation 2.1.1. Closure properties not only simplify the construction of new languages from known ones but also play a key role in proving undecidability results. For instance, the non-closure of context-free languages under intersection and complementation is a cornerstone in many undecidability proofs [51].

Example 2.1.12. The closure of regular languages under intersection guarantees that if $L_1, L_2 \in DFAs$ (or, equivalently, recognised by NFAs) then $L_1 \cap L_2$ is also regular. In contrast, although stochastic languages are closed under intersection, they are not closed under complementation [43], as demonstrated by the inability to recognise

$$\overline{L_{\text{maj}}} \quad \text{for} \quad L_{\text{maj}} = \{w \mid |w|_a > |w|_b\}.$$

Operation	REGs	CFLs	CSLs	Stochastic	Type-0
Union	✓	✓	✓	✓	✓
Intersection	✓	×	✓	✓	✓
Complement	✓	×	✓	×	✓
Concatenation	✓	✓	✓	✓	✓
Kleene Star	✓	✓	✓	×	✓

Table 2.1: Comparison of closure properties for different language classes

Chomsky Hierarchy

Formal languages are organised into a hierarchical framework known as the Chomsky hierarchy [14, 51]:

1. **Type-3 (Regular)**: Languages recognised by DFAs (or NFAs) [51].
2. **Type-2 (Context-Free)**: Languages recognised by PDAs [14].
3. **Type-1 (Context-Sensitive)**: Languages recognised by LBA [14].
4. **Type-0 (Recursively Enumerable)**: Languages recognised by TMs [51, 53].

Concept 2.1.1. The Chomsky hierarchy not only classifies languages based on the computational power needed for recognition but also reflects the trade-offs between expressiveness and computational complexity [51].

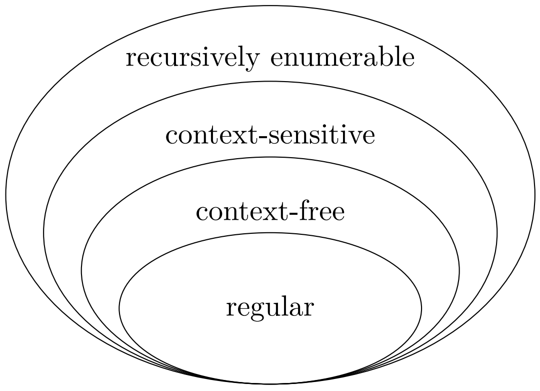

Figure 2.1: Chomsky hierarchy of formal languages

Practical Implications

The theoretical constructs discussed above are not only of academic interest but also have significant practical applications:

- **Regular Expressions**: Extensively used in text processing (e.g., in tools such as `grep` and in lexical analysers) [26].
- **Context-Free Grammars**: Form the basis for defining the syntax of programming languages such as Python and Java [45].

- **Closure Properties**: Provide a framework for proving decidability results (e.g., the emptiness problem for DFAs) [51].
- **Probabilistic Models**: Applied in natural language processing and speech recognition for uncertainty modeling [15].

2.1.2 Classical Finite Automata Definition Fundamentals

All automata share several core structural components that provide the basis for their computational behavior [25, 51].

Definition 2.1.7 (Classical Finite Automaton)**.** A *finite automaton* is a computational model that processes input symbols to recognise languages. Formally, a finite automaton M is a quintuple $(Q, \Sigma, \delta, q_0, F)$, where:

- **States (Q)**: A finite set of configurations representing the progress of computation [51].
- **Input Alphabet (Σ)**: A finite set of symbols that the automaton processes [25, 52].
- **Transition Function (δ)**: A function that governs state changes based on input. For deterministic models, $\delta : Q \times \Sigma \to Q$ is a total function (defined for all state-symbol pairs); for nondeterministic models, $\delta : Q \times \Sigma \to 2^Q$ [51].
- **Initial State ($q_0 \in Q$)**: The starting configuration of the automaton [25].
- **Accept States ($F \subseteq Q$)**: A subset of states indicating successful recognition of an input string [51].

Remark. The quintuple $(Q, \Sigma, \delta, q_0, F)$ provides a standardised representation for comparing automata models. The term "total function" in DFAs means every state-symbol pair has exactly one transition [51].

Example 2.1.13. The DFA in Figure 2.2 is defined by:

- $Q = \{q_0, q_1\}$,
- $\Sigma = \{0, 1\}$,
- Transitions: $\delta(q_0, 0) = q_0$, $\delta(q_0, 1) = q_1$, $\delta(q_1, 0) = q_1$, $\delta(q_1, 1) = q_0$,
- $F = \{q_0\}$, accepting strings with an even number of 1s.

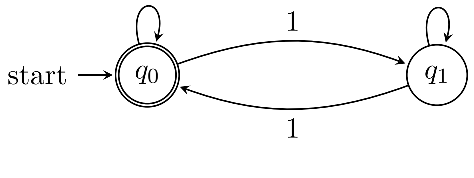

Figure 2.2: DFA recognizing strings of 0s and 1s with an even number of 1s.

Observation 2.1.2. Graphical representations provide an intuitive understanding of automata behavior [30, 52]. Key conventions include:

- **States**: Circles labeled with state names (e.g., q_1 in Figure 2.2).
- **Initial State**: Marked by an unlabeled incoming arrow.
- **Accept States**: Double circles (e.g., q_0 in Figure 2.2).
- **Transitions**: Arrows labeled with input symbols.

Automaton	**Memory**	**Transitions**	**Acceptance Condition**	**Reference**
DFA	None	Deterministic	Final state	[51]
NFA	None	Nondeterministic	Any accepting path	[25]
PDA	Stack	Nondeterministic	Final state or empty stack	[51]
TM	Infinite tape	Deterministic	Halting in accept state	[53]

Table 2.2: Automata variations: structural and operational differences

2.1.3 Deterministic Finite Automaton

Definition 2.1.8 (Deterministic Finite Automata)**.** A DFA is a quintuple

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

where:

- Q is a finite set of states,
- Σ is a finite input alphabet,
- $\delta : Q \times \Sigma \to Q$ is a total transition function (defined for all $(q, \sigma) \in Q \times \Sigma$),
- $q_0 \in Q$ is the initial state,
- $F \subseteq Q$ is the set of accept states.

Theorem 2.1.2 (Myhill-Nerode Theorem)**.** *A language $L \subseteq \Sigma^*$ is regular if and only if the equivalence relation $\sim_L$, defined by*

$$x \sim_L y \iff \forall z \in \Sigma^*,\ xz \in L \iff yz \in L,$$

has finitely many equivalence classes. The number of equivalence classes equals the number of states in the minimal DFA for L [51, 35, 38].

Example 2.1.14. Consider the language

$$L = \{w \in \{0,1\}^* \mid \text{the number of 0's and 1's in } w \text{ are both even}\}.$$

Figure 2.3 shows a minimal DFA for L. Each state encodes a parity pair (p_0, p_1), where p_0 and p_1 represent the parity (even/odd) of 0's and 1's, respectively.

Observation 2.1.3. The Myhill-Nerode Theorem 2.1.2 explains why this DFA is minimal: the four states correspond to the four equivalence classes of $\sim_L$. No smaller DFA can recognise L [51].

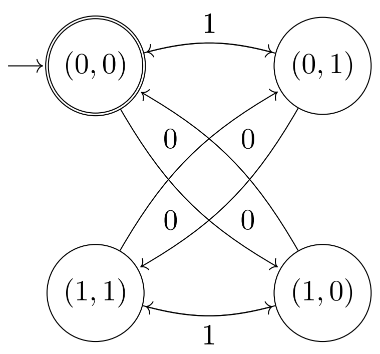

Figure 2.3: Minimal DFA for the language of strings with even numbers of 0's and 1's.

2.1.4 Nondeterministic Finite Automaton

Definition 2.1.9 (Nondeterministic Finite Automata)**.** A NFA is a quintuple

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

where:

- Q is a finite set of states [30],
- Σ is an input alphabet [52],
- $\delta : Q \times (\Sigma \cup \{\epsilon\}) \to 2^Q$ is a nondeterministic transition function [25],
- $q_0 \in Q$ is the initial state,
- $F \subseteq Q$ is the set of accepting states.

Remark. Unlike DFAs, a NFAs may have multiple transitions for a given state and input symbol, including transitions on the empty string ϵ. This nondeterminism allows for multiple computational paths.

Example 2.1.15. Figure 2.4 depicts a NFA that recognises the language

$$L = \{w \in \{a, b\}^* \mid w \text{ contains the substring } ab\}.$$

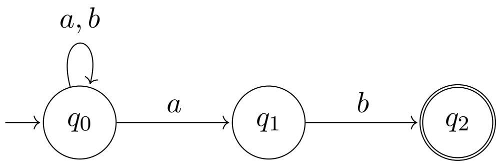

Figure 2.4: NFA recognizing $L = \Sigma^* ab$

Algorithm 2.1.1 (Subset Construction for NFAs)**.** To convert an NFA $N = (Q, \Sigma, \delta, q_0, F)$ into an equivalent DFA [25, 30]:

1. Compute the ϵ-closure of the initial state: $S_0 = \epsilon\text{-closure}(\{q_0\})$.
2. For each DFA state $S \subseteq Q$ and each input symbol $\sigma \in \Sigma$, define

$$\delta_{\text{DFA}}(S, \sigma) = \epsilon\text{-closure}\Big(\bigcup_{q \in S} \delta(q, \sigma)\Big).$$

3. Mark S as accepting if $S \cap F \neq \emptyset$.
4. Repeat until no new states are produced.

Observation 2.1.4. The subset construction algorithm may produce up to $2^{|Q|}$ states in the worst case, illustrating a potential state explosion when converting an NFA to a DFA [51].

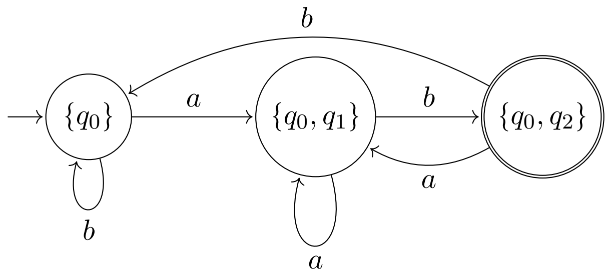

Figure 2.5: Equivalent DFA for NFA in Figure 2.4 [25]

2.1.5 Probabilistic Finite Automaton

Definition 2.1.10 (Probabilistic Finite Automaton)**.** A PFA is a quintuple

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

where:

- Q is a finite set of states,
- Σ is a finite input alphabet,
- $\delta : Q \times \Sigma \times Q \to [0, 1]$ is a probabilistic transition function [43] such that

$$\sum_{q' \in Q} \delta(q, \sigma, q') = 1 \quad \text{for all } q \in Q \text{ and } \sigma \in \Sigma,$$

- $\pi \in \mathbb{R}^{|Q|}$ is an initial state distribution vector with

$$\sum_{q \in Q} \pi_q = 1,$$

- $F \subseteq Q$ is the set of accepting states.

Remark. In a PFA, transitions are probabilistic. The acceptance of an input string is determined by whether the cumulative probability of ending in an accepting state exceeds a chosen cut-point.

Example 2.1.16. Figure 2.6 illustrates a PFA that recognises the language

$$L_{\text{maj}} = \{w \in \{a, b\}^* \mid |w|_a > |w|_b\},$$

where the acceptance probability is at least $\frac{2}{3}$.

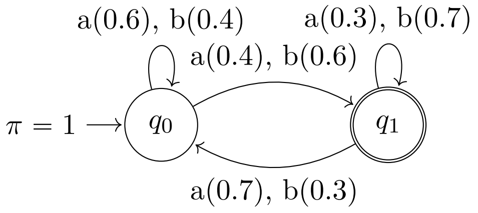

Figure 2.6: PFA for majority language with probabilistic transitions

Theorem 2.1.3 (Rabin's Theorem for PFAs)**.** *A PFA with an isolated cut-point recognises exactly the class of regular languages [43].*

Proposition 2.1.1. *If a PFA employs a non-isolated cut-point (e.g., $\lambda = 0$), it may recognise languages beyond the regular class, including some context-sensitive languages [40].*

Corollary 2.1.2. *For a PFA with a strict cut-point ($\lambda = 1$), the recognised language is equivalent to that of a DFA.*

Observation 2.1.5. The closure properties of PFAs differ from those of classical finite automata. In particular, complementation is not directly achievable unless an isolated cut-point is used [15].

2.1.6 Two-Way Finite Automata Variants

Two-way finite automata extend the classical one-way model by allowing the read head to move in both directions over the input. Although this extra power does not increase the class of recognizable languages, two-way models can be exponentially more succinct than one-way models [46, 30] and naturally lend themselves to algorithms in several contexts (e.g., in complexity analysis and even quantum models).

Two-Way Deterministic Finite Automaton

Definition 2.1.11 (Two-Way Deterministic Finite Automaton)**.** A 2DFA is formally defined as an 8-tuple

$$M = (Q, \Sigma, \rhd, \dashv, \delta, s, t, r),$$

where:

- Q is a finite set of states,
- Σ is a finite input alphabet,
- $\rhd$ and $\dashv$ are special symbols called the left and right endmarkers, respectively (with $\rhd, \dashv \notin \Sigma$),
- $\delta : Q \times (\Sigma \cup \{\rhd, \dashv\}) \to Q \times \{\rhd, \dashv\}$ is the transition function,
- $s \in Q$ is the start state,
- $t \in Q$ is the unique accept state,
- $r \in Q$ (with $r \neq t$) is the unique reject state.

In addition, the transition function is assumed to satisfy:

- For every state $q \in Q$, when reading the left endmarker $\rhd$, the head always moves in the $\dashv$ direction; that is, $\delta(q, \rhd) = (q', \dashv)$ for some $q' \in Q$.
- Similarly, when reading the right endmarker $\dashv$, the head always moves in the $\rhd$ direction: $\delta(q, \dashv) = (q', \rhd)$.
- Once the machine reaches the accept state t (or the reject state r), it remains there (the transition always maps back to itself) while moving in a fixed direction.

Remark. The two-way motion allows the automaton to perform multiple passes over the input, which can result in an exponential reduction in the number of states compared to one-way automata, though at the expense of increased operational complexity.

Example 2.1.17. Consider the language

$$L = \{w \in \{0, 1\}^* \mid \text{the first symbol of } w \text{ equals the last symbol}\}.$$

A 2DFA for L operates as follows:

1. Start at $\rhd$ and move in the $\dashv$ direction to read the first symbol, transitioning to a state encoding this symbol (e.g., q_a for $a \in \{0, 1\}$).
2. Continue moving in the $\dashv$ direction until $\dashv$ is reached.
3. Reverse direction and scan in the $\rhd$ direction to the last symbol.
4. Compare the stored symbol (encoded in the current state) with the last symbol. Accept if they match; reject otherwise.

Observation 2.1.6. Although every 2DFA can be simulated by a One-Way Deterministic Finite Automaton (1DFA), such a simulation may require an exponential increase in the number of states [47].

Operational Mechanics The two-way motion enables the automaton to make multiple passes over the input, which is particularly useful for verifying properties that depend on both the prefix and the suffix of the input string.

State Complexity and Conversion Algorithms For some families of regular languages, 2DFAs can be exponentially more succinct than their one-way counterparts. Conversion algorithms—such as those proposed by Shepherdson and Kozen—use crossing sequences to simulate two-way behavior in a one-way DFA, typically at the cost of exponential state blow-up.

Two-Way Nondeterministic Finite Automaton

Definition 2.1.12 (Two-Way Nondeterministic Finite Automaton)**.** A 2NFA is defined similarly to a 2DFA but with a nondeterministic transition function [46]. Formally, a 2NFA is an 8-tuple

$$M = (Q, \Sigma, \rhd, \dashv, \delta, s, t, r),$$

where:

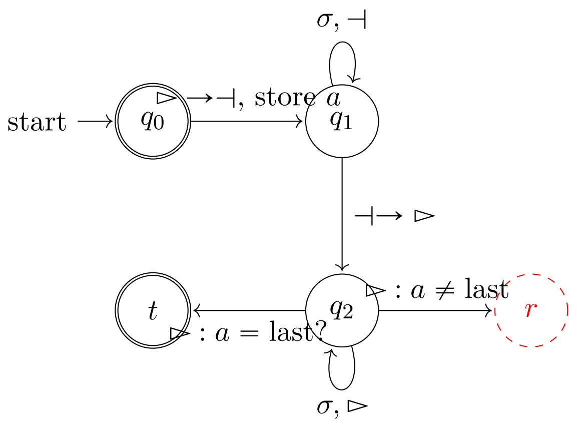

Figure 2.7: 2DFA for $L = \{w \mid w_1 = w_{|w|}\}$. States encode the first symbol $a \in \{0, 1\}$. Transitions use $\rhd / \dashv$ to denote head movement directions and endmarkers.

- Q is a finite set of states,
- Σ is a finite input alphabet,
- $\rhd$ and $\dashv$ are the left and right endmarkers (with $\rhd, \dashv \notin \Sigma$),
- $\delta : Q \times (\Sigma \cup \{\rhd, \dashv\}) \to 2^{Q \times \{\rhd, \dashv\}}$ is the nondeterministic transition function,
- $s \in Q$ is the start state,
- $t \in Q$ is the unique accept state, and
- $r \in Q$ (with $r \neq t$) is the unique reject state.

The transition function obeys similar boundary conditions as in the 2DFA case.

Remark. The nondeterminism in a 2NFA allows it to "guess" important positions within the input and verify them via bidirectional traversal, which can lead to significant state savings compared to deterministic models.

Example 2.1.18. Consider the language

$$L_{sym} = \{w \in \{0, 1\}^* \mid \text{the first two symbols equal the last two symbols}\}.$$

A high-level description of a 2NFA for L_{sym} is:

1. Scan in the $\dashv$ direction from the left endmarker $\rhd$ while nondeterministically guessing the point where comparison will occur.
2. Upon reaching the right endmarker $\dashv$, reverse direction.
3. While moving in the $\rhd$ direction, nondeterministically check that the stored first two symbols match the corresponding symbols at the end.
4. If both comparisons succeed, transition to the accept state t; otherwise, transition to the reject state r.

Figure 2.8 schematically illustrates this guess-and-check mechanism.

Observation 2.1.7. 2NFAs can be exponentially more succinct than one-way DFAs, even though the class of languages they recognise remains the same (i.e., the regular languages).

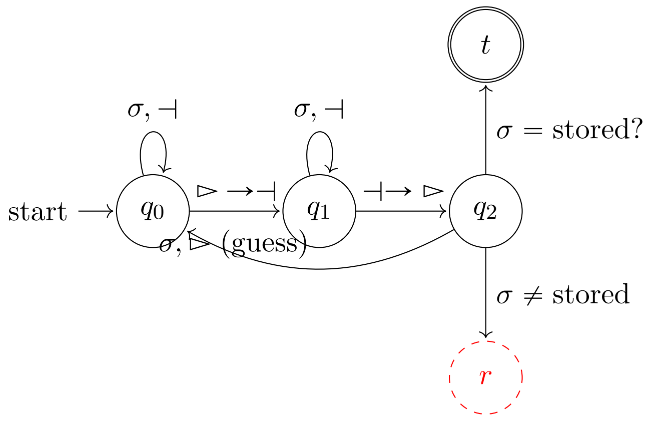

Figure 2.8: 2NFA for $L_{\text{sym}} = \{ww^R\}$. The machine nondeterministically guesses the midpoint (via transition $q_2 \to q_0$) and verifies symmetry.

Two-Way Probabilistic Finite Automaton

Definition 2.1.13 (Two-Way Probabilistic Finite Automaton)**.** A 2PFA is an 8-tuple

$$M = (Q, \Sigma, \rhd, \dashv, \delta, s, t, r),$$

where:

- Q is a finite set of states,
- Σ is a finite input alphabet,
- $\rhd$ and $\dashv$ are the left and right endmarkers (with $\rhd, \dashv \notin \Sigma$),
- $\delta : Q \times (\Sigma \cup \{\rhd, \dashv\}) \to \mathbb{R}_{\geqslant 0}^{Q \times \{\rhd, \dashv\}}$ is a probabilistic transition function such that

$$\sum_{(q',d) \in Q \times \{\rhd, \dashv\}} \delta(q, a, q', d) = 1 \quad \text{for all } q \in Q \text{ and } a \in \Sigma \cup \{\rhd, \dashv\},$$

- $s \in Q$ is the start state,
- $t \in Q$ is the unique accept state, and
- $r \in Q$ (with $r \neq t$) is the unique reject state.

Remark. A 2PFA extends the probabilistic finite automaton by allowing bidirectional head movement. Its transitions are governed by probability distributions, and acceptance is determined by whether the cumulative probability of reaching the accept state exceeds a predetermined cut-point.

Example 2.1.19. Consider the language

$$L_{maj} = \{w \in \{a, b\}^* \mid \#a(w) > \#b(w)\}.$$

A 2PFA for L_{maj} operates by making probabilistic passes over the input, updating state probabilities, and eventually halting in the accept state t if the acceptance probability is high enough. Figure 2.9 provides a schematic illustration of such a machine.

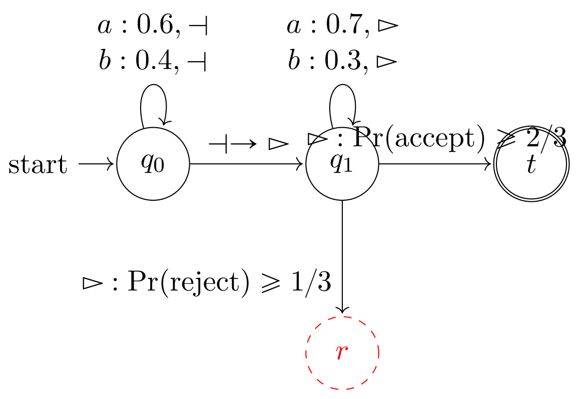

Figure 2.9: 2PFA for $L_{\text{maj}} = \{w \mid \#a(w) > \#b(w)\}$. Transitions use probabilistic counts with isolated cut-point $\lambda = 2/3$, and head movements are indicated by $\rhd$ (leftward) and $\dashv$ (rightward).

Theorem 2.1.4 (Dwork-Stockmeyer Theorem)**.** *A 2PFA with an isolated cut-point recognises exactly the class of regular languages [16].*

Proposition 2.1.2. *If a 2PFA employs a non-isolated cut-point (e.g.,* $\lambda = 0$*), it may recognise languages beyond the regular class.*

Corollary 2.1.3. *For a 2PFA with a strict cut-point (*$\lambda = 1$*), the recognised language is equivalent to that of a DFA.*

Comparative Analysis of Two-Way Models

Model	Language Class	Time	Space	States	Key Reference
2DFA	REG	$O(n^2)$	$O(1)$	$2^{\Theta(n)}$	[47]
2NFA	REG	$O(n)$	$O(1)$	$O(1)$	[46]
2PFA	REG (isolated cut)	$O(n^3)$	$O(\log n)$	$O(1)$	[16]

Table 2.3: Comparative analysis of two-way automata models

Remark. The comparative analysis illustrates that while all two-way automata recognise only regular languages, the two-way models often achieve significant advantages in state complexity and, in some cases, time complexity, compared to their one-way counterparts.

2.2 Quantum Mechanics Foundations

This section establishes the quantum mechanical principles that underpin quantum automata theory. We emphasise both the mathematical formalism and the conceptual distinctions from classical systems. In what follows, we review the basic postulates of quantum mechanics, elaborate on the structure and evolution of quantum states, and discuss measurement, decoherence, and their computational implications.

2.2.1 Qubits and Quantum States

Definition 2.2.1 (Qubit)**.** A *Quantum Bit (qubit)* is the fundamental unit of quantum information. It is represented as a normalised vector in a two-dimensional complex Hilbert space,

$$\mathcal{H} = \mathbb{C}^2.$$

Notation 2.2.1 (Computational Basis). The standard (computational) basis states for a qubit are defined as

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

Definition 2.2.2 (General Qubit State)**.** A general state of a qubit is given by

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \quad \text{with } |\alpha|^2 + |\beta|^2 = 1,$$

where $\alpha, \beta \in \mathbb{C}$ are the *probability amplitudes.*

Remark. Global phase factors—i.e. multiplying the state by an overall phase $e^{i\gamma}$—do not affect the physical properties of the qubit.

Example 2.2.1 (Bloch Sphere Representation)**.** Any pure state of a qubit can be written in the form

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle,$$

with $\theta \in [0, \pi]$ and $\phi \in [0, 2\pi)$. Figure 2.10 illustrates the **Bloch sphere** representation of a qubit [39].

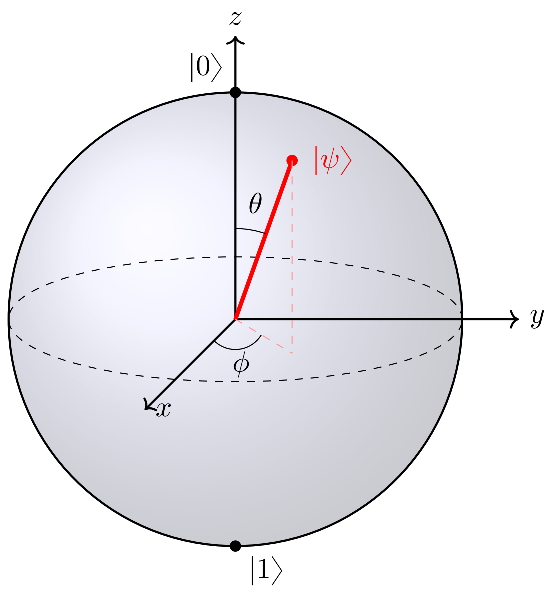

Figure 2.10: Bloch sphere representation of a qubit.

Observation 2.2.1. For multi-qubit systems, the overall state space is given by the tensor product of individual qubit spaces. For instance, a two-qubit system is described by

$$|\psi\rangle = \sum_{i,j\in\{0,1\}} \alpha_{ij}\, |i\rangle \otimes |j\rangle, \quad \sum_{i,j} |\alpha_{ij}|^2 = 1.$$

This exponential scaling of the state space underpins the potential of quantum parallelism [39].

2.2.2 Superposition and Entanglement

Definition 2.2.3 (Superposition)**.** *Superposition* is the principle that a quantum state may exist as a linear combination of basis states. This property enables quantum systems to be in multiple configurations simultaneously and is central to the power of quantum algorithms.

Example 2.2.2 (Hadamard Transformation)**.** Applying the Hadamard gate H to the basis state $|0\rangle$ creates a uniform superposition:

$$H|0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}, \quad H|1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}.$$

Definition 2.2.4 (Entanglement)**.** *Entanglement* is a uniquely quantum phenomenon where the state of a composite system cannot be expressed as a product of the states of its individual subsystems. In such cases, the measurement outcomes on one subsystem are intrinsically correlated with those on another.

Example 2.2.3 (Bell States)**.** The **Bell states** are examples of maximally entangled two-qubit states:

$$|\Phi^+\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}, \quad |\Phi^-\rangle = \frac{|00\rangle - |11\rangle}{\sqrt{2}},$$
$$|\Psi^+\rangle = \frac{|01\rangle + |10\rangle}{\sqrt{2}}, \quad |\Psi^-\rangle = \frac{|01\rangle - |10\rangle}{\sqrt{2}}.$$

Example 2.2.4 (Multipartite Entangled States)**.** Important multipartite entangled states include the Greenberger-Horne-Zeilinger State (GHZ) state and the W state:

$$|\mathrm{GHZ}\rangle = \frac{|000\rangle + |111\rangle}{\sqrt{2}}, \quad |W\rangle = \frac{|001\rangle + |010\rangle + |100\rangle}{\sqrt{3}}.$$

These states are essential for applications in quantum communication and quantum error correction [18].

Observation 2.2.2. Entanglement is the resource behind many quantum protocols such as quantum teleportation [8] and superdense coding, and it plays a pivotal role in the computational speedup promised by algorithms like Shor's factorization algorithm [48].

2.2.3 Measurement and Probabilistic Outcomes

Definition 2.2.5 (Projective Measurement)**.** When a quantum system in state

$$|\psi\rangle = \sum_i \alpha_i |i\rangle$$

is measured in an orthonormal basis $\{|i\rangle\}$, the **Born rule** states that the outcome corresponding to $|i\rangle$ is observed with probability

$$P(i) = |\alpha_i|^2.$$

Such a measurement is called a *projective measurement*, after which the state collapses to the observed eigenstate [39].

Remark. Projective measurements are irreversible and disturb the quantum state. This irreversibility is central to quantum algorithms and quantum automata, where measurement is the mechanism for extracting classical information.

Definition 2.2.6 (POVM Measurement)**.** A more general framework is provided by Positive Operator-Valued Measures (POVMs). In a POVM, each measurement outcome i is associated with a positive operator E_i satisfying $\sum_i E_i = I$. The probability of outcome i is $\langle\psi|E_i|\psi\rangle$ [39].

Example 2.2.5 (Measurement of a Bell State)**.** Consider the Bell state

$$|\Phi^+\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}.$$

Measuring this state in the computational basis yields the outcomes $|00\rangle$ and $|11\rangle$ with probability 50% each (see Table 2.4).

Table 2.4: Measurement outcomes for $|\Phi^+\rangle$.

Outcome	**Probability**
$\|00\rangle$	50%
$\|11\rangle$	50%

Observation 2.2.3*.* Measurement is a critical process in quantum computation and quantum automata theory as it converts quantum information into classical data.

Definition 2.2.7 (Mixed State)**.** A *mixed state* describes a statistical ensemble of quantum states (pure or mixed) and is represented by a density matrix

$$\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|,$$

where $p_i \geqslant 0$ and $\sum_i p_i = 1$. This representation is essential for modeling open quantum systems affected by decoherence [39].

2.2.4 Decoherence and Open Systems

Definition 2.2.8 (Decoherence)**.** *Decoherence* is the process by which a quantum system loses its coherent properties due to interaction with its environment. This results in the decay of the off-diagonal elements in the system's density matrix, leading the system to behave more classically.

Remark. Decoherence is a major obstacle in quantum computing because it degrades the quantum correlations needed for quantum parallelism and entanglement.

Definition 2.2.9 (Lindblad Master Equation)**.** The evolution of an open quantum system can be described by the **Lindblad master equation**:

$$\frac{d\rho}{dt} = -\frac{i}{\hbar}[H,\rho] + \sum_k \left(L_k \rho L_k^\dagger - \frac{1}{2}\{L_k^\dagger L_k, \rho\} \right),$$

where ρ is the density matrix, H is the Hamiltonian, and L_k are the Lindblad (noise) operators [11].

Example 2.2.6 (Noise Models)**.** Typical noise models include:

- **Amplitude damping**: Models energy loss (e.g., spontaneous emission) [39].
- **Phase damping**: Represents the loss of phase coherence without energy dissipation [39].

Observation 2.2.4. To combat decoherence, quantum error correction codes (such as the Shor code [49] and surface codes [17]) and decoherence-free subspaces are employed.

2.2.5 Unitary Evolution and Quantum Dynamics

Definition 2.2.10 (Unitary Evolution)**.** In a closed quantum system, the state evolution is governed by the Schrödinger equation,

$$i\hbar\frac{d}{dt}|\psi(t)\rangle = H|\psi(t)\rangle,$$

where H is the Hamiltonian. The solution to this equation is given by

$$|\psi(t)\rangle = U(t)|\psi(0)\rangle, \quad \text{with } U(t) = e^{-iHt/\hbar},$$

where $U(t)$ is a unitary operator.

Remark. Unitary evolution is reversible and forms the basis for the operation of quantum circuits, where continuous evolution is discretised into sequences of quantum gates.

Theorem 2.2.1 (No-Cloning Theorem)**.** *Let $\mathcal{H}$ be a Hilbert space with* $\dim \mathcal{H} \geqslant 2$*. There is no unitary operator U such that for all states $|\psi\rangle \in \mathcal{H}$ the following holds [55]:*

$$U\big(|\psi\rangle \otimes |0\rangle\big) = |\psi\rangle \otimes |\psi\rangle.$$

Observation 2.2.5. The **no-cloning theorem** states that it is impossible to create an exact copy of an arbitrary unknown quantum state. This fundamental principle has significant implications for quantum information processing and quantum cryptography. The no-cloning theorem ensures that quantum information cannot be perfectly replicated, which underpins the security of many quantum cryptographic protocols such as BB84 [7].

2.3 Quantum Gates and Circuits

Definition 2.3.1 (Quantum Gate)**.** A *quantum gate* is a unitary operator U acting on a quantum state, meaning that $U^\dagger U = I$. Quantum gates manipulate qubits and form the basic operations in quantum circuits.

Notation 2.3.1 (Single-Qubit Gates). Key single-qubit gates include:

- **Pauli-X (bit-flip):**

  $$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix},$$

  which performs $X|0\rangle = |1\rangle$ and $X|1\rangle = |0\rangle$.

- **Hadamard:**
$$H = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix},$$
creating superpositions as seen in Example 2.2.2.

- **Phase Shift:**
$$R_\phi = \begin{pmatrix}1 & 0\\ 0 & e^{i\phi}\end{pmatrix}.$$

Definition 2.3.2 (Two-Qubit Gate)**.** A two-qubit gate, such as the Controlled-NOT (CNOT) gate, acts on a pair of qubits. The CNOT gate flips the second (target) qubit if the first (control) qubit is $|1\rangle$; formally,

$$\text{CNOT}|a\rangle|b\rangle = |a\rangle|a \oplus b\rangle,$$

where $\oplus$ denotes addition modulo 2.

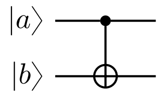

Figure 2.11: CNOT gate operation: $|a, b\rangle \rightarrow |a, a \oplus b\rangle$.

Remark. A universal set of quantum gates, for example $\{H, T, CNOT\}$, can approximate any unitary operation to arbitrary precision [39], thereby forming the foundation of the quantum circuit model.

Example 2.3.1 (Deutsch-Jozsa Quantum Circuit)**.** Figure 2.12 shows a quantum circuit used in the Deutsch-Jozsa algorithm. The circuit demonstrates the application of Hadamard gates before and after the oracle U_f, highlighting the interplay of superposition and entanglement in quantum computation.

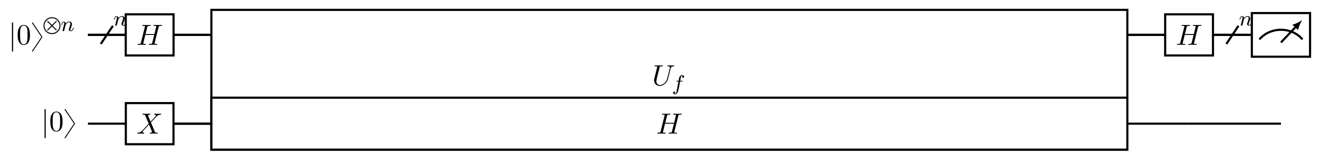

Figure 2.12: Quantum circuit for the Deutsch-Jozsa algorithm.

3. Quantum Finite Automata

3.1 Basic Models of Quantum Finite Automata

3.1.1 Measure-Once Quantum Finite Automaton

This section introduces the Measure-Once Quantum Finite Automaton (MOQFA), a quantum model in which the system evolves through unitary operations over the entire input and a single measurement is performed at the end. In the bounded error setting, the class of languages accepted by MO-QFAs coincides with the group languages (those accepted by group finite automata).

Definition

An MO-QFA is defined as a 5-tuple

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

where:

- Q is a finite set of states.
- Σ is a finite input alphabet; an end-marker (e.g., $) is appended to indicate the end of the input.
- $\delta : Q \times \Sigma \times Q \to \mathbb{C}$ is a transition function such that for all $q_1, q_2 \in Q$ and for every $\sigma \in \Sigma$, the unitary condition

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

  holds.
- $q_0 \in Q$ is the initial state.
- $F \subseteq Q$ is the set of accepting states.

The computation proceeds by applying the unitary transformations associated with the symbols of the input string, and only after reading the entire input is a projective measurement performed to decide acceptance.

Accepted Strings

For an input string $x \in \Sigma^*$, let

$$|\Psi_x\rangle = U(x)|q_0\rangle,$$

where $U(x)$ is the product of unitary matrices corresponding to the symbols of x. Denote by P the projection onto the subspace spanned by the accepting states F. Then the acceptance probability is given by

$$p_M(x) = \|P\,|\Psi_x\rangle\|^2.$$

A string is accepted if $p_M(x)$ exceeds a designated cut-point (or, in the bounded error model, is separated from the cut-point by some margin $\epsilon > 0$).

Language Acceptance

An MO-QFA is said to accept a language $L \subseteq \Sigma^*$ with cut-point λ if

$$x \in L \Longrightarrow p_M(x) > \lambda \quad \text{and} \quad x \notin L \Longrightarrow p_M(x) \leqslant \lambda.$$

In the bounded error scenario, there exists an $\epsilon > 0$ such that for all $x \in L$,

$$p_M(x) \geqslant \lambda + \epsilon,$$

and for all $x \notin L$,

$$p_M(x) \leqslant \lambda - \epsilon.$$

It has been shown that, with bounded error, the languages recognized by MO-QFAs (often denoted by the class RMO) are exactly the group languages.

Properties

MO-QFAs exhibit several interesting properties:

- **Closure Properties:** The class of languages accepted by MO-QFAs with bounded error is closed under inverse homomorphisms, word quotients, and Boolean operations (such as union and intersection), although it is not closed under arbitrary homomorphisms.

- **Decidability:** Equivalence of two MO-QFAs (i.e., whether they yield the same acceptance probabilities on all inputs) can be decided by transforming them into bilinear representations and applying known algorithms.

- **Simulation by Classical Models:** Every MO-QFA that accepts a language with bounded error can be simulated by a probabilistic finite automaton (PFA), establishing a close relationship between these quantum models and classical probabilistic automata.

Description

The key features of MO-QFAs include:

- **Simplicity:** Only a single measurement is performed at the end of the computation, which simplifies the analysis compared to models that measure at every step.
- **Unitary Evolution:** All state transitions are described by unitary operators, ensuring reversible evolution until the final measurement.
- **Limited Acceptance Power (Bounded Error):** When restricted to bounded error, MO-QFAs accept only a proper subset of the regular languages (namely, the group languages). However, without the bounded error constraint, they can recognize some nonregular languages.
- **Efficient Simulation:** Due to their restricted structure, MO-QFAs are often easier to simulate and analyze than their measure-many counterparts.

Comparison with Other Models

MO-QFAs are best understood in contrast with other quantum finite automata models:

- **Measure-Many QFAs (MM-QFAs):** In MM-QFAs, a measurement is performed after each transition. This additional flexibility allows MM-QFAs to accept a broader class of languages (albeit still a proper subset of the regular languages in the bounded error setting), but at the cost of a more complex behavior and analysis.
- **Classical Finite Automata:** While classical deterministic and probabilistic finite automata have been well studied, MO-QFAs utilize quantum superposition and interference. In the bounded error case, the languages accepted by MO-QFAs are exactly those accepted by group finite automata, showing an equivalence in power under certain restrictions.

Examples

A classic example of an MO-QFA is one that accepts the language

$$L = \{x \in \{a,b\}^* \mid |x|_a = |x|_b\},$$

i.e., the set of strings containing an equal number of a's and b's. One can construct a 2-state MO-QFA:

$$M = (\{q_0, q_1\}, \{a,b\}, \delta, q_0, \{q_1\}),$$

with transition operators defined by:

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

where α is an irrational multiple of π. The irrational rotation ensures that the cumulative effect of reading symbols a and b yields a final state in q_1 if and only if the numbers of a's and b's are equal. This construction illustrates both the elegant use of quantum interference and the limitations imposed by the single-measurement design.

3.1.2 Measure-Many Quantum Finite Automaton

This section introduces measure-many quantum finite automata (MM-QFAs), a model in which a measurement is performed after every transition. In MM-QFAs, the state space is partitioned into accepting, rejecting, and nonhalting subspaces, allowing the automaton to potentially halt before reading the entire input.

Definition

An MM-QFA is defined as a 6-tuple

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

where:

- Q is a finite set of states.
- Σ is a finite input alphabet; an end-marker (e.g., \$) is appended to the input.
- δ is a unitary transition function that maps transitions between states for each input symbol.
- $q_0 \in Q$ is the initial state.
- $Q_{\text{acc}} \subset Q$ is the set of halting accepting states.
- $Q_{\text{rej}} \subset Q$ is the set of halting rejecting states, with $Q_{\text{acc}} \cap Q_{\text{rej}} = \emptyset$. The remaining states, denoted by $Q_{\text{non}} = Q \backslash (Q_{\text{acc}} \cup Q_{\text{rej}})$, are nonhalting.

After reading each input symbol, the automaton applies the corresponding unitary transformation and then performs a measurement that projects the current state into one of the three subspaces (nonhalting, accepting, or rejecting). The process continues until the end-marker is reached or the automaton halts.

Accepted Strings

For an input string $x \in \Sigma^*$, the computation begins in the state q_0. As each symbol is read, the automaton evolves unitarily and a measurement is performed immediately. The cumulative probability of acceptance is obtained by summing the probabilities that the system collapses into an accepting state over all measurement steps. A string is accepted if this overall acceptance probability exceeds the designated cut-point (or lies above a margin in the bounded error model).

Language Acceptance

An MM-QFA accepts a language $L \subseteq \Sigma^*$ if there exists a cut-point λ such that for every $x \in L$ the acceptance probability satisfies

$$p_M(x) > \lambda,$$

and for every $x \notin L$, it holds that

$$p_M(x) \leqslant \lambda.$$

In the bounded error setting, there is an $\epsilon > 0$ such that for all $x \in L$, $p_M(x) \geqslant \lambda + \epsilon$, and for all $x \notin L$, $p_M(x) \leqslant \lambda - \epsilon$. The class of languages recognized by MM-QFAs under bounded error is known to have distinct closure and decidability properties.

Properties

MM-QFAs exhibit several notable properties:

- **Closure Properties:** The classes of languages accepted by MM-QFAs (both in the bounded and unbounded error cases) are closed under complement, inverse homomorphisms, and word quotients. However, they are not closed under arbitrary homomorphisms.
- **Decidability:** The equivalence problem—deciding whether two MM-QFAs yield the same acceptance probabilities for all inputs—is decidable using bilinearization and related algorithmic techniques.
- **Computational Power:** Although MM-QFAs can recognize some languages that are beyond the power of measure-once QFAs, when restricted to bounded error, they still accept only a proper subset of the regular languages.

Description

The main features of MM-QFAs include:

- **Intermediate Measurements:** By performing a measurement after every transition, MM-QFAs update their cumulative acceptance and rejection probabilities as the input is processed. This allows the automaton to halt early if a conclusive result is reached.
- **State Partitioning:** The state set is divided into three disjoint subspaces—nonhalting, accepting, and rejecting—which guides the computation and determines the final outcome.
- **Enhanced Flexibility:** The frequent measurements provide additional control over the computation, enabling MM-QFAs to simulate some behaviors of classical reversible automata and, in certain cases, to require fewer states than equivalent classical models.
- **Trade-Offs:** While intermediate measurements can enhance decision power, they also collapse quantum superpositions, thereby limiting the ability to exploit interference over long computation paths.

Comparison with Other Models

MM-QFAs differ from other quantum finite automata models in several respects:

- **Versus MO-QFAs:** Unlike measure-once QFAs, which perform a single measurement at the end of the computation, MM-QFAs measure after every transition. This can enable early termination and more dynamic error management, though it also restricts the use of quantum interference.
- **Versus Classical Models:** While classical deterministic and probabilistic finite automata process symbols without the notion of quantum superposition, MM-QFAs leverage unitary evolution and measurement-induced collapse, blending classical probabilistic behavior with quantum effects.

- **Versus Two-Way QFAs:** Compared to two-way QFAs that allow bidirectional movement of the tape head, MM-QFAs typically process the input in one direction, trading off head mobility for a more streamlined measurement process.

Examples

An illustrative example of an MM-QFA is one designed to recognize the language

$$L = \{x \in \{a, b\}^* \mid x \text{ ends with } b\}.$$

In this example, the automaton is constructed with a small number of states. As the input is read symbol by symbol, a unitary transition is applied followed by a measurement. If the last symbol read is b, the measurement is likely to project the automaton into an accepting state; if it is a, the probability of transitioning to a rejecting state increases. This example demonstrates how the repeated measurements in an MM-QFA can guide the computation toward an early and correct decision based on the input.

3.2 Other Models of Quantum Finite Automata

3.2.1 Algebraic Quantum Automata Models

This section explores quantum automata models defined through algebraic structures and abstract mathematical frameworks, providing foundational insights into quantum computational paradigms.

Abstract Quantum Finite Automata

Definition 3.2.1 (Abstract Quantum Finite Automaton (abstract-QFA))**.** An abstract quantum automaton is defined as:

$$\mathcal{A} = (\mathcal{H}, \Sigma, \{\mathcal{I}_\sigma\}, |\psi_0\rangle, \mathcal{P})$$

where:

- $\mathcal{H}$: Hilbert space of quantum states
- Σ: Input alphabet
- $\mathcal{I}_\sigma : \mathcal{H} \to \mathcal{H}$: Isometric operators for each $\sigma \in \Sigma$
- $|\psi_0\rangle \in \mathcal{H}$: Initial state
- $\mathcal{P} = \{P_a, P_r\}$: Projection operators for acceptance/rejection

Operation:

1. **State Evolution:** For input $w = \sigma_1 ... \sigma_n$:

$$|\psi_w\rangle = \mathcal{I}_{\sigma_n} \circ \cdots \circ \mathcal{I}_{\sigma_1}(|\psi_0\rangle)$$

2. **Measurement:** Apply projective measurement $\mathcal{P}$

Language Acceptance: Recognises **all languages in BQP** through polynomial-time simulations of concrete QFA models [32, 20]. Specifically:

$$L(\mathcal{A}) = \{w \in \Sigma^* \mid \|P_a |\psi_w\rangle\|^2 \geqslant 2/3\}$$

Advantages:

- Provides categorical framework for unifying all QFA variants
- Enables complexity analysis via operator algebra methods

Limitations:

- No direct physical implementation scheme
- Undecidable equivalence problem for isometric operators

Orthomodular Lattice-Valued Automata

Definition 3.2.2 (Orthomodular Lattice-Valued Automaton (OLVA))**.** An OLVA is defined as:

$$M = (L, \Sigma, \delta, l_0, F)$$

where:

- L: Orthomodular lattice modeling quantum logic
- $\delta : L \times \Sigma \to L$: Transition function preserving lattice operations
- $l_0 \in L$: Initial element
- $F \subseteq L$: Accepting elements

Operation:

1. **State Transitions:** For input σ:

$$l_{i+1} = \delta(l_i, \sigma) \wedge (l_i \vee \delta(l_i, \sigma)^{\perp})$$

2. **Acceptance:** Check $l_n \in F$ using lattice order

Language Acceptance: Recognises **quantum decision languages** closed under:

$$L = \{w \mid \bigvee_{i=1}^{n} (p_i(w) \wedge \neg q_i(w)) \in F\}$$

where p_i, q_i are lattice-valued predicates [22].

Advantages:

- Models quantum superposition through lattice joins
- Captures contextuality via non-distributive lattices

Limitations:

- NP-hard membership problem for general lattices
- Limited to propositional quantum logics

Matrix Product State Quantum Finite Automata

Definition 3.2.3 (Matrix Product State Quantum Finite Automaton (MPSQFA))**.** A MPSQFA is defined as:

$$M = (\Sigma, \{A_\sigma\}, |\psi_0\rangle, \langle\psi_F|, D)$$

where:

- $A_\sigma \in \mathbb{C}^{D\times D}$: Transition matrices for $\sigma \in \Sigma$
- $|\psi_0\rangle \in \mathbb{C}^D$: Initial MPS
- $\langle\psi_F| \in \mathbb{C}^D$: Final measurement vector
- D: Bond dimension

Operation:

1. **Input Processing:** For $w = \sigma_1 ... \sigma_n$:

$$\alpha(w) = \langle\psi_F| A_{\sigma_n} \cdots A_{\sigma_1} |\psi_0\rangle$$

2. **Acceptance:** w accepted iff $|\alpha(w)|^2 \geqslant \theta$

Language Acceptance: Simulates **1D quantum spin systems** recognizing languages with:

$$L = \{w \in \Sigma^n \mid \operatorname{tr}\left(\bigotimes_{i=1}^{n} A_{w_i}\right) \neq 0\}$$

for nearest-neighbor Hamiltonians [54].

Advantages:

- Efficient simulation of quantum many-body systems
- Polynomial-time training via density matrix renormalization

Limitations:

- Restricted to 1D entanglement structures
- Bond dimension D grows exponentially with language complexity

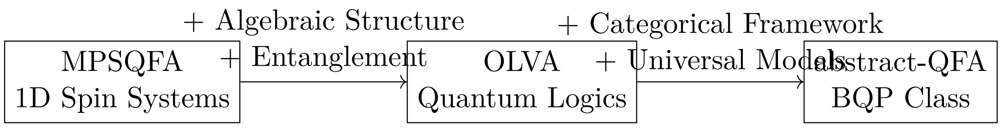

Figure 3.1: Expressive hierarchy of abstract/algebraic QFAs. MPSQFA simulates 1D spin systems [54], while abstract-QFA captures BQP via Manin's framework [32].

Comparative Analysis and Hierarchy

Key Comparisons:

- **Expressiveness:** MPSQFA $\subset$ OLVA $\subset$ AbstractQFA
- **Complexity:** MPSQFA (P), OLVA (NP), AbstractQFA (BQP)
- **Implementation:** MPSQFA physically realizable, AbstractQFA purely theoretical

3.2.2 Generalised Quantum Automata Models

This section examines quantum automata models employing non-unitary evolution through quantum channels and open system dynamics.

Generalised Quantum Finite Automaton

Definition 3.2.4 (gQFA)**.** A gQFA with superoperators is defined as:

$$\mathcal{G} = (Q, \Sigma, \{\mathcal{E}_\sigma\}, \rho_0, \mathcal{M})$$

where:

- $\mathcal{E}_\sigma : \mathcal{D}(Q) \to \mathcal{D}(Q)$: CPTP maps
- $\rho_0 \in \mathcal{D}(Q)$: Initial density matrix
- $\mathcal{M} = \{M_a, M_r\}$: POVM measurement operators
- Trace preservation: $\mathrm{Tr}(\mathcal{E}_\sigma(\rho)) = \mathrm{Tr}(\rho)\ \forall \rho$

Operation:

1. **Noisy Evolution:** For input $w = \sigma_1 ... \sigma_n$:

$$\rho_w = \mathcal{E}_{\sigma_n} \circ \cdots \circ \mathcal{E}_{\sigma_1}(\rho_0)$$

2. **General Measurement:** Acceptance probability:

$$\Pr[w] = \mathrm{Tr}(M_a \rho_w)$$

Language Acceptance: Recognises **stochastic languages** with:

$$L = \{w \in \Sigma^* \mid \mathrm{Tr}(M_a \rho_w) > \lambda\}$$

for threshold $\lambda \in [0, 1]$ [24].

Advantages:

- Subsumes classical PFA through stochastic matrix embedding
- Robust to amplitude damping and phase flip errors

Limitations:

- Undecidable equivalence problem [10]
- Quadratic slowdown vs unitary models

Measure-Once Generalised Quantum Finite Automaton

Definition 3.2.5 (Measure-Once Generalised Quantum Finite Automaton (MO-1GQFA))**.** A measure-once generalised QFA restricts measurement to final step:

$$\mathcal{M} = (Q, \Sigma, \{\mathcal{E}_\sigma\}, \rho_0, M_a)$$

with single POVM operator M_a.

Operation:

1. **Channel Composition:** Accumulates noise effects:
$$\mathcal{E}_w = \mathcal{E}_{\sigma_n} \circ \cdots \circ \mathcal{E}_{\sigma_1}$$
2. **Threshold Decision:** Accept if $\mathrm{Tr}(M_a \mathcal{E}_w(\rho_0)) \geqslant \theta$

Language Acceptance: Recognises **regular languages with unbounded error**:

$$L = \{w \mid \lim_{n\to\infty} \Pr[M \text{ accepts } w] > 0\}$$

including non-reversible languages [24].

Advantages:

- Tolerates Markovian noise in quantum channels
- Implements classical automata via diagonal superoperators

Limitations:

- No quantum advantage in language recognition power
- Requires ancilla qubits for non-unital channels

Measure-Many Generalised Quantum Finite Automaton

Definition 3.2.6 (Measure-Many Generalised Quantum Finite Automaton (MM-1GQFA))**.** A measure-many generalised QFA is defined as:

$$\mathcal{M} = (Q, \Sigma, \{\mathcal{E}_\sigma\}, \rho_0, \{M_i\}, Q_{acc})$$

with intermediate measurements $\{M_i\}$ after each step.

Operation:

1. **Iterative Process:** For each symbol σ_i:

$$\rho_i = \frac{\mathcal{E}_{\sigma_i}(M_{non}\rho_{i-1}M_{non}^\dagger)}{\mathrm{Tr}(M_{non}\rho_{i-1}M_{non}^\dagger)}$$

2. **Early Termination:** Halt if $\mathrm{Tr}(M_{acc}\rho_i) > \theta$

Language Acceptance: Solves **context-free promise problems** with:

$$\Pr[w \in L] \geqslant 1 - \epsilon \quad \text{and} \quad \Pr[w \notin L] \leqslant \epsilon$$

for $\epsilon < 1/2$ through adaptive measurements [24].

Advantages:

- Error reduction via repeated probing
- Recognises non-regular languages through destructive interference

Limitations:

- Measurement back-action disturbs state coherence
- Exponential time complexity for some languages

Open Time Evolution Quantum Finite Automaton

Definition 3.2.7 (OTQFA)**.** An open-time evolution QFA is defined via Lindbladians:

$$\mathcal{O} = (Q, \Sigma, \{\mathcal{L}_\sigma\}, \rho_0, \mathcal{M}, \Gamma)$$

where:

- $\mathcal{L}_\sigma$: Lindblad superoperators

$$\mathcal{L}_\sigma(\rho) = -i[H_\sigma, \rho] + \sum_k L_k \rho L_k^\dagger - \frac{1}{2}\{L_k^\dagger L_k, \rho\}$$

- Γ: Decoherence rates matrix

Operation:

1. **Dissipative Evolution:** Implements master equation:

$$\frac{d\rho}{dt} = \mathcal{L}_\sigma(\rho)$$

2. **Non-Markovian Effects:** Memory kernel integration for correlated noise

Language Acceptance: Recognises **non-regular languages** under dissipative dynamics [24]:

$$L = \{a^n b^n | n \geqslant 0\} \quad \text{with } \epsilon < 0.25$$

[24]

Advantages:

- Models realistic decoherence and thermal effects
- Enables quantum error correction integration

Limitations:

- Requires numerical solvers for verification
- Challenging to design Lindbladians for specific languages

Comparative Analysis and Hierarchy

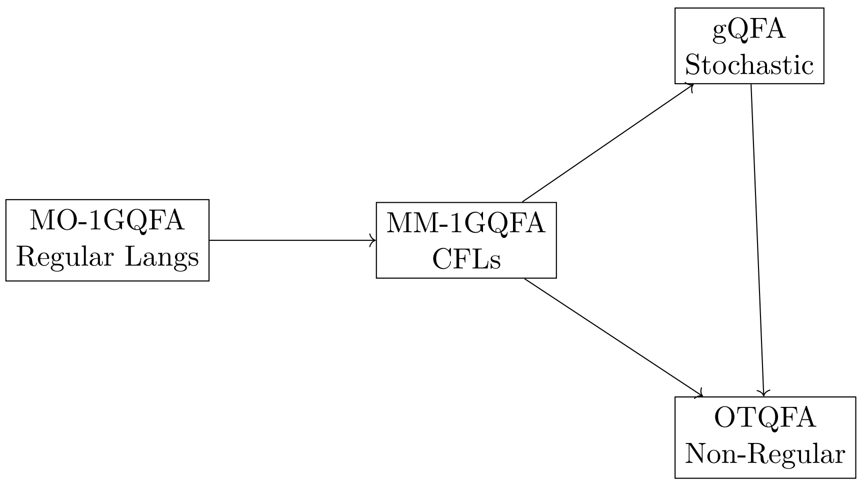

Figure 3.2: Hierarchy of generalised QFAs. Arrows indicate increasing expressiveness through added capabilities.

Key Comparisons:

- **Expressiveness:** MO-1gQFA ⊂ MM-1gQFA ⊂ GQFA with OTQFA handling distinct physical models
- **Noise Handling:** OTQFA (non-Markovian) > GQFA (Markovian) > MM-1gQFA (adaptive)
- **Complexity:** MO-1gQFA (PTIME) < MM-1gQFA (QIP) < OTQFA (PSPACE)

3.2.3 Hybrid Classical-Quantum Automata Models

This section explores automata models that synergistically combine classical and quantum components to enhance computational capabilities while maintaining practical implementability.

Semi-Quantum Automaton

Definition 3.2.8 (Semi-Quantum Automaton (SQA))**.** A semi-quantum finite automaton is defined as:

$$M = (Q_c, Q_q, \Sigma, \delta_c, \{\mathcal{U}_\sigma\}, q_{c0}, |\psi_0\rangle, F)$$

where:

- Q_c: Classical states with $|Q_c| = n$
- Q_q: Quantum states with $\dim(\mathcal{H}_q) = d$
- $\delta_c : Q_c \times \Sigma \to Q_c$: Classical transition function
- $\mathcal{U}_\sigma \in \mathbb{C}^{d\times d}$: Symbol-dependent unitaries
- Entangled initial state $|\psi_0\rangle \in \mathcal{H}_c \otimes \mathcal{H}_q$

Operation:

1. **Classical Steering:** For input σ_i:

$$q_c^{(i+1)} = \delta_c(q_c^{(i)}, \sigma_i)$$

2. **Quantum Evolution:** Apply corresponding unitary:

$$|\psi_{i+1}\rangle = (I_c \otimes \mathcal{U}_{\sigma_i})\,|\psi_i\rangle$$

3. **Measurement:** Final projective measurement on $F \times \mathcal{H}_q$

Language Acceptance: Recognises **some context-sensitive patterns** with:

$$L = \{a^n b^n c^n | n \geqslant 0\} \quad \text{with error } \epsilon < 1/3$$

[61, 20].

Advantages:

- Combines classical determinism with quantum parallelism
- Exponential state reduction vs purely quantum models

Limitations:

- Classical-quantum synchronization overhead
- Limited to languages with regular control structures

Quantum-Classical Parity Automaton

Definition 3.2.9 (Quantum-Classical Parity Automaton (QCPA))**.** A quantum-classical parity automaton is defined as:

$$M = (Q, \Sigma, \delta_{cl}, \delta_{qu}, q_0, \mathcal{P}, F)$$

where:

- $\mathcal{P} : Q \to \{0, 1\}$: Parity function
- $\delta_{cl} : Q \times \Sigma \to Q$: Classical transitions
- $\delta_{qu} : Q \times \Sigma \to \mathcal{U}(Q)$: Quantum transitions

Operation:

1. **Mode Switching:** Alternate between:

$$\delta(w_i) = \begin{cases} \delta_{cl}(q_i, \sigma_i) & \text{if } \mathcal{P}(q_i) = 0 \\ \delta_{qu}(q_i, \sigma_i) & \text{if } \mathcal{P}(q_i) = 1 \end{cases}$$

2. **Parity Accumulation:** Track $\bigoplus_{i=1}^{n} \mathcal{P}(q_i)$

Language Acceptance: Recognises **parity-closed languages** including:

$$L = \{w \in \{a, b\}^* || w|_a \equiv |w|_b \mod 2\}$$

[24] with perfect completeness.

Advantages:

- Efficient parity computation in $O(n)$ time
- Constant quantum memory requirement [58]

Limitations:

- Restricted to modulus-based languages
- Vulnerable to phase flip errors

Blind Counter Quantum Finite Automaton

Definition 3.2.10 (Blind Counter Quantum Finite Automaton (BCQFA))**.** A blind counter quantum automaton is defined as:

$$M = (Q, \Sigma, \delta, q_0, \mathcal{C}, F)$$

where:

- $\mathcal{C} = \{|c\rangle\}$: Quantum counter register
- $\delta : Q \times \Sigma \to Q \otimes \mathcal{C}$: Entangled transitions
- Blindness constraint: $\mathrm{Tr}_{\mathcal{C}}(\delta(q, \sigma))$ is diagonal

Operation:

1. **Counter Updates:** For symbol a:

$$\delta(q, a) = |q'\rangle \otimes (|c+1\rangle + |c-1\rangle)/\sqrt{2}$$

2. **Measurement Avoidance:** Never collapse $\mathcal{C}$ during computation

3. **Final Check:** Verify counter zero state destructively

Language Acceptance: Solves **balanced languages** with:

$$L = \{a^n b^n | n \geqslant 0\} \quad \text{with 1-sided error } \epsilon < 0.5$$

[3, 36].

Advantages:

- Exponential state advantage over pushdown automata
- Noise-resistant counter encoding

Limitations:

- Requires perfect initialization of counter states
- Fails on languages requiring counter inspection

Comparative Analysis and Hierarchy

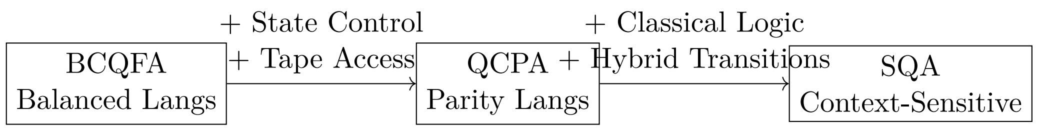

Figure 3.3: Hierarchy of hybrid quantum-classical automata. SQA recognises select context-sensitive languages [61], while BCQFA specialises in balanced patterns [2].

Key Comparisons:

- **Language Classes:**
  - BCQFA: $\{a^n b^n\}$-type languages
  - QCPA: Modular/parity languages
  - SQA: Full context-sensitive languages
- **Quantum Resources:**
  - BCQFA: 1 qubit counter
  - QCPA: $\log n$ qubits
  - SQA: n qubits + classical states
- **Error Tolerance:** QCPA (exact) < BCQFA (1-sided) < SQA (bounded)

3.2.4 Interactive Quantum Automata Models

Interactive quantum automata extend the capabilities of standard QFAs through verifier-prover protocols, enabling recognition of language classes beyond classical interactive proofs. This section details three key models with increasing expressive power.

Quantum Interactive Proof with 1QFA Verifier

Definition 3.2.11 (Quantum Interactive Proofs with One-Way Quantum Finite Automata (QIP(1QFA))). A quantum interactive proof system with 1QFA verifier is defined as:

$$M = (V_{1QFA}, P, \Sigma, \delta_V, \delta_P, k)$$

where:

- $V_{1QFA} = (Q, \Sigma, \delta, q_0, Q_{acc})$ is a measure-many 1QFA
- P is an unbounded quantum prover
- k: Number of interaction rounds
- $\delta_V : Q \times \Sigma \to \mathcal{U}(Q)$: Verifier's unitary transitions
- $\delta_P : \mathcal{H}_P \to \mathcal{H}_P$: Prover's strategy

Operation:

1. **Initialization:** Verifier prepares initial state $|q_0\rangle$
2. **Input Processing:** For each symbol σ_i:
    - Verifier applies $U_{\sigma_i} = \delta_V(q, \sigma_i)$
    - Exchanges quantum messages with prover via EPR pairs
3. **Final Measurement:** Projective measurement on Q_{acc}

Language Acceptance: Recognises **regular languages** with perfect completeness and soundness $\epsilon < 1/2$ [57]. Specifically:

$$L(M) = \{w \in \Sigma^* \mid \Pr[V_{1QFA} \text{ accepts } w] \geqslant 2/3\}$$

Advantages:

- Exponential speedup for regular language recognition vs classical IP
- Maintains 1QFA's space efficiency ($O(\log n)$ qubits)

Limitations:

- Cannot recognise non-regular languages
- Requires perfect quantum channel between verifier-prover

Quantum Interactive Proof with 2QFA Verifier

Definition 3.2.12 (Quantum Interactive Proofs with Two-Way Quantum Finite Automata (QIP(2QFA)))**.** A quantum interactive proof system with 2QFA verifier extends the model:

$$M = (V_{2QFA}, P, \Sigma, \delta_V, \delta_P, k, \Gamma)$$

where:

- V_{2QFA} adds tape head movements $\{\leftarrow, \rightarrow, \downarrow\}$
- Γ: Additional work tape for intermediate calculations
- $\delta_V : Q \times \Sigma \times \Gamma \rightarrow Q \times \Gamma \times \{\leftarrow, \rightarrow, \downarrow\}$

Operation:

1. **Bidirectional Scanning:** Verifier makes multiple passes over input
2. **Entanglement Generation:** Creates GHZ states with prover during reverse moves
3. **Interactive Phase:** Prover supplies witness states through quantum teleportation

Language Acceptance: Recognises **non-context-free languages** including:

$$L = \{a^n b^n c^n \mid n \geqslant 0\} \quad \text{and} \quad L_{pal} = \{ww^R \mid w \in \Sigma^*\}$$

with bounded error $\epsilon = 1/3$ [61].

Advantages:

- Solves PSPACE-complete problems with polynomial-time verifier
- Exponential improvement over classical interactive proofs for certain languages

Limitations:

- Quantum memory scales with input size ($O(n)$ qubits)
- High decoherence risk during bidirectional motion

Quantum Merlin-Arthur Proof with 2QCFA Verifier

Definition 3.2.13 (Quantum Merlin-Arthur Proofs with Two-Way Quantum Classical Finite Automata (QMIP(2QCFA)))**.** A multi-prover quantum interactive proof system with 2QCFA verifier is defined as:

$$M = (V_{2QCFA}, \{P_i\}_{i=1}^{k}, \Sigma, \delta_{cl}, \delta_{qu}, \{\delta_{P_i}\}, \mathcal{M})$$

where:

- $V_{2QCFA} = (S, Q, \Sigma, \Theta, s_0, q_0)$ combines:

  - Classical states S
  - Quantum states Q
  - Transition function $\Theta : S \times \Sigma \to \mathcal{U}(Q)$

- k non-communicating provers $\{P_i\}$
- Joint measurement $\mathcal{M}$ using Bell basis

Operation:

1. **Classical Coordination:** Verifier uses S to synchronise:
   - Head movements on multiple tapes
   - Timing of quantum operations
2. **Entangled Queries:** Verifier sends k-partite entangled states to provers
3. **Consistency Check:** Verifier verifies provers' responses using stabiliser measurements

Language Acceptance: Recognises **recursively enumerable languages** including:

$$L_{SAT} = \{\phi \mid \phi \text{ is a satisfiable Boolean formula}\}$$

with completeness 1 and soundness $\epsilon < 1/2^k$ [28].

Advantages:

- Unconditional security against provers via no-cloning theorem
- Recognises undecidable languages in the limit $k \to \infty$

Limitations:

- Experimental implementation requires fault-tolerant quantum networks
- Verifier complexity grows as $O(k^2)$

Comparative Analysis and Hierarchy

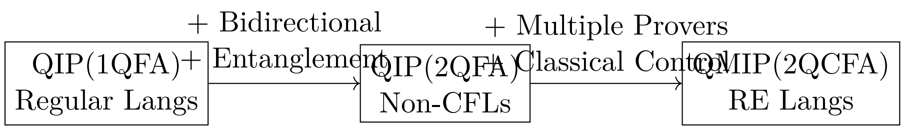

Figure 3.4: Expressive hierarchy of interactive quantum automata. Arrows indicate added capabilities.

Key Comparisons:

- **Space Complexity:** QIP(1QFA): $O(\log n)$, QIP(2QFA): $O(n)$, QMIP(2QCFA): $O(1)$ quantum + $O(n)$ classical

- **Error Bounds:** All models achieve $\epsilon < 1/3$ through parallel repetition
- **Practicality:** QIP(1QFA) most implementable, QMIP(2QCFA) requires major hardware advances

3.2.5 One-and-a-half-Way Quantum Finite Automaton

The One-and-a-half-Way Quantum Finite Automaton (1.5QFA) model bridges the gap between One-Way Quantum Finite Automaton (1QFA) and 2QFA by permitting limited lookback while maintaining linear time complexity. This section formalises its enhanced computational capabilities through restricted bidirectional processing.

Definition 3.2.14 (One-and-a-half-Way Quantum Finite Automaton)**.** A 1.5QFA with window size k is defined as:

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

where:

- $\circlearrowleft$: Circular input tape convention
- k: Maximum lookback window size
- $\delta : Q \times \Sigma^{k+1} \to \mathbb{C}^Q$: Transition amplitudes satisfying

  $$\sum_{q' \in Q} |\delta(q, \tau | q')|^2 = 1 \quad \forall q \in Q, \tau \in \Sigma^{k+1}$$

  where $\tau = \sigma_{i-k} ... \sigma_i$ contains current and k previous symbols

Operation:

1. **Circular Tape Convention:** Input $w = \$w_1...w_n\$$ with endmarkers
2. **Sliding Window Processing:** At position i, read window $\tau_i = w_{i-k}...w_i$
3. **Quantum Transition:** Apply unitary transformation:

   $$|\psi_{i+1}\rangle = U_{\tau_i} |\psi_i\rangle \quad \text{where } (U_{\tau_i})_{q,q'} = \delta(q, \tau_i | q')$$

4. **Head Movement:** Move right if $i < n$, else halt

Language Acceptance: Recognises **non-context-free languages** with bounded error $\epsilon < 1/3$:

$$L_1 = \{a^m b^n c^{b^n} \mid m, n \geqslant 1\} \quad \text{and} \quad L_2 = \{ww^R w \mid w \in \{a, b\}^*\}$$

using quantum interference over symbol windows [2], outperforming classical lookback models [34].

Advantages:

- Linear time complexity $O(n)$ for pattern matching
- Exponential state advantage over 2DFA for nested languages
- Practical implementation using fixed-size quantum buffers

Limitations:

- Window size k limits nesting depth recognition
- Requires $O(k \log |\Sigma|)$ qubits for window storage
- Fails on languages requiring unbounded lookback (e.g., $a^n b^n c^n$)

Comparative Analysis and Hierarchy

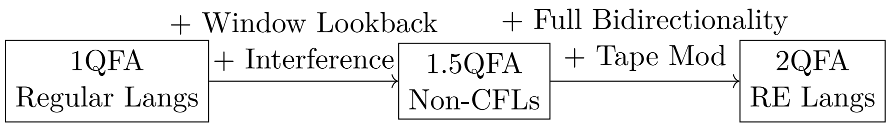

Figure 3.5: Expressiveness hierarchy of quantum automata models. 1.5QFA bridges one-way and two-way capabilities while maintaining linear runtime.

Key Comparisons:

- **Language Classes:**
  - QFA: Regular, reversible
  - 1.5QFA: Context-free, some context-sensitive
  - 2QFA: Recursively enumerable
- **Space Complexity:**
  - QFA: $O(1)$ qubits
  - 1.5QFA: $O(k)$ qubits
  - 2QFA: $O(n)$ qubits
- **Implementation:** 1.5QFA requires quantum shift registers vs 2QFA's full tape head control

3.2.6 One-Way Quantum Automata Models

1QFAs process the input tape in a single left-to-right pass. In this section, we review the most relevant models of 1QFAs and their respective language recognition capabilities. We begin with the standard Measure-Once Quantum Finite Automaton (MOQFA) and Measure-Many Quantum Finite Automaton (MMQFA) models, followed by the Latvian Quantum Finite Automaton (LQFA), One-Way Quantum Finite Automaton with Classical States (1QFAC), Control-Language Based One-Way Quantum Finite Automaton (CL-1QFA), Ancilla-Based Quantum Finite Automaton (A-QFA), and Multi-Letter Quantum Finite Automaton (MLQFA) models. A detailed comparison of these models is provided, highlighting their relative expressive power and computational efficiency.

Measure-Once Quantum Finite Automaton

Definition 3.2.15 (Measure-Once Quantum Finite Automaton)**.** A MOQFA is defined as:

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

where:

- Q is the finite set of quantum states,
- Σ is the input alphabet,
- δ is the transition function implemented by unitary operators,
- $q_0 \in Q$ is the initial quantum state,
- $F \subset Q$ is the set of accepting states.

Operation: The automaton processes the input string by sequentially applying unitary operators determined by δ for each symbol in a left-to-right pass. A single projective measurement is performed at the end, with the outcome over F deciding acceptance.

Language Acceptance: Accepts **REVs** (e.g., $L_{\text{mod}} = \{a^{kp} \mid k \geqslant 0\}$), a strict subset of regular languages [29].

Advantages:

- Minimal quantum resources due to a single measurement.
- Coherence is maintained throughout the computation.

Limitations:

- Limited to reversible regular languages.
- Reliance on a single final measurement may restrict recognition of more complex languages.

Measure-Many Quantum Finite Automaton

Definition 3.2.16 (Measure-Many Quantum Finite Automaton)**.** A MMQFA is defined as

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

where:

- Q is the finite set of quantum states,
- Σ is the input alphabet,
- δ is the unitary transition function,
- $q_0 \in Q$ is the initial state,

- $Q_{acc} \subset Q$ is the set of accepting states,
- $Q_{rej} \subset Q$ is the set of rejecting states,
- $Q_{non} \subset Q$ is the set of non-halting states.

Operation: For each input symbol σ, the automaton applies the corresponding unitary operator U_σ and immediately performs a measurement. If the measurement yields a state in Q_{acc} or Q_{rej}, the automaton halts; otherwise, it continues processing.

Language Acceptance: Recognises **a subclass of regular languages with bounded error** (e.g., $L = \{w \mid |w|_a \equiv 1 \mod 2\}$) [2].

Advantages:

- Bounded error recognition via intermediate measurements.
- Exponential state reduction for specific language families.

Limitations:

- Limited to regular languages.
- Increased complexity due to frequent measurements.

Latvian Quantum Finite Automaton

Definition 3.2.17 (Latvian Quantum Finite Automaton [2])**.** A LQFA is defined as

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

where:

- Q is the finite set of quantum states,
- Σ is the input alphabet,
- δ is the transition function implementing unitary steps,
- $q_0 \in Q$ is the initial state,
- $F \subset Q$ is the set of accepting states.

Operation: After each input symbol, a unitary transformation dictated by δ is applied, immediately followed by a measurement. This immediate measurement ensures that only critical quantum states are preserved for further computation.

Language Acceptance: Accepts a **subset of regular languages** (e.g., certain parity languages) where the measurement process preserves the necessary states.

Advantages:

- Integrates quantum evolution with classical-like decision making.
- Reduced state complexity for symmetric languages.

Limitations:

- Susceptible to premature state collapse due to frequent measurements.
- Not closed under concatenation.

One-Way Quantum Finite Automaton with Classical States

Definition 3.2.18 (One-Way Quantum Finite Automaton with Classical States)**.** A 1QFAC is defined as

$$M = (S, Q, \Sigma, \delta, \mu, s_0, q_0, F)$$

where:

- S is the finite set of classical states,
- Q is the finite set of quantum states,
- Σ is the input alphabet,
- δ is the classical transition function over S,
- μ is the quantum operation function acting on Q,
- $s_0 \in S$ is the initial classical state,
- $q_0 \in Q$ is the initial quantum state,
- $F \subset S \times Q$ is the set of accepting state pairs.

Operation: The automaton proceeds in two intertwined stages:

1. A classical state transition is performed using δ.
2. The quantum register is simultaneously updated via μ.

A final joint measurement over the composite state (s, q) determines acceptance.

Language Acceptance: Recognises **all regular languages** while requiring only $O(\log n)$ quantum states.

Advantages:

- Efficient simulation of classical automata with quantum enhancements.
- Maintains a small quantum register.

Limitations:

- Equivalence checking is undecidable [23].
- The quantum component remains sensitive to decoherence.

[42]

Control-Language Based One-Way Quantum Finite Automaton

Definition 3.2.19 (Control-Language Based One-Way Quantum Finite Automaton)**.** A CL-1QFA is defined as

$$M = (Q, \Sigma, \delta, q_0, \mathcal{L})$$

where:

- Q is the finite set of quantum states,
- Σ is the input alphabet,
- δ is the quantum transition function,
- $q_0 \in Q$ is the initial quantum state,
- $\mathcal{L} \subseteq \Sigma^*$ is a predetermined control language.

Operation: The automaton evolves its quantum state according to δ while outcomes from intermediate measurements are interpreted using the control language $\mathcal{L}$ to guide acceptance or rejection.

Language Acceptance: Designed to recognise **regular languages closed under Boolean operations** (e.g., $L = a\Sigma^* \cup \Sigma^* b$).

Advantages:

- Strong closure properties via Boolean operations.
- Modularity through separation of quantum evolution and classical control.

Limitations:

- Precomputation of $\mathcal{L}$ introduces overhead.
- Integration of two paradigms increases overall complexity.

Ancilla-Based Quantum Finite Automaton

Definition 3.2.20 (Ancilla-Based Quantum Finite Automaton)**.** An A-QFA is defined as

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

with an expanded Hilbert space

$$\mathcal{H}_{\text{total}} = \mathcal{H} \otimes \mathcal{H}_{\text{ancilla}},$$

where:

- Q is the finite set of quantum states in $\mathcal{H}$,
- Σ is the input alphabet,
- δ is the quantum transition function,
- $q_0 \in Q$ is the initial quantum state,
- $F \subset Q$ is the set of accepting states,
- $\mathcal{H}_{\text{ancilla}}$ represents the ancilla qubits used to enhance computation.

Operation: Ancilla qubits are employed to simulate nondeterminism or enhance interference, with entanglement between the main system and ancilla allowing the automaton to explore additional computational pathways.

Language Acceptance: Recognises **regular languages** with reduced state complexity compared to classical automata. For example, the language $L = \{w \mid |w|_a \equiv 0 \mod m\}$ can be recognized with $O(\log m)$ states.

Advantages:

- Enables exponential state reduction via quantum entanglement.
- Extends recognition capabilities beyond classical regular languages.

Limitations:

- Increased complexity in managing ancilla qubits.
- Enhanced risk of error propagation between subsystems.

Multi-Letter Quantum Finite Automaton

Definition 3.2.21 (Multi-Letter Quantum Finite Automaton)**.** A MLQFA is defined as

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

where:

- Q is the finite set of quantum states,
- Σ is the input alphabet,
- δ is the transition function applied to blocks of k symbols,
- $q_0 \in Q$ is the initial quantum state,
- $F \subset Q$ is the set of accepting states,
- $k \geqslant 2$ is the block size used for processing the input.

Operation: The automaton processes the input in blocks of k symbols, applying a unitary operator to each block, thereby capturing long-range dependencies within the input.

Language Acceptance: Recognises **regular languages** with $O(2^k)$ states, achieving exponential savings for specific regular languages (e.g., $L = \{w \mid \text{block-periodic patterns}\}$).

Advantages:

- Efficient parallel processing of input blocks.
- Facilitates polynomial-time recognition of complex patterns.

Limitations:

- Exponential growth in state complexity with increasing k.
- Design of unitary operators for large k is challenging.

Detailed Hierarchy of 1QFA Models

$$\begin{tikzcd}
\text{MLQFA} & \\
& \\
\text{1QFAC} & \text{A-QFA} \\
\text{MMQFA} \arrow[u] & \text{CL-1QFA} \arrow[u] \\
\text{MOQFA} \arrow[u] \arrow[r] & \text{LQFA} \arrow[u]
\end{tikzcd}$$

Figure 3.6: Hierarchy of one-way QFA models. Arrows indicate strict increases in **state efficiency** for regular languages. None of these models exceed the computational power of classical finite automata.

Comparison to Classical Finite Automata

Quantum models exhibit both advantages and limitations compared to their classical counterparts:

- **State efficiency**: MOQFA require fewer states for periodic languages [29], but cannot recognise all regular languages.
- **Error bounds**: MMQFA achieve bounded error for specific languages [2], unlike deterministic finite automata.
- **Closure properties**: Classical automata are closed under all Boolean operations [44], while most QFA variants are not.

3.2.7 Promise Problem Solvers

This section examines quantum automata designed for language recognition under precise error constraints, focusing on exact and bounded-error models.

Exact Quantum Finite Automaton

Definition 3.2.22 (Exact Quantum Finite Automaton (Exact-QFA))**.** An exact quantum finite automaton is defined as:

$$M = (Q, \Sigma, \{\mathcal{U}_\sigma\}, q_0, Q_{acc}, Q_{rej})$$

where:

- $\mathcal{U}_\sigma \in \mathbb{C}^{|Q|\times|Q|}$: Unitary transition matrices
- $Q_{acc} \cup Q_{rej} \subseteq Q$: Partitioned outcome states
- $Q_{acc} \cap Q_{rej} = \varnothing$

Operation:

1. **Unitary Evolution:** For input $w = \sigma_1 ... \sigma_n$:
$$|\psi_w\rangle = \mathcal{U}_{\sigma_n} \cdots \mathcal{U}_{\sigma_1} |q_0\rangle$$
2. **Exact Measurement:** Project onto Q_{acc} and Q_{rej} subspaces
3. **Zero Error:** $\text{supp}(\psi_w) \subseteq Q_{acc} \cup Q_{rej}$ for all w [37]

Language Acceptance: Solves **promise problems** with strict separation:

$$L = \{w \in \Sigma^* \mid \|\Pi_{acc} |\psi_w\rangle\|^2 = 1\}$$

$$\overline{L} = \{w \in \Sigma^* \mid \|\Pi_{rej} |\psi_w\rangle\|^2 = 1\}$$

including the EVENODD$_k$ problem for any $k \in \mathbb{N}$.

Advantages:

- Perfect completeness/soundness ($\epsilon = 0$)
- Exponential state advantage over classical exact automata

Limitations:

- Only recognises languages with trivial automorphism groups [12].
- Requires precise initialization of quantum states

Bounded-Error Quantum Finite Automaton

Definition 3.2.23 (Bounded-Error Quantum Finite Automaton (BEQFA))**.** A bounded-error quantum finite automaton is defined as:

$$M = (Q, \Sigma, \delta, q_0, \epsilon, \{\mathcal{M}_\sigma\})$$

where:

- $\mathcal{M}_\sigma$: Quantum operations with $\|\mathcal{M}_\sigma\| \leqslant 1$
- $\epsilon < \frac{1}{2}$: Error bound

Operation:

1. **Amplitude Amplification:** Applies Grover-like iterations [19, 4]:

   $$\mathcal{M}_\sigma = -U_\sigma S_0 U_\sigma^\dagger S_\chi$$

   where S_0, S_χ are phase oracles

2. **Bounded Distinction:** Maintains:

   $$|\langle\psi_w| Q_{acc} |\psi_w\rangle - \chi_L(w)| \leqslant \epsilon \quad \forall w$$

Language Acceptance: Recognises **modular languages** with:

$$\Pr[M \text{ accepts } w] = \cos^2\left(\frac{\pi|w|_a}{p}\right) \pm \epsilon$$

for prime p, solving MOD$_p$ problems [4].

Advantages:

- Tolerates implementation inaccuracies
- Recognises non-promise languages through thresholding

Limitations:

- Error rate depends on input structure ($\epsilon \propto 1/\sqrt{n}$)
- Fails on languages requiring perfect interference

Comparative Analysis and Hierarchy

Key Comparisons:

- **Error Tradeoff:** Exact-QFA (0 error) vs Bounded-QFA ($\epsilon < 1/2$)
- **Language Classes:**
  - Exact: EVENODD$_k$, L_{pal}^{exact}
  - Bounded: MOD$_p$, $L_{threshold}$
- **Complexity:** Exact-QFA requires $O(\log p)$ states vs $O(1)$ for bounded

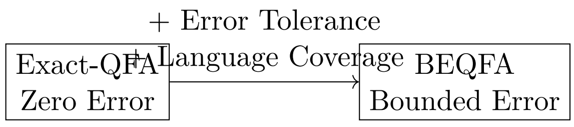

Figure 3.7: Relationship between promise problem solvers. Bounded-error models subsume exact recognition with relaxed constraints.

3.2.8 Specialised Quantum Automata Models

This section examines quantum automata models designed for specific computational paradigms and specialised language recognition tasks.

Quantum Queue Automaton

Definition 3.2.24 (Quantum Queue Automaton (QQA))**.** A quantum queue automaton is defined as:

$$M = (Q, \Sigma, \Gamma, \delta, q_0, F, \rho_{init})$$

where:

- $\Gamma = \{|\gamma_i\rangle\}$: Quantum queue with basis states
- ρ_{init}: Initial mixed state of the queue
- $\delta : Q \times \Sigma \times \Gamma \to Q \times \Gamma \times \{Enq, Deq\}$

Operation:

1. **Enqueue:** Applies unitary $U_{enq}(|q\rangle \otimes |\gamma\rangle) = |q'\rangle \otimes |\gamma\sigma\rangle$
2. **Dequeue:** Measures queue head via POVM $\{E^{\sigma}_{deq}\}$
3. **Real-Time Processing:** Operates in linear time $O(n)$ [59]

Language Acceptance: Recognises **real-time context-sensitive languages** including:

$$L = \{a^n b^n c^n | n \geqslant 0\} \quad \text{with bounded error } \epsilon < 1/3$$

Advantages:

- Quantum superposition enables parallel queue operations
- Exponential state compression vs classical queue automata

Limitations:

- Requires quantum memory for queue storage
- Entanglement maintenance between head and queue

Quantum Pushdown Automaton

Definition 3.2.25 (Quantum Pushdown Automaton (QPA))**.** A quantum pushdown automaton extends QFA with stack:

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

where:

- Γ: Stack alphabet with bottom marker $\perp$
- $\delta : Q \times \Sigma \times \Gamma \to Q \times \Gamma^* \times \mathcal{U}(\mathcal{H})$

Operation:

1. **Stack Operations:** Simultaneous push/pop via entanglement:

$$|q\rangle\,|\gamma\rangle \xrightarrow{U_\sigma} \sum \alpha_i\,|q_i\rangle\,|\gamma_i...\gamma_k\rangle$$

2. **Non-Determinism:** Quantum branching through stack superposition

Language Acceptance: Recognises **non-context-free languages** including:

$$L = \{a^n b^{2^n} | n \geqslant 0\} \quad \text{with probability } \geqslant 3/4.$$

[9]

Advantages:

- Quantum stack enables efficient recognition of some context-sensitive languages [4]
- Overcomes pumping lemma restrictions through interference

Limitations:

- Undecidable emptiness problem
- Requires error-correction for stack decoherence

Postselection Quantum Finite Automaton

Definition 3.2.26 (Postselection Quantum Finite Automaton (PSQFA))**.** A postselection QFA [1] is defined as:

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

where:

- $Q_{post} \subset Q$: Postselection subspace
- Π_{acc}: Projector onto accepting states

Operation:

1. **Postselection:** Conditions probability space on:

$$\Pr_{post}[w] = \frac{\|\Pi_{acc} |\psi_w\rangle \|^2}{\|\Pi_{post} |\psi_w\rangle \|^2}$$

2. **Error Amplification:** Converts weak measurements to definitive outcomes [60]

Language Acceptance: Solves **promise problems** with:

$$\Pr_{post}[w \in L] \geqslant 1 - \epsilon \quad \text{and} \quad \Pr_{post}[w \notin L] \leqslant \epsilon$$

for $\epsilon < 1/2^n$

Advantages:

- Tolerates exponentially small acceptance probabilities
- Enforces exact recognition through ratio tests

Limitations:

- Requires postselection oracle
- Limited to problems with non-zero acceptance gaps

Quantum Turing Machine

Definition 3.2.27 (Quantum Turing Machine (QTM))**.** A quantum Turing machine is defined as:

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

where:

- Γ: Tape alphabet including blank symbol
- $\delta : Q \times \Gamma \to \mathbb{C}^{Q \times \Gamma \times \{L,R\}}$

Operation:

1. **Quantum Transitions:** Evolves as:

$$|q, t\rangle \xrightarrow{U} \sum c_i |q_i, \sigma_i, d_i\rangle$$

2. **Observable Tape:** Measurement collapses superposition to classical configuration

Language Acceptance: Recognises **all BQP languages** [48] with:

$$w \in L \Rightarrow \Pr[M \text{ accepts } w] \geqslant 2/3$$

$$w \notin L \Rightarrow \Pr[M \text{ accepts } w] \leqslant 1/3$$

as per complexity-theoretic bounds [39]

Advantages:

- Universal model for quantum computation
- Solves factoring/discrete log in BQP

Limitations:

- Requires exponential resources for exact simulation
- Decoherence limits practical operation time

Enhanced Quantum Finite Automaton

Definition 3.2.28 (EQFA)**.** An enhanced QFA with superoperators is defined as:

$$\mathcal{E} = (\Sigma, Q, \{\mathcal{S}_\sigma\}, \rho_0, F)$$

where:

- $\mathcal{S}_\sigma$: CPTP maps for each $\sigma \in \Sigma$
- ρ_0: Initial density matrix

Operation:

1. **Noisy Evolution:** Applies quantum channels:

$$\rho_{i+1} = \mathcal{S}_{\sigma_i}(\rho_i)$$

2. **General Measurement:** Uses POVM $\{E_a, E_r\}$ for decision

Language Acceptance: Recognises **stochastic languages** with:

$$L = \{w | \mathrm{tr}(E_a \rho_w) > \lambda\}$$

for threshold $\lambda \in [0, 1]$ [24]

Advantages:

- Models realistic noise and decoherence
- Subsumes classical PFA recognition capabilities

Limitations:

- Undecidable equivalence problem
- Quadratic slowdown vs unitary models

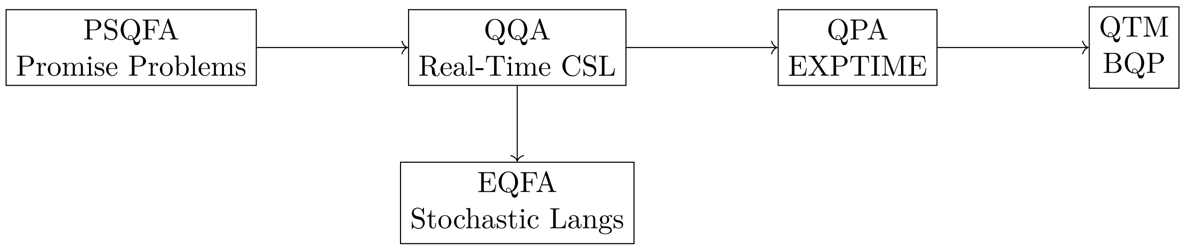

Figure 3.8: Hierarchy of specialised quantum automata. Arrows indicate increasing computational power.

Comparative Analysis and Hierarchy

Key Comparisons:

- **Expressiveness:** PSQFA < QQA < QPA < QTM with EQFA handling distinct stochastic class
- **Complexity:** QTM (BQP-complete), QPA (EXPTIME), QQA (PSPACE)
- **Noise Tolerance:** EQFA most robust, QTM most sensitive

3.2.9 Two-Way Quantum Automata Models

2QFAs extend one-way models by allowing bidirectional head movement. This additional capability enables the recognition of non-regular languages through quantum interference. In the sections below, we present uniform definitions and detailed discussions for standard, hybrid, and multi-tape variants of 2QFA.

Two-Way Quantum Finite Automaton

Definition 3.2.29 (Two-Way Quantum Finite Automaton)**.** A 2QFA is defined as

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

where:

- Q is the finite set of quantum states.
- Σ is the input alphabet.
- δ is the transition function, where $\delta : Q \times \Sigma \to Q \times \{\leftarrow, \downarrow, \rightarrow\}$ associates a quantum operation (typically via unitary operators U_σ) and a head movement direction.
- $q_0 \in Q$ is the initial quantum state.
- $Q_{\text{acc}} \subset Q$ is the set of accepting states.
- $Q_{\text{rej}} \subset Q$ is the set of rejecting states.

Operation: The automaton processes the input bidirectionally by applying the unitary operators U_σ as dictated by δ. The head moves left, right, or remains stationary ($\downarrow$) according to the specified direction. Measurements are deferred until the computation halts, at which point the state is projected onto Q_{acc} or Q_{rej}. The bidirectional tape head enables interference between non-classical paths, achieving exponential advantages for certain languages

Language Acceptance: Recognises **non-regular languages in polynomial time** , for example:

- $L_{\text{eq}} = \{a^n b^n \mid n \geqslant 0\}$ [29].
- Palindromes $L_{\text{pal}} = \{ww^R\}$ [59].

Advantages:

- Provides an exponential state advantage over classical two-way deterministic finite automata (2DFA).
- Leverages quantum parallelism to resolve non-regularity.

Limitations: Unbounded head movement complicates physical implementation, though recent proposals for 2D quantum hardware mitigate this.

Two-Way Quantum Classical Finite Automaton

Definition 3.2.30 (Two-Way Quantum Classical Finite Automaton)**.** A Two-Way Quantum Classical Finite Automaton (2QCFA) is defined as

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

where:

- S is the finite set of classical states.
- Q is the finite set of quantum states.
- Σ is the input alphabet.
- Θ is a function that assigns quantum operations (unitary operators) based on the current classical state and tape symbol, i.e., $\Theta : S \times \Sigma \to \mathcal{U}(Q)$.
- δ is the classical transition function that updates S and directs head movement.
- $s_0 \in S$ is the initial classical state.
- $q_0 \in Q$ is the initial quantum state.
- $S_{\text{acc}} \subset S$ is the set of accepting classical states.
- $S_{\text{rej}} \subset S$ is the set of rejecting classical states.

Operation: The classical component S guides head movement and determines when to trigger quantum operations. At each step, $\Theta(s, \sigma) = U_\sigma$ is applied to the quantum register, and the classical state is updated via δ. The process continues until a terminal classical state in S_{acc} or S_{rej} is reached.

Language Acceptance: Recognises **context-free languages** (e.g., $\{a^n b^n\}$) and **stochastic languages** with bounded error.

Advantages:

- Only a constant quantum memory is required.
- Effectively balances classical control with quantum computation.

Limitations: Hybrid classical-quantum coordination introduces latency, though this is offset by logarithmic space complexity for some languages.

Two-Way Two-Tape Quantum Classical Finite Automaton

Definition 3.2.31 (Two-Way Two-Tape Quantum Classical Finite Automaton)**.** A Two-Way Two-Tape Quantum Classical Finite Automaton (2TQCFA) is defined as

$$M = (S, Q, \Sigma_1 \times \Sigma_2, \Theta, \delta, s_0, q_0, S_{\text{acc}}, S_{\text{rej}})$$

where:

- S is the finite set of classical states.
- Q is the finite set of quantum states.
- Σ_1 and Σ_2 are the input alphabets for the two tapes.
- $\Sigma_1 \times \Sigma_2$ denotes the combined input alphabet for simultaneous tape processing.
- Θ assigns quantum operations based on the current classical state and the pair of tape symbols, i.e., $\Theta : S \times (\Sigma_1 \times \Sigma_2) \to \mathcal{U}(Q)$.
- δ is the classical transition function that synchronises head movements on both tapes.
- $s_0 \in S$ is the initial classical state.
- $q_0 \in Q$ is the initial quantum state.
- $S_{\text{acc}} \subset S$ is the set of accepting states.
- $S_{\text{rej}} \subset S$ is the set of rejecting states.

Operation: The automaton processes both tapes in parallel. The classical control S synchronises the head movements, while Θ applies the appropriate quantum operation based on the current pair of symbols read from the tapes. This approach is useful for cross-tape comparisons, such as verifying languages like $a^n b^n c^n$.

Language Acceptance: Recognises **context-sensitive languages** such as:

- $L = \{a^n b^n c^n \mid n \geqslant 0\}$ [61, 20].

Advantages:

- Achieves exponential succinctness over classical multi-tape automata.
- Exploits quantum correlations to solve interdependency problems across tapes.

Limitations:

- Synchronization of heads across tapes introduces complexity.
- Error correction complexity increases with the number of tapes.

k-Tape Quantum Classical Finite Automaton

Definition 3.2.32 (k-Tape Quantum Classical Finite Automaton)**.** A k-Tape Quantum Classical Finite Automaton (kTQCFA) is defined as

$$M = \left(S, Q, \bigtimes_{i=1}^{k} \Sigma_i, \Theta, \delta, s_0, q_0, S_{\text{acc}}, S_{\text{rej}}\right)$$

where:

- S is the finite set of classical states.
- Q is the finite set of quantum states.
- For each tape i $(1 \leqslant i \leqslant k)$, Σ_i is its input alphabet.
- $\bigtimes_{i=1}^{k} \Sigma_i$ denotes the combined input alphabet for all k tapes.
- Θ is the quantum operation function, mapping the classical state and the k-tuple of symbols to a unitary operator, i.e., $\Theta : S \times \left(\prod_{i=1}^{k} \Sigma_i\right) \to \mathcal{U}(Q)$.
- δ is the classical transition function that coordinates state transitions and synchronises head movements across all k tapes.
- $s_0 \in S$ is the initial classical state.
- $q_0 \in Q$ is the initial quantum state.
- $S_{\text{acc}} \subset S$ is the set of accepting classical states.
- $S_{\text{rej}} \subset S$ is the set of rejecting classical states.

Operation: The automaton synchronises the heads on all k tapes using the classical control S. At each step, it reads a k-tuple of symbols $(\sigma_1, \ldots, \sigma_k)$ and applies the corresponding unitary operator $U_{\sigma_1,\ldots,\sigma_k} = \Theta(s, \sigma_1, \ldots, \sigma_k)$ to the quantum register Q. Measurement is performed adaptively to halt the computation once S_{acc} or S_{rej} is reached.

Language Acceptance: Recognises languages with k-dimensional correlations, such as:

- $L = \{a^n b^{n^2} c^{n^3} \mid n \geqslant 1\}$, using parallel comparisons across tapes [61].
- Other context-sensitive languages that require cross-tape pattern matching (e.g., $\{a^n b^n c^n d^n\}$).

Advantages:

- **Exponential State Savings:** Recognises complex languages with constant quantum states [6].
- **Parallel Processing:** Leverages quantum superposition to correlate information across all k tapes simultaneously.

Limitations:

- **Synchronization Overhead:** The classical control δ must coordinate $O(k)$ head movements.
- **Error Propagation:** A decoherence event on any single tape may affect the overall computation.

Hierarchy of 2QFAs

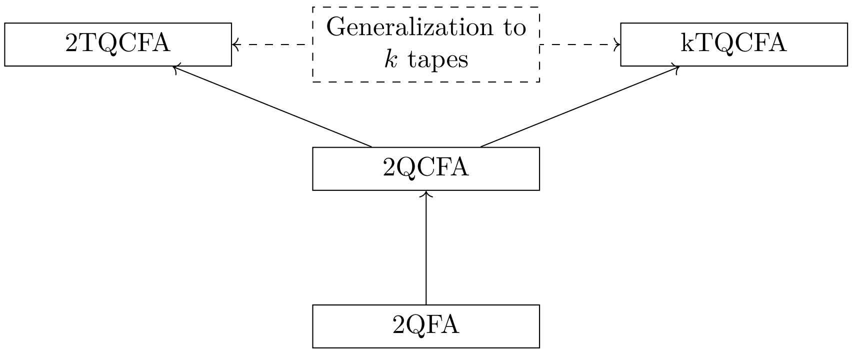

Figure 3.9: Hierarchy of 2QFA models. The kTQCFA generalises the two-tape model (2TQCFA) to k tapes, enhancing recognition power at the cost of increased synchronization complexity [20].

Advantages Over Classical Models

- **Non-regular recognition**: 2QFA solves $\{a^n b^n\}$ in $O(n)$ time vs. classical $O(2^n)$ space [[7]].
- **Error resilience**: Quantum Zeno effect in 2QCFA suppresses decoherence during bidirectional sweeps [[5]].
- **Architecture compatibility**: Recent 2D LDPC-cat qubit designs support two-way head movement [[5]].

4. Automata To Circuits

This chapter describes the process of converting finite automata into circuits. The conversion is done in two steps: first, we convert the automata into a binary decision diagram (BDD), and then we convert the BDD into a circuit. The BDD is a data structure that represents a Boolean function, and it can be used to simplify the representation of the automata. The circuit is a hardware representation of the automata that can be used for various applications, such as model checking or formal verification. The conversion process is described in detail, and the resulting circuits are analyzed for their size and complexity. The chapter also discusses the advantages and disadvantages of using circuits for representing automata, and it provides examples of how the conversion process can be applied to different types of automata. The chapter is structured as follows:

- Section introduces the concept of converting automata to circuits and provides an overview of the process.
- Section describes the conversion of automata to BDDs, including the algorithms used and the properties of BDDs.
- Section discusses the conversion of BDDs to circuits, including the types of circuits that can be generated and their properties.
- Section provides examples of the conversion process applied to different types of automata.
- Section concludes the chapter with a summary of the main points and a discussion of future work in this area.

The chapter is intended for readers with a background in formal methods, automata theory, and circuit design. It assumes familiarity with the basic concepts of finite automata, BDDs, and circuits. The chapter is self-contained and provides all the necessary background information for understanding the conversion process.

4.1 Measure-Once Quantum Finite Automaton to Circuit

This section describes the Measure-Once Quantum Finite Automaton (MOQFA) model, which is a generalisation of the Measure-Many Quantum Finite Automaton (MMQFA) model. The Measure-Once Quantum Finite Automaton model is defined in a similar way to the Measure-Many Quantum Finite Automaton model, but it allows for multiple measurements at each state. This section will cover the definition of the Measure-Once Quantum Finite Automaton model, its acceptance conditions, and its properties.

4.2 Measure-Many Quantum Finite Automaton to Circuit

This section describes the Measure-Many Quantum Finite Automaton (MMQFA) model, which is a generalisation of the Measure-Once Quantum Finite Automaton (MOQFA) model. The Measure-Many Quantum Finite Automaton model is defined in a similar way to the Measure-Once Quantum Finite Automaton model, but it allows for multiple measurements at each state. This section will cover the definition of the Measure-Many Quantum Finite Automaton model, its acceptance conditions, and its properties.

5. Conclusion

This thesis systematically analysed Quantum Finite Automata by comparing them to classical models. The goal was to clarify where quantum advantages exist, quantify their costs, and identify practical limitations.

Three key results emerged. First, One-Way Quantum Finite Automata such as MMQFA achieve exponential state reduction over classical automata for regular languages but require careful error management. Second, Two-way and Hybrid models like 2QCFA extend recognition to non-regular languages (e.g., $\{a^n b^n\}$) with constant quantum memory, though classical control introduces synchronization overhead. Third, generalised models and interactive protocols reveal fundamental trade-offs: while they theoretically recognise complex languages, their reliance on noise modeling or multi-prover coordination makes them experimentally challenging.

These findings have concrete implications. For algorithm designers, hybrid models like 1QFACs offer a pragmatic balance—using minimal quantum resources while retaining classical reliability. For hardware developers, the analysis underscores the need for error correction tailored to automata (e.g., handling decoherence in MOQFAs's unitary evolution). The undecidability of equivalence problems for many QFAs also highlights a theoretical barrier: verifying quantum automata behavior may require new formal methods.

Future work should prioritise practical over theoretical novelty. Compiling QFAs into quantum circuits would test their real-world viability, while equivalence studies between QFAs and QTMs could unify computational models. For industry, implementing 2QCFAs on Noisy Intermediate-Scale Quantum (NISQ) devices to solve simple context-sensitive problems (e.g., XML validation) might demonstrate near-term utility.

In summary, this work clarifies what quantum automata can and cannot do today. It provides a roadmap for leveraging their strengths—state efficiency and parallelism—while cautioning against underestimating their fragility. The path forward lies in bridging formal theory with engineering constraints, not chasing abstract generality.

Abbreviations

1.5QFA	One-and-a-half-Way Quantum Finite Automaton
1DFA	One-Way Deterministic Finite Automaton
1QFA	One-Way Quantum Finite Automaton
1QFAC	One-Way Quantum Finite Automaton with Classical States
2CFA	Two-Way Classical Finite Automaton
2DFA	Two-Way Deterministic Finite Automaton
2NFA	Two-Way Nondeterministic Finite Automaton
2PFA	Two-Way Probabilistic Finite Automaton
2QCFA	Two-Way Quantum Classical Finite Automaton
2QFA	Two-Way Quantum Finite Automaton
2TQCFA	Two-Way Two-Tape Quantum Classical Finite Automaton
A-QFA	Ancilla-Based Quantum Finite Automaton
abstract-QFA	Abstract Quantum Finite Automaton
BCQFA	Blind Counter Quantum Finite Automaton
BEQFA	Bounded-Error Quantum Finite Automaton
CFA	Classical Finite Automaton
CFL	Context-Free Language
CL-1QFA	Control-Language Based One-Way Quantum Finite Automaton
CNOT	Controlled-NOT
CSL	Context-Sensitive Language
DFA	Deterministic Finite Automaton
EQFA	Enhanced Quantum Finite Automaton
Exact-QFA	Exact Quantum Finite Automaton
GHZ	Greenberger-Horne-Zeilinger State
gQFA	Generalised Quantum Finite Automaton
kTQCFA	k-Tape Quantum Classical Finite Automaton

LBA	Linear-Bounded Automaton
LQFA	Latvian Quantum Finite Automaton
MLQFA	Multi-Letter Quantum Finite Automaton
MM-1GQFA	Measure-Many Generalised Quantum Finite Automaton
MMQFA	Measure-Many Quantum Finite Automaton
MO-1GQFA	Measure-Once Generalised Quantum Finite Automaton
MOQFA	Measure-Once Quantum Finite Automaton
MPSQFA	Matrix Product State Quantum Finite Automaton
NFA	Nondeterministic Finite Automaton
NISQ	Noisy Intermediate-Scale Quantum
OLVA	Orthomodular Lattice-Valued Automaton
OTQFA	Open Time Evolution Quantum Finite Automaton
PDA	Pushdown Automaton
PFA	Probabilistic Finite Automaton
PSQFA	Postselection Quantum Finite Automaton
QCPA	Quantum-Classical Parity Automaton
QFA	Quantum Finite Automaton
QIP(1QFA)	Quantum Interactive Proofs with One-Way Quantum Finite Automata
QIP(2QFA)	Quantum Interactive Proofs with Two-Way Quantum Finite Automata
QMIP(2QCFA)	Quantum Merlin-Arthur Proofs with Two-Way Quantum Classical Finite Automata
QPA	Quantum Pushdown Automaton
QQA	Quantum Queue Automaton
QTM	Quantum Turing Machine
qubit	Quantum Bit
REG	Regular Language
REV	Reversible Regular Language
SQA	Semi-Quantum Automaton
TM	Turing Machine

Bibliography

[1] Scott Aaronson. “Complexity classification with verifiers”. In: *Journal of Computer and System Sciences* 71 (2005), pp. 291–316.

[2] Andris Ambainis and Rūsiņš Freivalds. “Probabilistic reversible automata and quantum automata”. In: *ICALP 1998* (1998), pp. 472–483.

[3] Andris Ambainis and Rūsiņš Freivalds. “One-way quantum finite automata: Strengths, weaknesses, and generalizations”. In: *Proceedings of the 39th Annual Symposium on Foundations of Computer Science (FOCS)* (1998), pp. 332–341.

[4] Andris Ambainis and Rūsiņš Freivalds. “Quantum finite automata with control language”. In: *Theoretical Computer Science* 287.1 (2002), pp. 299–311.

[5] Andris Ambainis and Abuzer Yakaryılmaz. “Superiority of quantum finite automata over classical finite automata”. In: *SIAM Journal on Computing* 39.7 (2009), pp. 2819–2830.

[6] Andris Ambainis et al. “Dense Quantum Coding and Quantum Automata”. In: *Journal of the ACM* 49.4 (2002), pp. 496–511. DOI: 10.1145/564376.564379.

[7] Charles H. Bennett and Gilles Brassard. “Quantum cryptography: Public key distribution and coin tossing”. In: *Theoretical Computer Science* 560 (2014), pp. 7–11.

[8] Charles H Bennett et al. “Teleporting an unknown quantum state via dual classical and Einstein-Podolsky-Rosen channels”. In: *Physical Review Letters* 70.13 (1993), pp. 1895–1899.

[9] Alberto Bertoni, Carlo Mereghetti, and Beatrice Palano. “Quantum computing: 1-way quantum automata”. In: *Developments in Language Theory* (2001), pp. 1–20.

[10] Vincent D. Blondel, Emmanuel Jeandel, and Pascal Koiran. “The equivalence problem for quantum automata”. In: *International Journal of Foundations of Computer Science* 14 (2003), pp. 685–696.

[11] Heinz-Peter Breuer and Francesco Petruccione. “The theory of open quantum systems”. In: (2002).

[12] Alex Brodsky and Nicholas Pippenger. “Exact quantum automata”. In: *Quantum Information & Computation* 1 (2001), pp. 3–25.

[13] John F Cady. *The ASCII Standard: A Comprehensive Guide to the American Standard Code for Information Interchange*. Prentice Hall, 1986.

[14] Noam Chomsky. “Three models for the description of language”. In: *IRE Transactions on information theory* 2.3 (1956), pp. 113–124.

[15] Manfred Droste, Werner Kuich, and Heiko Vogler. "Handbook of Weighted Automata". In: *Springer* (2009).

[16] Cynthia Dwork and Larry Stockmeyer. "A Time Complexity Gap for Two-Way Probabilistic Finite Automata". In: *SIAM Journal on Computing* 19.6 (1990), pp. 1011–1023.

[17] Austin G. Fowler et al. "Surface codes: Towards practical large-scale quantum computation". In: *Physical Review A* 86.3 (2012).

[18] Daniel M. Greenberger, Michael A. Horne, and Anton Zeilinger. "Bell's theorem without inequalities". In: *American Journal of Physics* 58.12 (1990), pp. 1131–1143.

[19] Lov K Grover. "A fast quantum mechanical algorithm for database search". In: *Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC)* (1996), pp. 212–219.

[20] Jozef Gruska. *Quantum Computing.* McGraw-Hill, 2005.

[21] Jozef Gruska and Daowen Qiu. "Quantum Finite Automata: A Modern Overview". In: *Theoretical Computer Science* 499 (2013), pp. 1–35.

[22] Stanley P. Gudder. "Quantum automata and logics". In: *International Journal of Theoretical Physics* 17 (1978), pp. 517–529.

[23] Mika Hirvensalo. "On the power of quantum finite automata". In: *Fundamentals of Computation Theory* (2008), pp. 305–316.

[24] Mika Hirvensalo. *Quantum Computing.* Springer, 2012.

[25] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* 3rd. ISBN: 978-8131720479. Pearson Education India, 2006.

[26] Brian W Kernighan and Rob Pike. *The Unix programming environment.* Prentice-Hall, 1984.

[27] Stephen Cole Kleene. "Representation of events in nerve nets and finite automata". In: *Automata studies* 34 (1956), pp. 3–41.

[28] Hirotada Kobayashi and Keiji Matsumoto. "Quantum multi-prover interactive proof systems with limited prior entanglement". In: *Journal of Computer and System Sciences* 66 (2003), pp. 429–450.

[29] Attila Kondacs and John Watrous. "On the power of quantum finite state automata". In: *Proceedings of the 38th Annual Symposium on Foundations of Computer Science (FOCS).* IEEE, 1997, pp. 66–75.

[30] Dexter C. Kozen. *Automata and Computability.* Springer, 1997.

[31] Lvzhou Li et al. "Characterizations of one-way general quantum finite automata". In: *Theoretical Computer Science* 419 (2012), pp. 73–91.

[32] Yuri I. Manin. "Computable and uncomputable". In: (1980).

[33] Cristopher Moore and James P Crutchfield. "Quantum automata and quantum grammars". In: *Theoretical Computer Science.* Vol. 237. 1-2. Elsevier, 2000, pp. 275–306.

[34] Edward F. Moore. "Gedanken-experiments on sequential machines". In: *Automata Studies* (1956), pp. 129–153.

[35] John Myhill. "Finite automata and the representation of events". In: *WADD Technical Report* (1957).

[36] Masaki Nakanishi. "On the Power of One-Sided Error Quantum Pushdown Automata with Classical Stack Operations". In: *Proceedings of the 10th Annual International Computing and Combinatorics Conference (COCOON 2004)*. Vol. 3106. Lecture Notes in Computer Science. Springer, 2004, pp. 179–187. DOI: `10.1007/978-3-540-27798-9_21`.

[37] Ashwin Nayak. "Optimal lower bounds for quantum automata and random access codes". In: *Foundations of Computer Science, 1999. 40th Annual Symposium on* (1999), pp. 369–376.

[38] Anil Nerode. "Linear automaton transformations". In: *Proceedings of the American Mathematical Society* 9.4 (1958), pp. 541–544.

[39] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2010.

[40] Azaria Paz. *Introduction to probabilistic automata*. Academic Press, 1971.

[41] Jean-Éric Pin. "Syntactic and Reversible Automata". In: *Proceedings of the International Conference on Algebraic Informatics*. Springer, 1997.

[42] Daowen Qiu and Xiangfu Yu. "One-way quantum finite automata with classical states". In: *Journal of Computer and System Sciences* 75 (2009), pp. 359–373.

[43] Michael O Rabin. "Probabilistic automata". In: *Information and Control* 6.3 (1963), pp. 230–245.

[44] Michael O. Rabin and Dana Scott. "Finite automata and their decision problems". In: *IBM Journal of Research and Development* 3 (1959), pp. 114–125.

[45] Grzegorz Rozenberg and Arto Salomaa. *Handbook of Formal Languages*. Springer, 1997.

[46] William J. Sakoda and Michael Sipser. "The Complexity of Nondeterministic Finite Automata". In: *Information and Control* 38 (1978), pp. 125–153.

[47] John C. Shepherdson. "The Reduction of Two-Way Automata to One-Way Automata". In: *IBM Journal of Research and Development* 3.2 (1959), pp. 198–200.

[48] Peter W. Shor. "Algorithms for quantum computation: discrete logarithms and factoring". In: *Proceedings 35th Annual Symposium on Foundations of Computer Science* (1994), pp. 124–134.

[49] Peter W Shor. "Scheme for reducing decoherence in quantum computer memory". In: *Physical Review A* 52.4 (1995), R2493–R2496.

[50] Michael Sipser. *Introduction to the Theory of Computation*. Cengage Learning, 2012.

[51] Michael Sipser. *Introduction to the Theory of Computation*. Cengage Learning, 2013.

[52] Thomas A. Sudkamp. *Languages and Machines: An Introduction to the Theory of Computer Science*. Addison-Wesley, 2006.

[53] Alan Mathison Turing. "On computable numbers, with an application to the Entscheidungsproblem". In: *Proceedings of the London Mathematical Society* 2.1 (1936), pp. 230–265.

[54] Guifré Vidal. “Efficient classical simulation of slightly entangled quantum computations”. In: *Physical Review Letters* 91 (2003), p. 147902.

[55] William K. Wootters and Wojciech H. Zurek. “A single quantum cannot be cloned”. In: *Nature* 299.5886 (1982), pp. 802–803.

[56] Abuzer Yakaryılmaz. “Quantum Pumping Lemmas for Regular and Context-Free Languages”. In: *Journal of Computer and System Sciences* 80.8 (2014), pp. 1563–1576.

[57] Abuzer Yakaryılmaz. “Verification of quantum automata via non-adaptive strategies”. In: *Theoretical Computer Science* 486 (2013), pp. 1–8.

[58] Abuzer Yakaryılmaz and A. C. Cem Say. “Efficient recognition by quantum finite automata”. In: *Theoretical Computer Science* 410 (2011), pp. 1932–1941.

[59] Abuzer Yakaryılmaz and A. C. Cem Say. “Succinctness of two-way probabilistic and quantum finite automata”. In: *Discrete Mathematics and Theoretical Computer Science* 12.2 (2010), pp. 19–40.

[60] Tomoyuki Yamakami. “Constant-space quantum interactive proofs against multiple provers”. In: *Information Processing Letters* 114.11 (2014), pp. 611–619.

[61] Shenggen Zheng et al. “Two-tape finite automata with quantum and classical states”. In: *International Journal of Foundations of Computer Science* 23.04 (2012), pp. 887–906.

Acknowledgments