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

Laureando
Marta Musso

Matricola 122360

Relatore
Relatore Name

Correlatore
Correlatore 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 standardizes definitions and systematically compares classical finite automata with their quantum counterparts.

The work begins with a comprehensive review of classical finite automata, including deterministic, nondeterministic, probabilistic, and two-way models, to lay the necessary theoretical foundation. It then introduces the foundational principles of quantum mechanics, such as superposition, entanglement, and measurement, and explains how these principles can be used to define quantum finite automata. The thesis then reviews various models of quantum finite automata, analyzing their formal definitions, computational dynamics, and language recognition properties. Through a detailed literature review and comparative analysis, this document clarifies longstanding ambiguities and identifies open research challenges, such as issues in equivalence checking and complexity trade-offs.

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, quantum-classical hybrids, 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, entanglement, and measurement. At its core lies quantum automata theory, which seeks to understand how these principles redefine the boundaries of classical computation. Classical finite automata—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, QFAs exhibit probabilistic and non-deterministic behaviors that transcend classical limits, necessitating a coherent framework to classify and analyze 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. 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 [3].

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 (MM-1QFA), to recognize languages with exponentially fewer states than their classical counterparts [19]. Practically, as quantum hardware advances, understanding the minimal resources required to implement QFAs becomes critical for designing efficient algorithms and robust error-correcting schemes. Yet, progress in the field has been hindered by ambiguities in model definitions. For example, early quantum automata models like the Measure-Once Quantum Finite Automaton (MO-1QFA) and MM-1QFA were defined with differing acceptance criteria, leading to confusion about their relative computational power [17]. 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 [32]. 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. For instance, although the expressive power of Two-way Quantum Finite

Automaton (2QFA) surpasses that of classical two-way automata, 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 [30]. Moreover, the literature review in this thesis is dedicated to clearly identifying the specific classes of languages each automaton model accepts. 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. 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 [13].

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-1QFA 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 [7]. Second, characterizing the relationships between classical and quantum models necessitates a framework that accounts for both their similarities (e.g., the ability of 1QFAC to simulate DFAs) and their divergences (e.g., the exponential state advantage of 2QFA over two-way probabilistic automata). Third, the absence of standardized 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 [1].

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 quantum finite automata. With this dual background in place, the work systematically explores various quantum models, ranging from early variants like the MO-1QFA [19] and the MM-1QFA [17] to advanced hybrids such as the 1QFAC and enhanced models (e.g., the Enhanced Quantum Finite Automaton (EQFA)). Each model is rigorously analyzed along several dimensions: its formal definition is standardized, its acceptance criteria are scrutinized, 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 recognized by each model.

A significant contribution of this work is the development of a hierarchical taxonomy of automata models, which organizes 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 [30]. 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 Generalized Quantum Finite Automata (gQFAs)) and the conditions under which quantum automata outperform probabilistic models in language recognition tasks [13].

The thesis is organized 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 3 presents a comprehensive catalog of quantum automata models; each model is formally defined, its computational dynamics are analyzed, and its language recognition capabilities are explicitly detailed. Chapter 4 synthesizes 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 [18], extending pumping lemmas to quantum models [1], and developing minimization algorithms for hybrid automata.

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 analyzed 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. [14]—which formalized the equivalence between DFAs and NFAs—and Rabin's seminal work on probabilistic automata [25] 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 contextualized 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 [21] serves as the backbone for these quantum operations.

This chapter is organized to mirror the later hierarchical taxonomy of automata and sets the stage for analyzing hybrid models such as the 1QFAC [32], where classical memory is integrated with quantum processing, as well as for reviewing advanced quantum models.

2.1 Classical Finite Automaton (CFA)

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, 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 [16] laid the groundwork for understanding the algebraic structure of languages. Chomsky's introduction of language hierarchies [12] further clarified how different classes of languages can be recognized by increasingly powerful computational models. Turing's conceptualization of computation [14] provided a model for algorithmic processes, while Rabin's introduction of probabilistic automata [25] expanded the framework to include models where transitions are governed by probability.

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. Similarly, the limitations of context-free languages have led to the development of more advanced parsing techniques in compilers.

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

An *alphabet* Σ is defined as a non-empty, finite set of symbols that serve as the basic elements for constructing strings and, consequently, languages. For instance:

Example 2.1.1. Binary Alphabet: $\Sigma = \{0, 1\}$ is central to digital computing and coding theory [14].

Example 2.1.2. ASCII Alphabet: Σ_{ASCII}, which contains 128 distinct characters used for text encoding [11].

A *string* (or *word*) w over Σ is a finite sequence of symbols $a_1 a_2 \dots a_n$, where each $a_i \in \Sigma$. The **length** of w, denoted by $\|w\|$, is defined as the total number of symbols in the string. The special string ϵ, known as the **empty string**, has a length of zero ($\|\epsilon\| = 0$) [14].

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., $(abc)^R = cba$) [14].
- **Substring**: A string v is a substring of w if there exist (possibly empty) strings x and y such that $w = xvy$ [14].

Languages and Operations

A *language* L is a subset of Σ^*, where Σ^* denotes the **Kleene closure** of Σ, defined as:

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

[16]

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 [14].

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 [14].

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

   $$L^* = \bigcup_{i=0}^{\infty} L^i, \quad \text{where } L^i = \underbrace{L \cdot L \cdots L}_{i \text{ times}}.$$

   Example 2.1.5. For $L = \{0, 1\}$, the set L^* comprises all binary strings, including ϵ [14].

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 [14].

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 [14].

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 [14].

Language Categories

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

1. **Regular Languages (REGs)**: These languages are recognized by DFA and NFA and can be generated by regular expressions [14].

   Example 2.1.6. $L = \{w \in \{a, b\}^* \mid w \text{ contains } aba\}$ is a regular language [14].

2. **Context-Free Languages (CFLs)**: These languages are recognized by Pushdown Automaton (PDA) and generated by context-free grammars [12, 14].

   Example 2.1.7. $L_{\text{pal}} = \{ww^R \mid w \in \{a, b\}^*\}$, the language of palindromes, is a classic example [12].

3. **Context-Sensitive Languages (CSLs)**: Recognized by linear-bounded automata, these languages have structural constraints that extend beyond context-free languages [12, 14].

   Example 2.1.8. $L = \{a^n b^n c^n \mid n \geqslant 1\}$ is an example of a context-sensitive language [12].

4. **Recursively Enumerable Languages (Type-0)**: Recognized by Turing Machines (TMs), these languages embody the notion of algorithmic computability [14, 29].

   Example 2.1.9. The language corresponding to the Halting Problem is recursively enumerable [14].

5. **Stochastic Languages**: Recognized by PFAs with bounded error, stochastic languages allow for probabilistic acceptance criteria [25].

   Example 2.1.10. $L_{\text{eq}} = \{a^n b^n \mid n \geqslant 1\}$ is stochastic but not regular. A PFA can accept it with probability at least $\frac{2}{3}$ for strings in L_{eq} and at most $\frac{1}{3}$ for strings not in L_{eq} [25].

Definition 2.1.1 (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.

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 $xy^i z \in L$.*

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 [14].
- **CFLs**: closed under union and Kleene star, but not under intersection or complement [12, 14].
- **CSLs**: closed under union, intersection, and complement [12, 14].
- **Stochastic Languages**: closed under union, intersection, and concatenation, but not under complementation or Kleene star [25, 24].

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.

Example 2.1.11. The closure of regular languages under intersection guarantees that if $L_1, L_2 \in DFAs$ (or, equivalently, recognized 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, as demonstrated by the inability to recognize

$$\overline{L_{\text{eq}}} \quad \text{for} \quad L_{\text{eq}} = \{a^n b^n \mid n \geqslant 1\}$$

[25].

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 organized into a hierarchical framework known as the Chomsky hierarchy [12, 14]:

1. **Type-3 (Regular)**: Languages recognized by DFAs (or NFAs) [14].
2. **Type-2 (Context-Free)**: Languages recognized by PDAs [12].
3. **Type-1 (Context-Sensitive)**: Languages recognized by linear-bounded automata [12].

4. **Type-0 (Recursively Enumerable)**: Languages recognized by TMs, which formalize the notion of algorithmic computability [14, 29].

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.

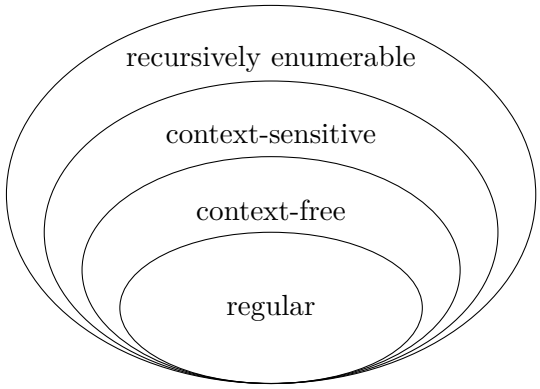


Figure 2.1: Chomsky hierarchy of formal languages

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 analyzers) [15, 14].
- **Context-Free Grammars**: Form the basis for defining the syntax of programming languages such as Python and Java [12, 14].
- **Closure Properties**: Provide a framework for proving decidability results (e.g., the emptiness problem for DFAs) [14].
- **Stochastic Models**: Are applied in areas like natural language processing and speech recognition, where probabilistic pattern matching is essential [25].

2.1.2 CFA Definition Fundamentals

All automata share several core structural components that provide the basis for their computational behavior [14, 12].

Definition 2.1.2 (Classical Finite Automaton (CFA))**.** A *finite automaton* is a computational model that processes input symbols to recognize 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 [14].
- **Input Alphabet (Σ)**: A finite set of symbols that the automaton processes [14].

- **Transition Function (δ)**: A function that governs state changes based on input. For deterministic models, $\delta : Q \times \Sigma \to Q$ (i.e., a DFA); for nondeterministic models, $\delta : Q \times \Sigma \to 2^Q$ (i.e., an NFA) [12, 14].
- **Initial State ($q_0 \in Q$)**: The starting configuration of the automaton [14].
- **Accept States ($F \subseteq Q$)**: A subset of states that, when reached, indicate successful recognition of an input string [14].

Remark. The quintuple $(Q, \Sigma, \delta, q_0, F)$ is a standard representation that facilitates uniform analysis across different classes of automata.

Example 2.1.12. The DFA in Figure 2.2 is defined by:

- $Q = \{q_0, q_1\}$,
- $\Sigma = \{0, 1\}$,
- Transitions such as $\delta(q_0, 1) = q_1$ and $\delta(q_1, 0) = q_1$ *(partial specification)*,
- $F = \{q_0\}$, indicating that the automaton accepts strings with an even number of 1s.

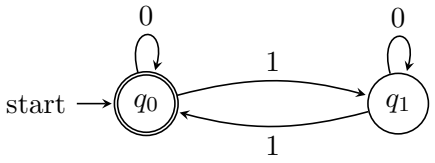


Figure 2.2: DFA example recognizing strings with even number of 1s

Observation 2.1.2. Graphical representations—using states, transitions, and designated initial/accept states—provide an intuitive understanding of automata behavior that complements the formal definitions.

Graphical notation typically includes:

- **States**: Represented by circles labeled with q_i.
- **Initial state**: Indicated by an incoming arrow (e.g., pointing to q_0).
- **Accept states**: Denoted by double circles (e.g., q_1 in Figure 2.2).
- **Transitions**: Illustrated by directed edges with labels representing input symbols.

Automaton	**State Memory**	**Transition Type**	**Acceptance Condition**
DFA	None	Deterministic	Final state membership [14]
NFA	None	Nondeterministic	Existence of an accepting path [14]
PDA	Stack	Deterministic/Nondeterministic	Final state and empty stack [12]
TM	Tape	Deterministic	Halting in an accept state [29]

Table 2.2: Automata representation variations

2.1.3 Deterministic Finite Automaton (DFA)

Definition 2.1.3 (DFA)**.** 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,
- $q_0 \in Q$ is the unique initial state, and
- $F \subseteq Q$ is the set of accepting states.

Remark. A DFA has the property that for every state $q \in Q$ and every input symbol $\sigma \in \Sigma$, there is exactly one transition defined, ensuring deterministic behavior.

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

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

has finitely many equivalence classes. Consequently, every regular language has a unique minimal DFA up to isomorphism.

Example 2.1.13. 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 DFA recognizing L.

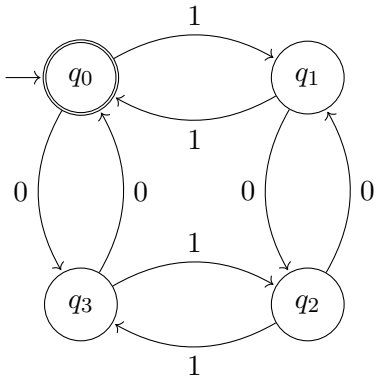


Figure 2.3: DFA recognizing even numbers of `0`s and `1`s.

Observation 2.1.3. Graphical representations (see Figure 2.3) provide an intuitive view of how the DFA tracks the parity of 0's and 1's, reinforcing its deterministic nature.

2.1.4 Nondeterministic Finite Automaton (NFA)

Definition 2.1.4 (NFA)**.** A NFA is a quintuple

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

where:

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

Remark. Unlike DFAs, a NFA 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 2.1.14. Figure 2.4 depicts a NFA that recognizes the language

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

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

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 $S \cap F \neq \varnothing$.
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.

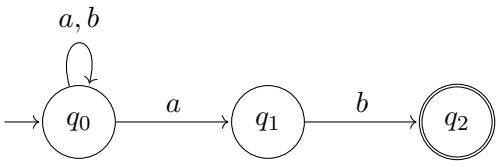


Figure 2.4: NFA recognizing $L = \Sigma^* ab$

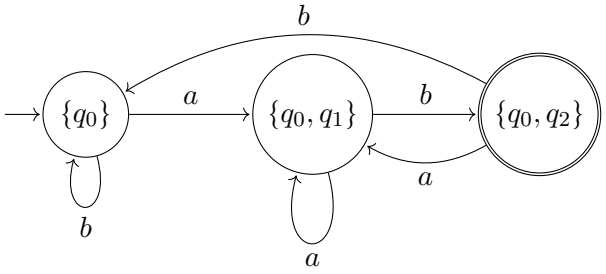


Figure 2.5: Equivalent DFA for NFA in Figure 2.4 [14]

2.1.5 Probabilistic Finite Automaton (PFA)

Definition 2.1.5 (PFA)**.** 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 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 2.1.15. Figure 2.6 illustrates a PFA that recognizes the language

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

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

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

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

Corollary 2.1.2. *For a PFA with a strict cut-point ($\lambda = 1$), the recognized 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.

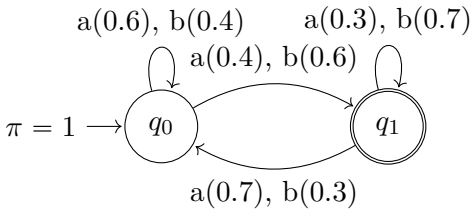


Figure 2.6: PFA for majority language with probabilistic transitions

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 (both one-way and two-way automata recognize exactly the regular languages), two-way models can be exponentially more succinct than one-way models and naturally lend themselves to algorithms in several contexts (e.g., in complexity analysis and even quantum models).

Two-way Deterministic Finite Automaton (2DFA)

Definition 2.1.6 (Two-Way Deterministic Finite Automaton)**.** A Two-way Deterministic Finite Automaton (2DFA) is formally defined as an 8-tuple

$$M = (Q, \Sigma, L, R, \delta, s, t, r),$$

where:

- Q is a finite set of states,
- Σ is a finite input alphabet,
- L and R are special symbols called the left and right endmarkers, respectively (with $L, R \notin \Sigma$),
- $\delta : Q \times (\Sigma \cup \{L, R\}) \to Q \times \{L, R\}$ is the 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.

In addition, the transition function is assumed to satisfy:

- For every state $q \in Q$, when reading the left endmarker L, the head always moves to the right; that is, $\delta(q, L) = (q', R)$ for some $q' \in Q$.
- Similarly, when reading the right endmarker R, the head always moves to the left: $\delta(q, R) = (q', L)$.
- 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 2.1.16 (First and Last Symbol Equality)**.** 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 the left endmarker L and immediately move right to read the first symbol; store it in the control.
2. Continue scanning right until the right endmarker R is reached.
3. Upon reading R, reverse direction (move left) and, in the process, skip any blank moves until the last symbol is reached.
4. Compare the stored first symbol with the last symbol. If they are equal, move to the accept state t; otherwise, move to the reject state r.

A simplified state diagram illustrating this process is provided in Figure **??**.

Observation 2.1.6. Although every 2DFA can be simulated by a one-way DFA, such a simulation may require an exponential increase in the number of states.

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 (2NFA)

Definition 2.1.7 (Two-Way Nondeterministic Finite Automaton)**.** A Two-way Nondeterministic Finite Automaton (2NFA) is defined similarly to a 2DFA but with a nondeterministic transition function. Formally, a 2NFA is an 8-tuple

$$M = (Q, \Sigma, L, R, \delta, s, t, r),$$

where:

- Q is a finite set of states,
- Σ is a finite input alphabet,
- L and R are the left and right endmarkers (with $L, R \notin \Sigma$),
- $\delta : Q \times (\Sigma \cup \{L, R\}) \to 2^{Q \times \{L,R\}}$ 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 2.1.17 (Symmetry Check (Toy Version))**.** 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 right from the left endmarker L while nondeterministically guessing the point where comparison will occur.
2. Upon reaching the right endmarker R, reverse direction.
3. While moving left, 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 2.7 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 recognize remains the same (i.e., the regular languages).

Two-way Probabilistic Finite Automaton (2PFA)

Definition 2.1.8 (Two-Way Probabilistic Finite Automaton)**.** A Two-way Probabilistic Finite Automaton (2PFA) is an 8-tuple

$$M = (Q, \Sigma, L, R, \delta, s, t, r),$$

where:

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

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

- $s \in Q$ 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 2.1.18 (Majority Language)**.** 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 2.8 provides a schematic illustration of such a machine.

Theorem 2.1.4 (Rabin's Theorem for PFAs)**.** *A PFA with an isolated cut-point recognizes exactly the class of regular languages. This result extends to 2PFAs under analogous conditions.*

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

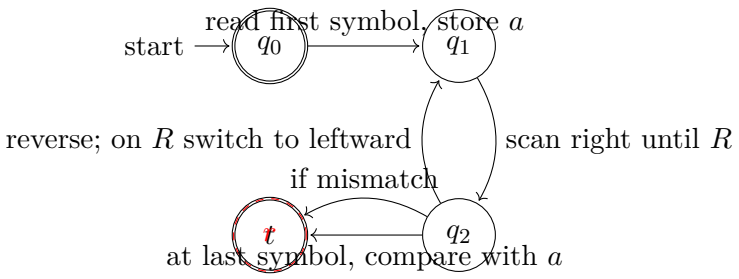
Corollary 2.1.3. *For a 2PFA with a strict cut-point ($\lambda = 1$), the recognized language is equivalent to that of a DFA.*

Comparative Analysis of Two-Way Models

Model	Language Class	Time Complexity	Space Complexity	State Complexity	Key Reference
2DFA	REG	$O(n^2)$	$O(1)$	May be exponentially smaller than 1DFA	[14]
2NFA	REG	$O(n)$	$O(1)$	Can be exponentially more succinct than 1DFA	[30]
2PFA	REG $\subset L \subseteq$ P	$O(n^3)$	$O(\log n)$	Varies with error bounds	[**freivalds1981probabilistic**]

Table 2.3: Comparison of classical two-way automata models (using explicit endmarkers).

Remark. The comparative analysis illustrates that while all two-way automata recognize 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.



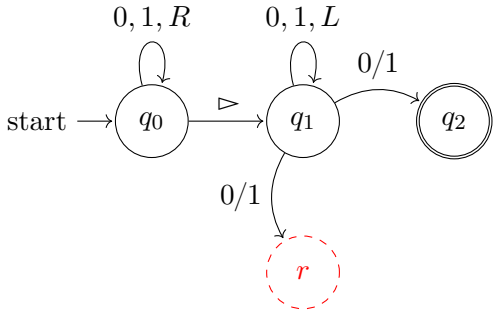


Figure 2.7: Schematic 2NFA illustrating a guess-and-check mechanism (for a toy symmetry language).

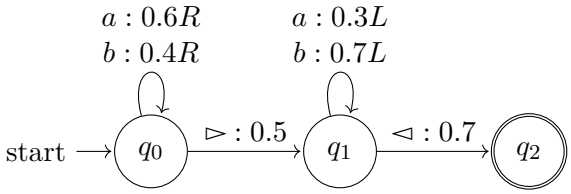


Figure 2.8: Schematic 2PFA for a majority language (with sample probabilistic transitions).

2.2 Quantum Mechanics Foundations

This section establishes the quantum mechanical principles that underpin quantum automata theory. We emphasize 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 normalized 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 2.9 illustrates the **Bloch sphere** representation of a qubit.

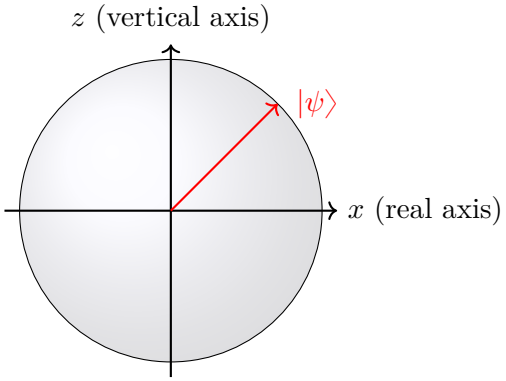


Figure 2.9: Bloch sphere representation of a Qubit.

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.

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:

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

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

Observation 2.2.2. Entanglement is the resource behind many quantum protocols such as quantum teleportation [5] and superdense coding, and it plays a pivotal role in the computational speedup promised by algorithms like Shor's factorization algorithm [27].

2.2.3 Quantum Gates and Circuits

Definition 2.2.5 (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 2.2.2 (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 the previous section.
- **Phase Shift:**
$$R_\phi = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix}.$$

Definition 2.2.6 (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.

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

Example 2.2.5 (Deutsch-Jozsa Quantum Circuit)**.** Figure 2.10 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.

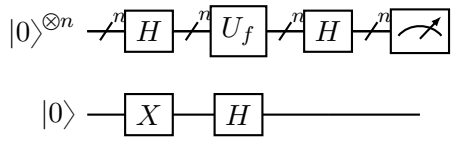


Figure 2.10: Quantum circuit for the Deutsch-Jozsa algorithm.

2.2.4 Measurement and Probabilistic Outcomes

Definition 2.2.7 (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.

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 2.2.8 (POVM Measurement)**.** A more general framework is provided by Positive Operator-Valued Measures (POVMs) (Positive Operator-Valued Measures). 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 when measuring the state $|\psi\rangle$ is given by

$$P(i) = \langle\psi|E_i|\psi\rangle.$$

Example 2.2.6 (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 2.2.9 (Mixed State)**.** A *mixed state* describes a statistical ensemble of quantum states 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 when dealing with open systems affected by decoherence.

2.2.5 Decoherence and Open Systems

Definition 2.2.10 (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 2.2.11 (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 [10].

Example 2.2.7 (Noise Models)**.** Typical noise models include:

- **Amplitude damping:** models energy loss (e.g., spontaneous emission).
- **Phase damping:** represents the loss of phase coherence without energy dissipation.

Observation 2.2.4. To combat decoherence, quantum error correction codes (such as the Shor code [28]) and decoherence-free subspaces are employed.

2.2.6 Additional Remarks: Unitary Evolution and Quantum Dynamics

Definition 2.2.12 (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 discretized 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:*

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

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.

3. Literature Review

This chapter reviews the literature on quantum finite automata QFA, juxtaposing the quantum models with their classical counterparts. We discuss one-way and 2QFA models, hybrid variants that combine classical control with quantum operations, enhanced models employing mixed states and open-system dynamics, as well as interactive models. For each model, we present formal definitions, operational descriptions, key features, language acceptance properties, and known limitations. Throughout, a homogeneous notation is employed: quantum states are represented as kets (e.g. $|q\rangle$), transition functions are expressed in either unitary or superoperator forms, and the state sets (e.g. Q_{acc}, Q_{rej}, Q_{non}) are uniformly denoted.

3.1 One-way QFA

One-way QFAs process the input tape in a single left-to-right pass. In this section, we review the standard models of one-way QFA, including both the measure-once and measure-many variants, and we also describe a notable alternative model—the Latvian Quantum Finite Automaton (LQFA). For each model we detail the formal definition, the operational mechanism, the class of languages accepted, and the inherent limitations.

3.1.1 Standard Models

Measure-Once Quantum Finite Automaton (MO-1QFA)

Definition 3.1.1 (MO-1QFA)**.** An MO-1QFA is defined as

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

where:

- Q is a finite set of quantum basis states.
- Σ is the input alphabet, which is usually augmented with a right end-marker (e.g., \$).
- δ is a transition function that induces a set of unitary operators $\{U_\sigma : \sigma \in \Sigma \cup \{\$\}\}$; these operators act on the Hilbert space $\mathcal{H}_{|Q|}$ spanned by Q.
- $q_0 \in Q$ is the initial state.
- $F \subseteq Q$ is the set of accepting states.

Operation: An MO-1QFA reads an input word $w = \sigma_1\sigma_2\cdots\sigma_n$ by applying, in sequence, the corresponding unitary operator for each symbol:

$$|\psi\rangle = U_{\sigma_n}\cdots U_{\sigma_1}|q_0\rangle.$$

Only after processing the entire input (typically terminated by the end-marker) is a single projective measurement performed. The measurement projects $|\psi\rangle$ onto the subspace spanned by F; if the projection is nonzero, the input is accepted [19, 6].

Language Acceptance: Due to the requirement that all transformations be unitary and the measurement occurs only once at the end, MO-1QFA are known to recognize exactly the reversible regular languages [**BrodskyPippenger2004**, 6]. In practice, this means that they:

- **Accept**: Regular languages with a reversible structure (e.g., certain periodic languages or those with symmetrical state transitions).
- **Reject**: Non-reversible regular languages, many finite languages, and all non-regular languages such as

$$L_{\text{eq}} = \{a^n b^n \mid n \geqslant 0\}.$$

Key Features and Limitations:

- **Simplicity:** The model's single measurement at the end simplifies the analysis of its evolution.
- **Efficiency:** Although MO-1QFA can be state-efficient, their expressive power is limited.
- **Limitations:** Their acceptance power is strictly contained within the class of regular languages; they cannot recognize languages that require multiple measurements or context-sensitive checks.

Measure-Many Quantum Finite Automaton (MM-1QFA)

Definition 3.1.2 (MM-1QFA)**.** An MM-1QFA is defined as

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

where:

- Q is a finite set of quantum states.
- Σ is the input alphabet, augmented with designated end-markers.
- δ is a transition function that assigns a unitary operator U_σ to each symbol $\sigma \in \Sigma \cup \{\#, \$\}$.
- $q_0 \in Q$ is the initial state.
- The state set Q is partitioned into three disjoint subsets:
  - Q_{acc}: the accepting (halting) states,

- Q_{rej}: the rejecting (halting) states,
- $Q_{non} = Q \backslash (Q_{acc} \cup Q_{rej})$: the non-halting states.

Operation: The MM-1QFA processes the input word symbol by symbol. After each unitary transformation U_σ, an intermediate measurement is performed using the observable defined by the orthogonal decomposition

$$\mathcal{H}_{|Q|} = \text{span}\{|q\rangle : q \in Q_{acc}\} \oplus \text{span}\{|q\rangle : q \in Q_{rej}\} \oplus \text{span}\{|q\rangle : q \in Q_{non}\}.$$

If the outcome falls in Q_{acc} or Q_{rej}, the computation halts with acceptance or rejection, respectively; if the outcome is in Q_{non}, the computation continues with the next symbol [17, 3].

Language Acceptance: MM-1QFA have been shown to accept certain languages with bounded error that are not accepted by MO-1QFA. For instance:

- They can recognize some non-regular patterns such as the balanced language

  $$L_{\text{eq}} = \{a^n b^n \mid n \geqslant 0\}$$

  under bounded-error acceptance conditions, although this comes with restrictions on error bounds [17].

- They offer an exponential state advantage over classical DFA for specific languages (e.g., modular languages $L_{\text{mod}} = \{w \in \Sigma^* \mid |w| \equiv 0 \mod p\}$) [3].

However, due to the reversibility imposed by unitarity and the intermediate measurement process, MM-1QFA still cannot recognize all regular languages and are generally less powerful than their two-way counterparts.

Key Features and Limitations:

- **Intermediate Measurements:** Frequent measurements enable a finer control of state evolution, which can sometimes allow the recognition of languages beyond the reach of MO-1QFA.
- **Space Efficiency:** In certain cases, MM-1QFA achieve an exponential reduction in the number of states compared to classical automata.
- **Limitations:** Despite these advantages, the model's overall computational power is still strictly bounded by regularity constraints and is generally weaker than two-way quantum models.

LQFA

Definition 3.1.3 (LQFA)**.** A LQFA is defined as

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

where:

- Q is a finite set of quantum states.

- Σ is the input alphabet, possibly extended with special markers.
- δ is a transition function that integrates both unitary operations and immediate projective measurements at each step.
- $q_0 \in Q$ is the initial state.
- $F \subseteq Q$ is the set of accepting states.

Operation: In an LQFA, each input symbol is processed in two stages. First, a unitary transformation corresponding to the symbol is applied. Then, a measurement is immediately performed, and the outcome determines whether the automaton halts (by entering an accepting state) or continues processing. This hybrid approach is designed to capture additional probabilistic behaviors without fully relinquishing unitarity [2].

Language Acceptance: The LQFA model typically recognizes a subset of the languages accepted by MM-1QFA. More precisely:

- **Accepts:** Certain regular languages that have an inherent reversible structure. However, due to the immediate measurements, the model may only correctly recognize languages where the measurement outcomes consistently reinforce the desired computation.
- **Rejects:** Some simple regular languages (e.g., languages of the form $a\Sigma^*$) may not be accepted reliably if the measurement collapses the state in an undesired way.

Key Features and Limitations:

- **Measurement-Driven Dynamics:** The interleaving of unitary operations with measurements can lead to lower overall state complexity for specific tasks.
- **Restricted Closure Properties:** The immediate measurement strategy results in a model that does not exhibit the same robust closure properties as MM-1QFA.
- **Expressive Power:** While LQFA can sometimes simulate MM-1QFA behavior, their expressive power remains a proper subset of the languages accepted by more general one-way quantum automata.

Summary of One-Way QFA Standard Models

Each of the standard one-way QFA models discussed exhibits distinct advantages and limitations:

- **MO-1QFA** perform a single measurement at the end of computation, resulting in a simple operational model that recognizes exactly the reversible regular languages.
- **MM-1QFA** incorporate intermediate measurements, enhancing their capability to recognize certain patterns and offering exponential state advantages in some cases, albeit still limited to a subset of regular languages.

- **LQFA** blend unitary transformations with immediate measurements at each step, resulting in a model with unique probabilistic dynamics but with reduced closure properties.

Overall, while one-way QFA models can outperform classical automata in state efficiency for some languages, their language acceptance power remains restricted by the constraints imposed by unitarity and measurement. Advanced models, including hybrid and two-way variants, are required to overcome these limitations.

3.1.2 Hybrid Models

Hybrid models combine classical and quantum components to exploit the advantages of both paradigms. In these models, a classical control unit steers the application of quantum operations, thus enabling the recognition of a wide class of languages while potentially reducing state complexity.

One-way Quantum Finite Automaton with Classical States (1QFAC)

Definition 3.1.4 (1QFAC)**.** A 1QFAC is defined as

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

where:

- S is a finite set of **classical control states**.
- Q is a finite set of quantum basis states.
- Σ is the input alphabet.
- $\delta : S \times \Sigma \to S$ is a classical transition function that updates the classical state.
- $\mu : S \times \Sigma \to \mathcal{U}(\mathcal{H})$ assigns a unitary operator (acting on the quantum state space $\mathcal{H}$) to each pair (s, σ).
- $s_0 \in S$ and $q_0 \in Q$ are the initial classical and quantum states, respectively.
- $F \subseteq S$ is the set of accepting classical states.

Operation: In a 1QFAC, the evolution of the computation is steered by the classical component. At each step, given the current classical state s and input symbol σ, the automaton updates its classical state via $\delta(s, \sigma)$ while simultaneously applying the unitary operator $\mu(s, \sigma)$ to the quantum component. After processing the entire input, a final measurement is performed on the quantum state to determine acceptance; the overall decision is then based on whether the classical state is in F [32]. This two-level approach allows the model to simulate classical deterministic behavior while harnessing quantum parallelism.

Language Acceptance: Due to the classical control, 1QFAC can recognize all regular languages. In some cases, they can also recognize non-regular languages with bounded or one-sided error, leveraging quantum interference for improved succinctness. Notably, 1QFAC have been shown to be exponentially more succinct than classical DFA for certain languages [9].

Key Features and Limitations:

- **Expressiveness:** The hybrid structure enables the simulation of classical automata while allowing for quantum enhancements.
- **Succinctness:** For certain languages, the combined model requires exponentially fewer states than classical models.
- **Error Management:** The interaction between the classical and quantum components necessitates sophisticated error correction and decoherence management techniques.

One-way Quantum Finite Automaton with Classical States (CL-1QFA)

Definition 3.1.5 (CL-1QFA)**.** A CL-1QFA is defined as

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

where:

- Q is a finite set of quantum states.
- Σ is the input alphabet.
- δ is a transition function that assigns unitary operators to input symbols.
- $q_0 \in Q$ is the initial state.
- $\mathcal{L} \subseteq \Sigma^*$ is a **control language** that constrains the allowable sequence of measurement outcomes, thereby guiding the computation.

Operation: In a CL-1QFA, the automaton processes the input by applying the corresponding unitary operators as determined by δ. At specific computation steps, measurements are performed and the resulting sequence of outcomes is required to belong to the control language $\mathcal{L}$. This additional layer acts as a filter, ensuring that only computations consistent with the desired language behavior lead to acceptance [8].

Language Acceptance: CL-1QFA recognize regular languages with bounded error. The imposition of the control language $\mathcal{L}$ not only enforces additional structure on the measurement outcomes but also helps in closing the model under Boolean operations, a property desirable for compositional language theory.

Key Features and Limitations:

- **Boolean Closure:** The constraint imposed by $\mathcal{L}$ renders the model closed under Boolean operations.
- **Expressiveness:** While capable of recognizing all regular languages with bounded error, the added complexity of the control language can increase the computational overhead.
- **Complexity:** Designing and managing the control language $\mathcal{L}$ introduces additional complexity in both the theoretical analysis and practical implementation.

Summary of Hybrid Models

The hybrid models (1QFAC and CL-1QFA) combine classical control with quantum operations:

- **1QFAC** use a classical control unit to direct quantum state transformations, enabling recognition of all regular languages with potential exponential succinctness.
- **CL-1QFA** incorporate a control language that constrains measurement outcomes, ensuring Boolean closure and bounded-error recognition of regular languages.

3.1.3 Enhanced Models

Enhanced models extend the capabilities of standard QFA by permitting more general quantum operations. Such models allow non-unitary evolution, intermediate measurements, and the use of ancillary qubits, thereby capturing a wider range of quantum effects.

Enhanced Quantum Finite Automaton (EQFA)

Definition 3.1.6 (EQFA)**.** An EQFA is defined as

$$M = (\Sigma, Q, \{U_\sigma\}_{\sigma\in\Gamma}, Q_{\text{acc}}, Q_{\text{rej}}, Q_{\text{non}}, q_0),$$

where:

- Σ is the input alphabet, extended to $\Gamma = \Sigma \cup \{\#, \$\}$ (incorporating end-markers).
- Q is a finite set of basis states.
- $\{U_\sigma\}$ is a collection of superoperators—possibly non-unitary—that act on the Hilbert space $\mathcal{H}_{|Q|}$ and model both unitary evolution and irreversible processes.
- $Q_{\text{acc}}, Q_{\text{rej}}, Q_{\text{non}}$ partition Q into accepting, rejecting, and non-halting states, respectively.
- $q_0 \in Q$ is the initial state.

Operation: An EQFA applies, for each input symbol, the corresponding superoperator U_σ followed by a measurement in the basis corresponding to the partition of Q. Unlike standard QFA, measurements can occur at intermediate steps and need not be projective onto strictly unitary subspaces. This flexible framework allows the automaton to simulate irreversible operations and model decoherence effects [23].

Language Acceptance: EQFA can simulate classical finite automata and, under unbounded error, can even recognize certain non-regular languages [20]. Their expressive power is enhanced by the ability to incorporate non-unitary processes; however, this comes at the expense of increased complexity in error analysis.

Key Features and Limitations:

- **Generalized Operations:** Supports arbitrary intermediate measurements and non-unitary evolutions.
- **Expressiveness:** Capable of simulating classical automata and, under specific error conditions, recognizing non-regular languages.
- **Complexity:** The introduction of irreversible operations complicates both the analysis and the practical implementation of error correction.

Open Time Evolution Quantum Finite Automaton (OTQFA)

Definition 3.1.7 (OTQFA)**.** An OTQFA is defined as

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

where:

- Σ is the input alphabet.
- Q is a finite set of quantum states.
- $\delta : \Sigma \to \mathrm{CPTP}(\mathcal{H}_{|Q|})$ is a transition function mapping each symbol to a completely positive, trace-preserving (CPTP) map, modeling decoherence and environmental interactions via Lindblad dynamics.
- $q_0 \in Q$ is the initial state.
- $F \subseteq Q$ is the set of final (accepting) states.

Operation: For an input $w = \sigma_1\sigma_2 \cdots \sigma_n$, the state evolves as

$$\rho' = \delta(\sigma_n) \circ \cdots \circ \delta(\sigma_1)\,(|q_0\rangle\langle q_0|).$$

Acceptance is determined by the projection of ρ' onto the subspace spanned by F. This model captures the effects of noise and decoherence in realistic quantum systems, thereby unifying various one-way QFA models under a common framework [13].

Language Acceptance: OTQFA are capable of recognizing all regular languages and, under certain conditions, even some non-regular languages. However, the open-system dynamics may lead to undecidability issues in certain decision problems.

Key Features and Limitations:

- **Realism:** Models the effects of noise and decoherence inherent in physical quantum systems.
- **Unification:** Provides a general framework that encompasses several one-way QFA models.
- **Undecidability:** Some decision problems become undecidable due to the non-unitary, open-system dynamics.

Ancilla-Based Quantum Finite Automaton (A-QFA)

Definition 3.1.8 (A-QFA)**.** An A-QFA extends the standard MO-1QFA model by incorporating ancillary qubits. It is defined as

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

where the computational Hilbert space is expanded to

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

allowing the automaton to simulate additional quantum resources or classical nondeterminism [23].

Operation: In an A-QFA, the evolution involves both the primary quantum state and the ancillary qubits. The unitary operator δ acts on the composite space, facilitating interference between the main system and the ancilla. At the end of the computation, a global measurement is performed on $\mathcal{H}_{\text{total}}$ to decide acceptance based on whether the resulting state projects onto the subspace corresponding to F.

Language Acceptance: A-QFA can simulate all regular languages with improved efficiency. Moreover, in some configurations, they are capable of recognizing non-regular languages with one-sided error, thus offering a trade-off between resource usage and recognition power.

Key Features and Limitations:

- **Enhanced Expressiveness:** The use of ancilla qubits extends the computational capabilities beyond those of standard MO-1QFA.
- **Succinctness:** Can achieve exponential savings in state complexity for certain languages.
- **Resource Overhead:** The need to manage additional ancillary qubits increases the complexity of both the physical implementation and the theoretical analysis.

Summary of Enhanced Models

Enhanced models extend the standard QFA framework by allowing non-unitary evolution and additional quantum resources:

- **EQFA** support intermediate measurements and non-unitary transitions, thereby simulating classical automata and—under unbounded error—recognizing some non-regular languages.
- **OTQFA** model open-system dynamics via CPTP maps, capturing realistic decoherence while recognizing all regular languages (and some non-regular languages under specific conditions).
- **A-QFA** incorporate ancillary qubits to expand the computational Hilbert space, offering improved state succinctness and extended recognition capabilities.

3.1.4 Advanced Variants

Advanced QFA variants extend the capabilities of one-way and 2QFA models by incorporating restricted bidirectional motion or by processing multiple symbols simultaneously. These models bridge the gap between the expressive power of strictly one-way QFA and full 2QFA, often achieving enhanced recognition capabilities with bounded error while retaining some of the simplicity of one-way motion.

One-and-a-half-way Quantum Finite Automaton (1.5QFA)

Definition 3.1.9 (1.5QFA)**.** A 1.5QFA is defined as

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

where:

- Q is a finite set of quantum basis states.
- Σ is the input alphabet.
- δ is a transition function that, besides inducing unitary operations, permits the tape head to move primarily rightward while allowing occasional stationary moves or limited leftward moves.
- $q_0 \in Q$ is the initial state.
- $F \subseteq Q$ is the set of accepting states.

Operation: The automaton processes the input from left to right. Unlike standard one-way QFA, a 1.5QFA is allowed limited backward (leftward) motion or to remain stationary on a symbol, thereby providing a restricted form of bidirectional scanning. This extra flexibility enables more refined control over state transitions and can improve recognition of certain non-regular patterns under bounded error [17].

Language Acceptance: A 1.5QFA accepts languages with bounded error probability. The model's restricted bidirectional motion enables it to recognize some non-regular languages that are beyond the reach of strictly one-way QFA, although its overall recognition power remains below that of full 2QFA models.

Key Features and Limitations:

- **Key Features:**
  - Enhanced computational power relative to strictly one-way QFA.
  - Ability to recognize certain non-regular languages under bounded error conditions.
  - Improved flexibility in managing state transitions via limited backward or stationary moves.
- **Limitations:**
  - The restricted backward motion still limits the overall expressive power compared to full two-way models.
  - Increased complexity in head movement can pose practical implementation challenges.

Multi-letter Quantum Finite Automaton (MLQFA)

Definition 3.1.10 (MLQFA)**.** An MLQFA processes the input in blocks of k symbols and is defined as

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

where:

- Q is a finite set of quantum states.
- Σ is the input alphabet.
- The transition function δ depends on k-length substrings rather than on individual symbols; for each block of k symbols, a corresponding unitary operator is applied [4].
- $q_0 \in Q$ is the initial state.
- $F \subseteq Q$ is the set of accepting states.

Operation: The MLQFA partitions the input into consecutive blocks of k symbols. For each block, it applies a unitary operator corresponding to the entire k-letter substring. This block-wise (or parallel) processing allows the automaton to capture dependencies over longer segments of the input, thereby simulating a form of multi-head or parallel processing [4].

Language Acceptance: MLQFA accept languages by processing multiple symbols simultaneously. This capability enables them to recognize some complex patterns, including certain context-sensitive languages, under bounded error. However, the approach comes at the cost of an exponential increase in state complexity with respect to the block size k.

Key Features and Limitations:

- **Key Features:**
  - Simulates multi-head or parallel processing by handling blocks of symbols.
  - Enhanced ability to capture long-range dependencies in the input.
  - Potential to recognize more complex language classes compared to single-letter QFA.
- **Limitations:**
  - Exponential growth in state complexity with increasing block size k.
  - Increased computational overhead in designing and managing block-based unitary operators.

Summary of Advanced Variants

The advanced QFA variants discussed above offer distinct trade-offs:

- **1.5QFA** extend the traditional one-way model by incorporating restricted bidirectional motion, which improves recognition of some non-regular languages under bounded error.
- **MLQFA** process input in blocks, simulating a parallel processing model that can capture more complex patterns, though with increased state complexity.

Overall, these advanced variants bridge the gap between the simplicity of one-way QFA and the expressive power of full 2QFA, while introducing additional challenges in terms of implementation and computational overhead.

Detailed Hierarchy of One-Way QFA Models

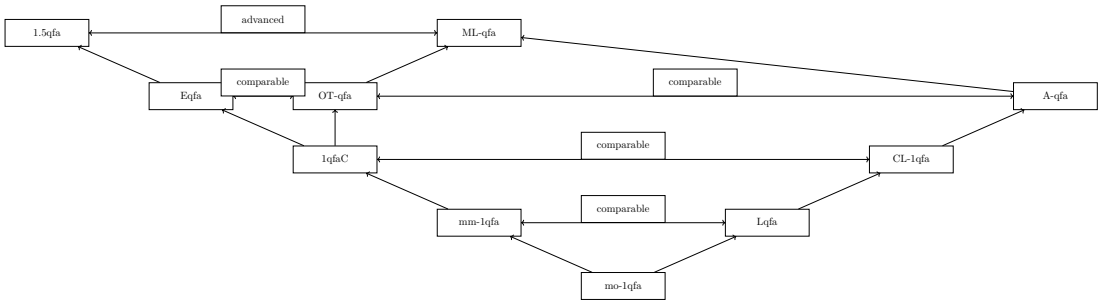


Figure 3.1: Detailed hierarchy of one-way QFA models by expressive power. Vertical arrows indicate a strict increase in expressive power from lower to higher levels, while horizontal arrows denote models with comparable expressive capabilities.

3.2 Two-way Quantum Finite Automaton (2QFA)

2QFAs extend one-way models by permitting bidirectional head movement, thereby enhancing their computational power. In 2QFA models the head may move left, right, or remain stationary, which allows the automaton to exploit quantum interference over multiple passes of the input. In the following subsections, we detail the standard models, hybrid variants, and multihead/tape extensions. For each model, we provide a formal definition, describe its operation, discuss language acceptance properties, and list key features and limitations.

3.2.1 Standard Models

Two-way Quantum Finite Automaton (2QFA)

Definition 3.2.1 (2QFA)**.** A 2QFA is defined as

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

where:

- Q is a finite set of quantum states, partitioned into accepting states Q_{acc}, rejecting states Q_{rej}, and non-halting states $Q_{\text{non}} = Q \backslash (Q_{\text{acc}} \cup Q_{\text{rej}})$.

- Σ is the input alphabet, extended with left and right end-markers (e.g., # and \$).
- $\delta : Q \times \Gamma \to \mathbb{C}^{Q \times \{\leftarrow, \downarrow, \rightarrow\}}$ is the transition function (with $\Gamma = \Sigma \cup \{\#, \$\}$) that specifies both the unitary evolution and the head movement.
- $q_0 \in Q$ is the initial state.

Operation: Given an input $w = \sigma_1 \sigma_2 \ldots \sigma_n$, the automaton repeatedly applies the corresponding unitary operators while moving its head left, right, or remaining stationary as dictated by δ. Intermediate measurements (or deferred measurement techniques) are employed so that quantum interference can be exploited across multiple passes of the input.

Language Acceptance: A 2QFA is capable of accepting non-regular languages with bounded error. For example, it has been shown that the language

$$L_{\text{eq}} = \{a^n b^n \mid n \geqslant 0\}$$

can be recognized with bounded error in linear time [17, 30].

Key Features and Limitations:

- **Key Features:**
  - Ability to recognize non-regular languages such as L_{eq} with bounded error.
  - Exploits quantum interference, which can lead to exponential state savings over classical two-way automata.
- **Limitations:**
  - Requires quantum registers whose size may scale with the input length.
  - Error management and decoherence pose significant challenges for practical implementations.

3.2.2 Hybrid Models

Two-way Quantum Classical Finite Automaton (2QCFA)

Definition 3.2.2 (2QCFA)**.** A 2QCFA is defined as

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

where:

- S is a finite set of classical control states.
- Q is a finite set of quantum states.
- Σ is the input alphabet, augmented with appropriate end-markers.
- Θ assigns quantum operations (unitary transformations or measurements) based on the current classical state and input symbol.

- $\delta : S \times \Sigma \to S$ governs the classical state transitions and head movements.
- $s_0 \in S$ and $q_0 \in Q$ are the initial classical and quantum states, respectively.
- S_{acc} and S_{rej} are the sets of accepting and rejecting classical states.

Operation: In a 2QCFA, the classical component directs the head movement and determines when to invoke quantum operations via Θ. Adaptive measurements are performed based on the current classical state, allowing the automaton to decide acceptance using a small quantum register.

Language Acceptance: 2QCFA can recognize certain non-regular languages, such as L_{eq} and palindromes, in polynomial time while using only a constant number of quantum states [2].

Key Features and Limitations:

- **Key Features:**
  - Combines classical control with quantum computation for enhanced efficiency.
  - Recognizes non-regular languages in polynomial time with a minimal quantum register.
- **Limitations:**
  - Synchronization between classical and quantum components increases design complexity.
  - The decidability of the equivalence problem for 2QCFA remains an open issue.

3.2.3 Multihead/Tape Extensions

Two-way Two-Tape Quantum Classical Finite Automaton (2TQCFA)

Definition 3.2.3 (2TQCFA)**.** A 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:

- Σ_1 and Σ_2 are the input alphabets for the two tapes.
- The remaining components $(S, Q, \Theta, \delta, s_0, q_0, S_{\text{acc}}, S_{\text{rej}})$ are defined analogously to those in a 2QCFA.
- The transition function δ synchronizes the head movements on both tapes.

Operation: The automaton simultaneously reads from two tapes, allowing it to verify language properties that involve correlations between substrings on different tapes. This parallel processing can be exploited to recognize languages like

$$L = \{a^n b^n c^n \mid n \geqslant 1\},$$

by comparing corresponding symbols from the two tapes [34].

Language Acceptance: 2TQCFA accept languages that require cross-tape comparisons and correlation between substrings, thereby extending the recognition power beyond that of single-tape models.

Key Features and Limitations:

- **Key Features:**
  - Enhanced recognition power compared to single-tape 2QCFA.
  - Efficient processing of languages with interdependent substrings.
- **Limitations:**
  - Increased complexity in synchronizing head movements across two tapes.
  - Higher error-correction overhead due to parallel tape processing.

k-Tape Quantum Classical Finite Automaton (kTQCFA)

Definition 3.2.4 (kTQCFA)**.** A kTQCFA generalizes the two-tape model to k tapes. It 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:

- Each tape has its own alphabet Σ_i for $i = 1, 2, \ldots, k$.
- The transition function δ coordinates the head movements across all k tapes.
- The remaining components are defined similarly to those in a 2QCFA.

Operation: The automaton processes input from k tapes simultaneously. This multi-tape configuration enables efficient recognition of languages with complex structural dependencies. For instance, languages such as

$$L = \{a^n b^{n^2} \mid n \geqslant 1\}$$

can be recognized more effectively by performing parallel comparisons across the tapes [34].

Language Acceptance: kTQCFA are designed to accept languages with intricate interdependencies that benefit from simultaneous multi-tape processing, thereby extending the expressive power of quantum finite automata.

Key Features and Limitations:

- **Key Features:**
  - Potential for exponential state savings compared to classical multi-tape automata.
  - Greater expressive power for recognizing languages with complex structural dependencies.

- **Limitations:**
  - Increased complexity and overhead with the number of tapes k.
  - Synchronization of multiple tape heads becomes progressively challenging.

Summary of 2QFAs Models

- **Standard 2QFA**: Allow unrestricted bidirectional head movement with full quantum control. They can recognize non-regular languages such as L_{eq} with bounded error, though they require quantum registers that scale with the input.
- **Hybrid 2QCFA**: Combine classical control with a small quantum register. These automata recognize non-regular languages (e.g., L_{eq} and palindromes) in polynomial time while using only a constant number of quantum states.
- **Multihead/Tape Extensions (2TQCFA and kTQCFA)**: Extend the two-way model to multiple tapes, enhancing recognition power for languages with complex structural dependencies at the cost of increased synchronization and error-correction complexity.

Detailed Hierarchy of 2QFAs Models

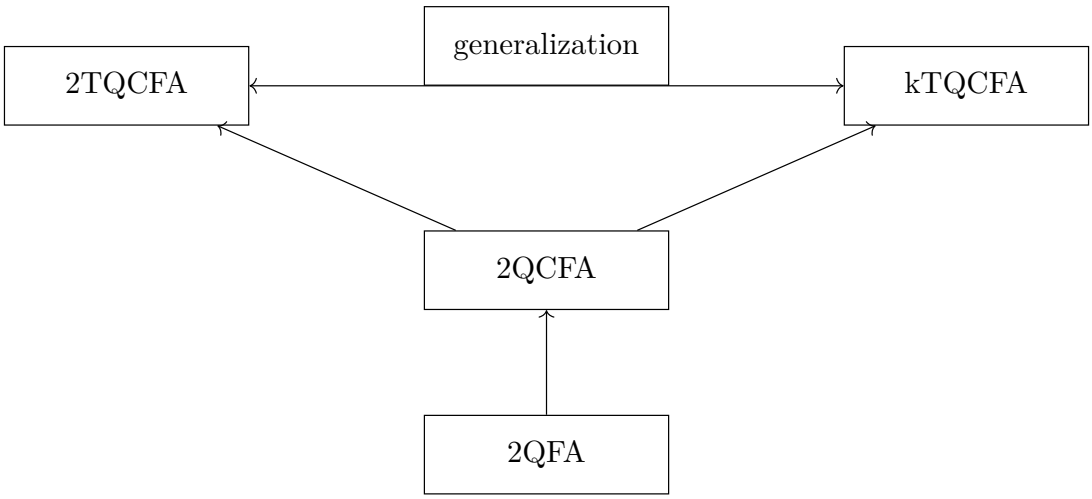


Figure 3.2: Detailed hierarchy of 2QFA models by expressive power. Vertical arrows indicate an increase in expressive power from standard to hybrid and then to multi-tape extensions, while the horizontal arrow denotes that kTQCFA generalize 2TQCFA.

3.3 Interactive Quantum Automata

Interactive quantum automata integrate communication between a resource-limited quantum verifier and an (often unbounded) prover. These models extend standard QFA by introducing interactive protocols where the verifier and prover exchange quantum messages over multiple rounds.

Quantum Interactive Proof (QIP) Systems

Definition 3.3.1 (QIP System)**.** A QIP system consists of a polynomial-time quantum verifier V and an unbounded quantum prover P who exchange messages via a quantum channel. The verifier is typically modeled as a QFA with limited memory. A system with k rounds of interaction is denoted as QIP(k) [22, 33].

Operation: The verifier processes an input w by alternating between local quantum operations and rounds of communication with the prover. The overall transition is governed by a function

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

where Γ is the communication alphabet. After the prescribed rounds of interaction, the verifier performs a measurement on its final state to decide acceptance.

Language Acceptance: A QIP system accepts an input w if the verifier's final measurement yields an accepting outcome with probability exceeding a specified threshold (typically above $\frac{1}{2}$), thereby ensuring bounded-error recognition of the language.

Key Features and Limitations:

- **Key Features:**
  - QIP systems have been shown to be equivalent in power to PSPACE in the classical setting.
  - QFA-based verifiers (e.g., using 2QFA) can recognize languages beyond the regular class with bounded error.
- **Limitations:**
  - Precise control of quantum communication channels is required.
  - The verifier's state complexity may scale with the input for more complex languages.

Quantum Merlin-Arthur Proof (QMIP) Systems

Definition 3.3.2 (QMIP System)**.** QMIP systems extend QIP by involving multiple non-communicating provers. Formally, a QMIP system with k provers is denoted as QMIP(k). In these systems, the verifier receives quantum witness states from each prover and performs a quantum verification procedure [26, 31].

Operation: The verifier, typically modeled as a 2QFA-based machine, receives quantum witness states from the provers and processes them using a quantum circuit. A final measurement is performed on the verifier's state to decide acceptance. The provers do not communicate with each other, and the verification process is non-adaptive.

Language Acceptance: A QMIP system accepts an input if the combined verification procedure yields an accepting outcome with probability above a predetermined threshold, thereby ensuring bounded-error recognition of the target language.

Key Features and Limitations:

- **Key Features:**
  - QMIP systems capture complexity classes such as MIP* and are strictly more powerful than single-prover QIP systems.
  - They can recognize languages (e.g., the palindrome language $L_{\text{pal}} = \{ww^R \mid w \in \{0,1\}^*\}$) with exponential state savings compared to classical multi-prover interactive proofs.
- **Limitations:**
  - Managing entanglement among multiple provers complicates the verification process.
  - Practical implementations are highly sensitive to noise and decoherence.

Summary of Interactive Quantum Automata Models

The interactive quantum automata models discussed above demonstrate the following trade-offs:

- **QIP Systems:**
  - Employ a single prover with a polynomial-time quantum verifier.
  - Are capable of recognizing languages with bounded error, leveraging quantum communication.
- **QMIP Systems:**
  - Extend QIP by using multiple non-communicating provers.
  - Offer enhanced verification power, capturing complexity classes such as MIP*, at the cost of increased complexity in managing entanglement and communication.

Graphical Comparison of Interactive Quantum Automata Models

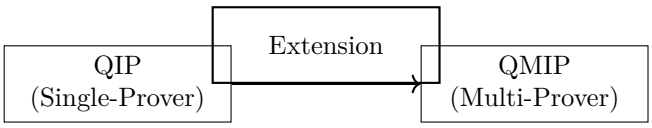


Figure 3.3: Graphical comparison of interactive quantum automata models. QMIP systems extend QIP systems by incorporating multiple non-communicating provers, thereby enhancing verification power.

4. Comparative Analysis

The literature on QFA reveals a diverse landscape of computational models, each tailored to balance quantum resources with expressive power. At the most fundamental level, one-way QFAs such as MO-1QFA and MM-1QFA offer elegant, conceptually simple frameworks. For instance, the MO-1QFA model—employing a single measurement at the end—boasts straightforward dynamics that, however, limit its language recognition capabilities to reversible regular languages. In contrast, the MM-1QFA interweaves unitary evolution with intermediate measurements, thereby extending its acceptance range to include a larger subset of regular languages and, in some cases, even non-regular languages under bounded error conditions. This increase in expressive power, however, is not without cost; the integration of frequent measurements adds complexity in terms of error management and circuit design.

Hybrid models, such as 1QFAC and CL-1QFA, seek to blend classical control mechanisms with quantum processing. The 1QFAC model, for example, leverages classical state control to effectively simulate deterministic finite automata, thereby recognizing all regular languages and sometimes even certain non-regular languages with significant state savings. Meanwhile, the CL-1QFA employs control languages that guide the measurement outcomes, ensuring desirable closure properties under Boolean operations. These models illustrate a recurring theme: as expressive power increases, so does the need for intricate coordination between quantum and classical elements.

Enhanced models push these boundaries further. EQFA and OT-QFA introduce non-unitary evolution through superoperators or completely positive trace-preserving (CPTP) maps, which not only enable the simulation of classical automata but also, in some cases, facilitate the recognition of non-regular languages. Similarly, the A-QFA model uses ancilla qubits to simulate additional quantum resources, providing a pathway to overcome the inherent limitations of simpler QFA models. However, the richer dynamics offered by these enhanced models come at the price of increased resource requirements and greater susceptibility to decoherence and noise.

Turning to two-way QFAs, models such as 2QFA and 2QCFA leverage bidirectional head movement to harness quantum interference more effectively. The 2QFA, with its unrestricted movement, can recognize certain non-regular languages (e.g., $\{a^n b^n\}$) in linear time under bounded error, yet it demands quantum registers that grow with the input size. On the other hand, the hybrid 2QCFA mitigates some of this cost by integrating classical control with a small quantum register. This trade-off allows 2QCFA to recognize complex languages—such as palindromes—using only a constant number of quantum states, albeit at the expense of a more complicated synchronization mechanism between the classical and quantum components.

Multihead and multitape extensions, exemplified by 2TQCFA and kTQCFA, further amplify computational power by enabling parallel processing across multiple input streams. These models are particularly effective for languages characterized by intricate

structural dependencies. While they offer substantial gains in expressiveness, their increased complexity in synchronizing multiple tape heads and managing error correction represents a significant engineering challenge.

Interactive models, encompassing QIP and QMIP systems, introduce an additional layer of computational interplay. QIP systems couple a resource-limited quantum verifier with a single unbounded prover, thereby linking QFA models to complexity classes such as PSPACE. QMIP systems extend this framework to incorporate multiple non-communicating provers, a development that not only increases verification power but also opens the door to recognizing languages beyond the reach of single-prover systems. The practical implementation of such systems, however, is complicated by the need to manage quantum entanglement and the inherent sensitivity of the communication channels.

In summary, the spectrum of QFA models represents a delicate balance: simpler one-way models achieve low resource overhead at the expense of expressive power, while hybrid, enhanced, two-way, and interactive models incrementally increase language recognition capabilities—albeit with concomitant increases in state complexity, error management demands, and implementation challenges.

Schematic Comparative Overview

Model Category	**Expressive Power (Languages Accepted)**
One-way QFAs	MO-1QFA: Reversible regular MM-1QFA: Larger subset of regular (+some non-regular) LQFA: Trade-off between measurement overhead and expressiveness
Hybrid Models	1QFAC: All regular (sometimes non-regular) CL-1QFA: Enhanced closure properties
Enhanced Models	EQFA/OT-QFA: Simulate classical automata, recognize some non-regular l A-QFA: Additional quantum resources via ancilla qubits
Two-way QFAs	2QFA: Some non-regular languages (e.g., $\{a^n b^n\}$) 2QCFA: Complex languages with constant quantum state usage
Multihead/Tape Extensions	2TQCFA/kTQCFA: Languages with intricate dependencies
Interactive Models	QIP: Bounded-error languages (linked to PSPACE) QMIP: Languages beyond single-prover capabilities (e.g., L_{pal})

Table 4.1: Comparative overview of QFA models in terms of expressive power and resource trade-offs.

Graphical Comparison of QFA Models

Summary of Promising Models for Future Research

Among the diverse models examined, the hybrid 2QCFA and the interactive QMIP systems appear particularly promising for future developments. The 2QCFA, by combining robust classical control with a minimal quantum component, offer an attractive balance between expressive power and practical resource constraints. Similarly, QMIP systems, despite their inherent complexity, pave the way for leveraging multiprover interactions to tackle verification problems beyond the reach of traditional single-prover systems.

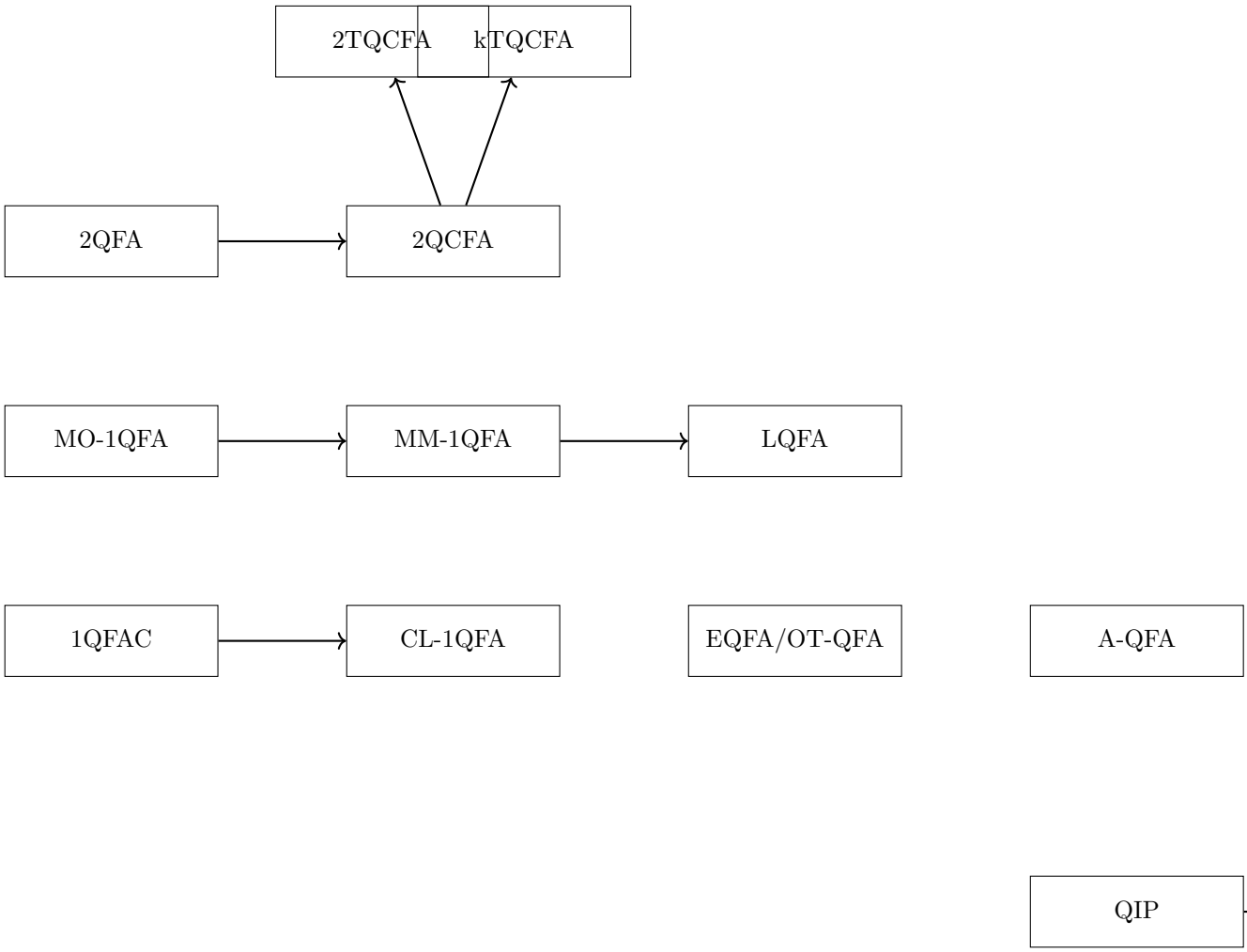


Figure 4.1: Graphical comparison of QFA models. Horizontal and vertical arrows indicate extensions in expressive power and additional resource requirements. Models higher in the diagram are generally more expressive, while interactive models form an orthogonal extension capturing advanced computational paradigms.

Overall, this taxonomy not only reconciles previous inconsistencies in QFA definitions and capabilities but also provides a clear roadmap for exploring new frontiers in quantum automata theory. As quantum hardware matures, these models will be instrumental in designing efficient quantum algorithms and automata implementations that fully harness the potential of quantum computing.

5. Conclusion

This thesis has established a unified framework for the analysis of quantum finite automata by rigorously formalizing both classical automata models and their quantum counterparts. The work began with precise definitions and characterizations of classical models such as DFAs, NFAs and PFAs, and then extended these concepts to a variety of QFAs models. In doing so, a systematic taxonomy was developed that organizes one-way models (e.g., MO-1QFAs, MM-1QFAs, 1QFACs) and two-way models (e.g., 2QFAs, 2QCFAs) in terms of their state complexity, language recognition capabilities, and error bounds.

The research results demonstrate that quantum finite automata can offer significant advantages over classical models, such as exponential state savings and improved recognition of certain non-regular languages under bounded error. Detailed comparisons of closure properties, decidability issues, and the effects of decoherence have been provided, offering quantitative insights that can inform both theoretical investigations and practical applications. These outcomes contribute a solid formal foundation to the field and may serve as a basis for designing efficient quantum algorithms and optimizing quantum circuit implementations.

Future work should address several promising directions. One key area is the rigorous exploration of the equivalence between quantum automata and quantum circuit models. Investigations into whether quantum automata can be compiled into quantum circuits (and vice versa) will deepen our understanding of quantum computational processes. Additional research is needed to refine error-correction techniques for hybrid models and to extend the taxonomy to cover interactive, multi-tape, and resource-constrained quantum systems. Addressing the decidability of equivalence problems and developing practical compilation techniques remain important challenges for advancing both theory and implementation in quantum computing.

Abbreviations

1.5QFA	One-and-a-half-way Quantum Finite Automaton
1QFAC	One-way Quantum Finite Automaton with Classical States
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
CFA	Classical Finite Automaton
CFL	Context-Free Language
CL-1QFA	One-way Quantum Finite Automaton with Classical States
CNOT	Controlled-NOT
CSL	Context-Sensitive Language
DFA	Deterministic Finite Automaton
EQFA	Enhanced Quantum Finite Automaton
GHZ	Greenberger-Horne-Zeilinger State
gQFA	Generalized Quantum Finite Automaton
kTQCFA	k-Tape Quantum Classical Finite Automaton
LQFA	Latvian Quantum Finite Automaton
MLQFA	Multi-letter Quantum Finite Automaton
MM-1QFA	Measure-Many Quantum Finite Automaton
MO-1QFA	Measure-Once Quantum Finite Automaton
NFA	Nondeterministic Finite Automaton

OTQFA	Open Time Evolution Quantum Finite Automaton
PDA	Pushdown Automaton
PFA	Probabilistic Finite Automaton
POVM	Positive Operator-Valued Measure
QFA	Quantum Finite Automaton
QIP	Quantum Interactive Proof
QMIP	Quantum Merlin-Arthur Proof
Qubit	Quantum Bit
REG	Regular Language
TM	Turing Machine

Bibliography

[1] 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.

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

[3] 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.

[4] Aleksandrs Belovs, Ansis Rosmanis, and Juris Smotrovs. "Multi-letter quantum finite automata: decidability and complexity". In: *International Conference on Unconventional Computation* (2007), pp. 48–59.

[5] 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.

[6] Alberto Bertoni and Marco Carpentieri. "Regular languages accepted by quantum automata". In: *Information and Computation* 165.2 (2001), pp. 174–182.

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

[8] Alberto Bertoni, Carlo Mereghetti, and Beatrice Palano. "Quantum computing: 1-way quantum automata". In: *International Conference on Developments in Language Theory* (2003), pp. 1–20.

[9] Maria Paola Bianchi, Carlo Mereghetti, and Beatrice Palano. "Size lower bounds for quantum automata". In: *Theoretical Computer Science* 551 (2014), pp. 102–115.

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

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

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

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

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

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

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

[17] 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.

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

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

[20] 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.

[21] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.

[22] Harumichi Nishimura and Tomoyuki Yamakami. "An application of quantum finite automata to interactive proof systems". In: *Journal of Computer and System Sciences* 75.4 (2009), pp. 255–269.

[23] Kathrin Paschen. "Quantum finite automata using ancilla qubits". In: *Technical Report, Karlsruhe University* (2000).

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

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

[26] Oksana Scegulnaja-Dubrovska, Lelde Lāce, and Rūşinš Freivalds. "Postselection finite quantum automata". In: *International Conference on Unconventional Computation* (2010), pp. 115–126.

[27] Peter W Shor. "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer". In: *SIAM Review* 41.2 (1999), pp. 303–332.

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

[29] 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.

[30] 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.

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

[32] Shenggen Zheng, Lvzhou Li, and Daowen Qiu. "One-way quantum finite automata with classical states". In: *Quantum Information Processing* 11.6 (2012), pp. 1501–1521.

[33] Shenggen Zheng, Daowen Qiu, and Jozef Gruska. "Power of the interactive proof systems with verifiers modeled by semi-quantum two-way finite automata". In: *Information and Computation* 241 (2015), pp. 197–214.

[34] 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