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 merges classical computational models with quantum mechanics to explore the capabilities and limitations of quantum systems. Despite its theoretical potential, the field remains fragmented by inconsistent notation, ambiguous model definitions, and a lack of systematic comparisons between classical and quantum finite automata (QFA). This thesis addresses these gaps by establishing a unified framework for analyzing classical and quantum finite automata, emphasizing standardized definitions, structural parallels, and rigorous computational property evaluations.

We first formalize classical finite automata—including deterministic (DFA), nondeterministic (NFA), probabilistic (PFA), and two-way variants—to establish foundational concepts. Building on this, we systematically catalog quantum finite automata models, categorizing one-way hybrids such as the one-way quantum finite automaton with classical states (1QFAC), two-way models such as the two-way quantum finite automaton (2QFA), and enhanced variants including the enhanced quantum finite automaton (EQFA) and quantum finite automaton with open time evolution (OT-QFA). For each model, we analyze formal definitions, computational dynamics, language acceptance criteria, expressive power, and closure properties, while contextualizing relationships through comparative studies.

By synthesizing results from foundational and contemporary literature, this work resolves ambiguities in prior formulations, such as quantum-classical state hybrids, and identifies open problems in equivalence checking, pumping lemma extensions, and quantum advantage thresholds. Key contributions include a hierarchical taxonomy of models, classifications of decidable vs. undecidable problems, and insights into size-space efficiency trade-offs. This thesis aims to serve as a foundational reference for researchers and provides a methodology to guide the development of new quantum automata 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 (DFA), non-deterministic (NFA), probabilistic (PFA), and two-way variants—have long served as the bedrock of formal language theory, providing mathematically rigorous frameworks for analyzing computational complexity and decidability. Yet, quantum automata introduce 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, how quantum parallelism enables certain QFA variants, such as the measure-many one-way quantum finite automaton (MM-1QFA), to recognize languages with exponentially fewer states than their classical counterparts [22]. Practically, as quantum hardware advances, understanding the minimal resources required to implement quantum automata becomes critical for designing efficient algorithms and error-correcting schemes. Yet, the field's progress has been hindered by ambiguities in model definitions. For example, early quantum automata models like the measure-once (MO-1QFA) and MM-1QFA were defined with differing acceptance criteria, leading to confusion about their relative computational power [20]. 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 [41]. These inconsistencies obscure the true capabilities of quantum models and hinder cross-disciplinary collaboration.

Central to this thesis is the observation that no single document systematically 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, the expressive power of two-way quantum finite automata (2QFA) surpasses that of classical two-way automata, yet the conditions under which this advantage manifests, such as the role of quantum interference in recognizing non-regular languages, remain underexplored in a comparative context [39]. 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 quantum automata with mixed states or the precise trade-offs between quantum entanglement and space efficiency [15].

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 quantum finite automata with open time evolution, OT-QFA) requires 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 deterministic finite automata) 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 quantum automata complicates efforts to classify their language recognition capabilities, a challenge this thesis tackles through comparative analysis of closure properties and equivalence criteria [1].

To address these challenges, this thesis employs a structured methodology. It begins by grounding the discussion in classical automata theory, revisiting deterministic (DFA), nondeterministic (NFA), probabilistic (PFA), and two-way variants to establish foundational concepts. Building on this, it systematically explores quantum models, from early variants such as MO-1QFA [22] and MM-1QFA [20] to advanced hybrids such as 1QFAC [41] and enhanced quantum finite automata (EQFA). Each model is analyzed through multiple dimensions: formal definitions are standardized, acceptance criteria are scrutinized, and computational dynamics—such as the role of measurement timing or quantum-classical state interactions—are dissected. By juxtaposing classical and quantum models across these dimensions, the thesis uncovers patterns in their expressive power, such as the ability of certain QFA variants to recognize non-regular languages with bounded error, a feat impossible for classical finite automata [3].

A key contribution of this work is its hierarchical taxonomy of automata models, which organizes classical and quantum automata into a coherent structure based on their computational features. This taxonomy reveals, for example, that two-way quantum automata (2QFA) occupy a higher complexity class than their one-way counterparts, while quantum automata with classical states (1QFAC) occupy an intermediate position, bridging purely quantum and classical models [39]. The taxonomy also highlights open problems, such as the precise relationship between quantum finite automata with ancilla qubits (A-QFA) and generalized quantum finite automata (gQFA), or the conditions under which quantum automata outperform probabilistic models in language recognition [15]. By mapping these relationships, the thesis provides a roadmap for future research on quantum advantage thresholds and the minimal resource requirements for quantum-enhanced computation.

The thesis is organized to guide the reader through increasingly complex layers of analysis. Following this introduction, Chapter 2 consolidates foundational concepts from classical automata theory and quantum mechanics, providing a unified background for subsequent discussions. Chapter 3 forms the core of the work, presenting a comprehensive catalog of quantum automata models. Each model is formally defined, analyzed for computational dynamics, and compared to classical and quantum alternatives. Chapter 4 synthesizes these analyses, evaluating expressive power, closure properties, and decidability between models. Finally, Chapter 5 concludes by reflecting on the thesis's contributions and describing directions for future research, such as solving open questions in equivalence checking for quantum automata [21], 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 framework. By standardizing definitions, clarifying model relationships, and identifying open challenges, it provides both a reference for researchers and a methodology for advancing the field. As quantum computing moves 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 enables 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. Deterministic finite automata (DFAs), non-deterministic finite automata (NFAs), probabilistic finite automata (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 their limitations in handling context-free or stochastic languages. The analysis draws on foundational works such as Hopcroft et al. [16], which formalized the equivalence between DFAs and NFAs, and Rabin's seminal work on probabilistic automata [31], which expanded the class of recognizable languages through probabilistic acceptance criteria.

The discourse then transitions to quantum mechanical principles essential for quantum computation. Key concepts such as qubit representation, quantum superposition, and entanglement are contextualized within computational frameworks, emphasizing their departure from classical bit-based processing. The measurement postulate and its implications for probabilistic outcomes are discussed in relation to quantum state collapse, a critical distinction from classical probabilistic models. These principles are synthesized with insights from Nielsen and Chuang's definitive text on quantum computation [26], which provides the mathematical formalism for quantum operations.

The chapter's structure is designed to mirror the hierarchical taxonomy developed in later chapters. By first rigorously defining classical models and their limitations, followed by an exposition of quantum principles and their computational implications, the groundwork is laid for analyzing hybrid models such as the one-way quantum finite automaton with classical states (1QFAC) [41]. Each section deliberately connects theoretical constructs to practical considerations, such as the role of decoherence in open quantum systems [11] and its impact on automata design. This approach ensures that subsequent discussions of quantum automata variants are rooted in both mathematical rigor and physical realizability.

2.1 Classical Finite Automata

Finite automata form the cornerstone of formal language theory, providing mathematical frameworks to analyze computational limits and language recognition capabilities. This section systematically examines deterministic, nondeterministic, probabilistic, and two-way variants, emphasizing their structural relationships and computational boundaries.

2.1.1 Shared Foundations

The study of automata begins with foundational concepts in formal language theory, pioneered by figures such as Stephen Kleene [19], Noam Chomsky [13], Alan Turing [16], and Michael Rabin [31]. Their work established the mathematical scaffolding for analyzing computational models. Below, we elaborate on core definitions, operations, and language classifications, augmented with practical examples and formal specifications.

Alphabets and Strings

An *alphabet* Σ is a non-empty, finite set of symbols. For instance:

- The **binary alphabet** $\Sigma = \{0, 1\}$ is foundational in digital computing [16].
- The **ASCII alphabet** Σ_{ASCII} contains 128 characters for text encoding [12].

A *string* (or *word*) w over Σ is a finite sequence of symbols $a_1 a_2 \ldots a_n$, where $a_i \in \Sigma$. The **length** of w, denoted $\|w\|$, is the number of symbols in w. The **empty string** ϵ has $\|\epsilon\| = 0$ [16].

Examples:

- For $\Sigma = \{a, b\}$, $w = aba$ has $\|w\| = 3$ [16].
- The string $w = \epsilon$ represents "no input" in automata models [16].

Key string operations include:

- **Reversal**: w^R reverses the order of symbols (e.g., $(abc)^R = cba$) [16].
- **Substring**: A string v is a substring of w if $w = xvy$ for some x, y [16].

Languages and Operations

A *language* L is a subset of Σ^*, the **Kleene closure** of Σ, defined as the set of all finite strings over Σ:

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

[19]

Language Operations:

1. **Concatenation**: For languages L_1 and L_2,

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

   Example: If $L_1 = \{a, ab\}$ and $L_2 = \{b, ba\}$, then $L_1 \cdot L_2 = \{ab, aba, abb, abba\}$.[16]

2. **Union/Intersection**:

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

   [16]

3. **Kleene Star**:

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

   [19] *Example*: If $L = \{0, 1\}$, then L^* includes all binary strings, including ϵ [16].

4. **Complement**: $\overline{L} = \Sigma^* \backslash L$ [16].

5. **Homomorphism**: A function $h : \Sigma^* \to \Gamma^*$ that replaces symbols (e.g., $h(a) = 01$ maps $a \to 01$) [16].

6. **Inverse Homomorphism**: $h^{-1}(L) = \{w \mid h(w) \in L\}$. [16]

Language Categories

Languages are classified by their recognition models and structural complexity:

1. **Regular Languages (REG)**: Recognized by *deterministic finite automata (DFA)*, *nondeterministic finite automata (NFA)*, or *regular expressions* [16]. *Example*: $L = \{w \in \{a, b\}^* \mid w \text{ contains } aba\}$ is regular [16].

2. **Context-Free Languages (CFL)**: Recognized by *pushdown automata (PDA)* [13, 16]. *Example*: $L_{\text{pal}} = \{ww^R \mid w \in \{a, b\}^*\}$ (palindromes) [13].

3. **Context-Sensitive Languages (CSL)**: Recognized by *linear-bounded automata* [13, 16]. *Example*: $L = \{a^n b^n c^n \mid n \geqslant 1\}$ [13].

4. **Recursively Enumerable Languages (Type-0)**: Recognized by *Turing machines*, formalized by Alan Turing to define computability limits [16]. *Example*: The Halting Problem's language [16].

5. **Stochastic Languages**: Recognized by *probabilistic finite automata (PFA)* with bounded error [31]. *Example*: $L_{\text{eq}} = \{a^n b^n \mid n \geqslant 1\}$ is stochastic but not regular. A PFA can accept this language with probability $\geqslant \frac{2}{3}$ for valid strings and $\leqslant \frac{1}{3}$ for invalid ones, leveraging probabilistic state transitions [31].

Class	**Recognizer**	**Example**	**Closure Properties**	**Pumping Lemma**
Regular (REG)	DFA/NFA	$\{w \mid w \text{ contains } aba\}$	Union, Concat, Kleene*	xyz with $\lvert xy\rvert \leqslant p$
Context-Free (CFL)	PDA	Palindromes	Union, Kleene*	$uvxyz$ with $\lvert vxy\rvert \leqslant p$
Context-Sensitive (CSL)	LBA	$\{a^n b^n c^n\}$	Intersection, Complement	-
Recursively Enumerable (Type-0)	Turing Machine	Halting Problem	All operations	-
Stochastic	PFA	$\{a^n b^n\}$	Union, Intersection	-

Table 2.1: Comparison of language classes

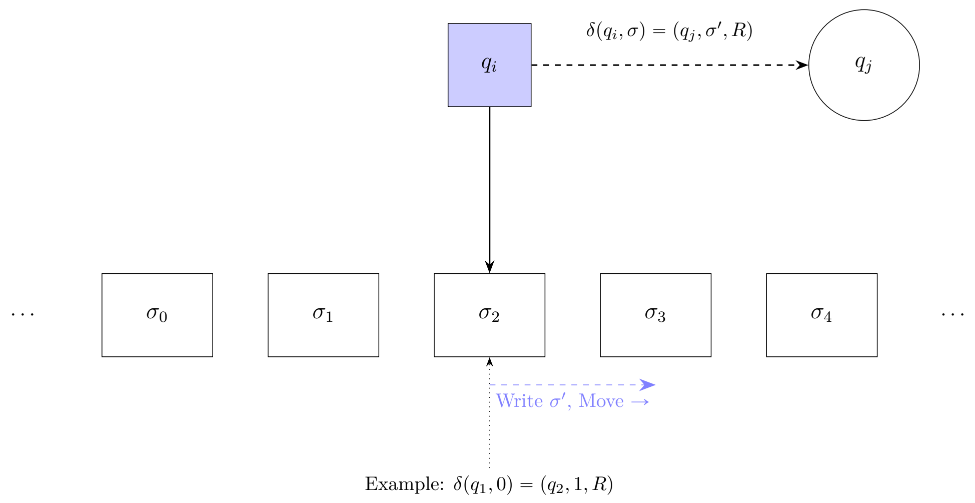
$\delta(q_i, \sigma) = (q_j, \sigma', R)$
q_i
q_j
$\cdots$
σ_0
σ_1
σ_2
σ_3
σ_4
$\cdots$
Write σ', Move $\rightarrow$
Example: $\delta(q_1, 0) = (q_2, 1, R)$

Figure 2.1: Schematic of a Turing machine: tape cells, head, and state transitions

Closure Properties

Closure properties determine how language classes behave under operations:

- REG: Closed under union, intersection, complement, concatenation, and Kleene star [16].
- CFL: Closed under union and Kleene star, but *not* under intersection or complement [13, 16].
- CSL: Closed under union, intersection, and complement [13, 16].
- Stochastic Languages: Closed under union, intersection, and concatenation, but *not* under complementation or Kleene star [31, 30].

Example: REG's closure under intersection ensures that $L_1 \cap L_2$ is regular if $L_1, L_2 \in$ REG. In contrast, stochastic languages are closed under intersection but not under complementation, as shown by their inability to recognize $\overline{L_{\text{eq}}}$ for $L_{\text{eq}} = \{a^n b^n \mid n \geqslant 1\}$ [31].

Operation	REG	CFL	CSL	Stochastic	Type-0
Union	✓	✓	✓	✓	✓
Intersection	✓	×	✓	✓	✓
Complement	✓	×	✓	×	✓
Concatenation	✓	✓	✓	✓	✓
Kleene*	✓	✓	✓	×	✓

Table 2.2: Closure properties comparison

Chomsky Hierarchy

Formal languages are stratified by the Chomsky hierarchy [13, 16]:

1. **Type-3 (Regular)**: Recognized by DFAs [16].

2. **Type-2 (Context-Free)**: Recognized by PDAs [13].

3. **Type-1 (Context-Sensitive)**: Recognized by linear-bounded automata [13].

4. **Type-0 (Recursively Enumerable)**: Recognized by Turing machines [16], which formalize the notion of *algorithmic computability* [37].

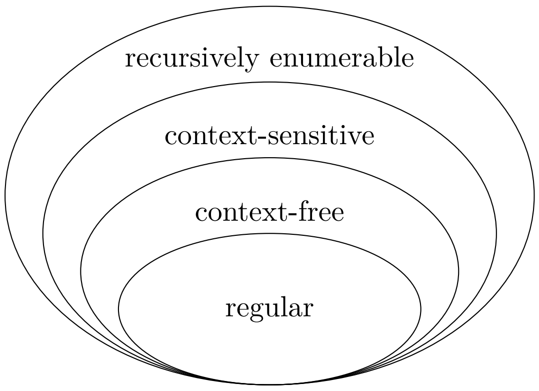

Figure 2.2: Chomsky hierarchy of formal languages

Practical Implications

- **Regular Expressions**: Used in text processing (e.g., `grep`, lexical analyzers) [18, 16].

- **Context-Free Grammars**: Define programming language syntax (e.g., Python's grammar) [13, 16].

- **Closure Properties**: Enable decidability proofs (e.g., emptiness testing for DFAs) [16].

- **Stochastic Models**: Applied in natural language processing and speech recognition for probabilistic pattern matching [31].

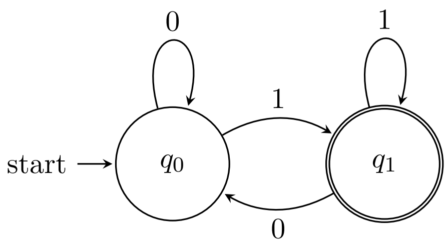

Figure 2.3: Example DFA recognizing even number of 1s

Automata Definition Fundamentals

All automata share core structural components [16, 13].

Definition 1 (Classical Finite Automaton)**.** *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 computational progress [16].*
- *Input Alphabet (Σ): Defined symbols the automaton processes [16].*
- *Transition Function (δ): Governs state changes based on input [13]:*
  - *Deterministic: $\delta : Q \times \Sigma \to Q$ (e.g., DFA) [16].*
  - *Nondeterministic: $\delta : Q \times \Sigma \to 2^Q$ (e.g., NFA) [31].*
- *Initial State ($q_0 \in Q$): The starting configuration [16].*
- *Accept States ($F \subseteq Q$): Terminal states indicating successful computation [16].*

Example: The DFA in Figure 2.4 has:

- $Q = \{q_0, q_1\}$
- $\Sigma = \{0, 1\}$
- $\delta(q_0, 1) = q_1$, $\delta(q_1, 0) = q_0$ *(partial specification)*
- $F = \{q_1\}$ (accepts even number of 1s)

Graphical notation:

- States: Circles with q_i labels
- Initial state: Arrow pointing to the state (q_0)
- Accept states: Double circles (q_1 in 2.4)
- Transitions: Directed edges labeled with input symbols

Automaton	**State Memory**	**Transition Type**	**Acceptance Condition**
DFA	None	Deterministic	Final state membership [16]
NFA	None	Nondeterministic	Existence of accepting path [16]
PDA	Stack	Deterministic/Nondet.	Final state + empty stack [13]
Turing Machine	Tape	Deterministic	Halting in accept state [37]

Table 2.3: Automata representation variations

2.1.2 Deterministic Finite Automata (DFA)

As a specialization of the general finite automaton framework 1 established in Section 2.1.1, a Deterministic Finite Automaton (DFA)[16] is formally defined as a quintuple $M = (Q, \Sigma, \delta, q_0, F)$ where:

- Q: Finite set of states

- Σ: Finite input alphabet
- $\delta : Q \times \Sigma \to Q$: Total transition function
- $q_0 \in Q$: Unique initial state
- $F \subseteq Q$: Set of accepting states

Language Recognition and Expressive Power

DFAs precisely characterize the class of regular languages (REG) within the Chomsky hierarchy [16]. Key properties include:

1. **Closure Properties**: Closed under union, intersection, complement, concatenation, and Kleene star [23].
2. **Limitations**: Cannot recognize non-regular languages like $\{a^n b^n | n \geqslant 0\}$ (pumping lemma consequence) [36].
3. **Equivalence**: All regular expressions have corresponding DFAs and vice versa (Kleene's theorem) [19].

The Myhill-Nerode theorem provides a canonical minimal DFA for any regular language, establishing state minimality criteria [25].

Graphical Representation

Consider the DFA in Figure 2.4 recognizing strings with an even number of `0`s and `1`s. The automaton transitions between states based on input symbols, accepting strings that satisfy the parity constraints.

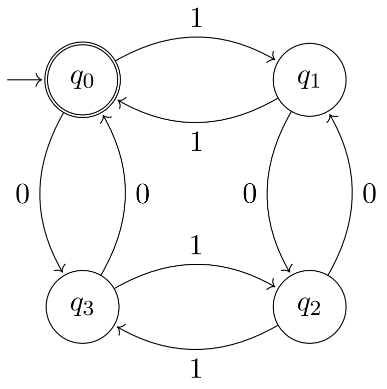

Figure 2.4: DFA recognizing even numbers of `0`s and `1`s.

$L = \{w \in \{0,1\}^* \mid \text{the number of 0's in } w \text{ is even and the number of 1's in } w \text{ is even}\}$ is the language recognized by this DFA.

The DFA is deterministic because for each state and input symbol (either `0` or `1`), there is exactly one transition defined. This ensures that from any given state, the next state is uniquely determined by the current input symbol, leaving no ambiguity in the automaton's behavior.

The structure of the DFA ensures that it keeps track of the parity (even or odd) of the counts of `0`s and `1`s as it processes each input symbol, transitioning between states accordingly to accept only those strings that meet the specified criteria.

2.1.3 Nondeterministic Finite Automata (NFA)

As a generalization of the deterministic framework established in Section 2.1.2, a Nondeterministic Finite Automaton (NFA)[16] allows multiple possible transitions for a given state-symbol pair. Formally, an NFA is defined as a quintuple $M = (Q, \Sigma, \delta, q_0, F)$ where:

- Q: Finite set of states
- Σ: Input alphabet
- $\delta : Q \times (\Sigma \cup \{\epsilon\}) \to 2^Q$: Nondeterministic transition function
- $q_0 \in Q$: Initial state
- $F \subseteq Q$: Accepting states

Language Recognition and Expressive Power

NFAs recognize the same class of regular languages (REG) as DFAs but offer greater representational flexibility through:

1. **Epsilon transitions**: Allowing state changes without input consumption [16]
2. **Multiple transitions**: Permitting multiple possible paths for a single input symbol [14]
3. **Subset equivalence**: NFAs can be converted to equivalent DFAs via the powerset construction, though potentially with $2^{|Q|}$ states [16]

Graphical Representation

Consider the NFA in Figure 2.5 recognizing $L = \Sigma^* ab$. The automaton demonstrates nondeterministic branching and epsilon transitions:

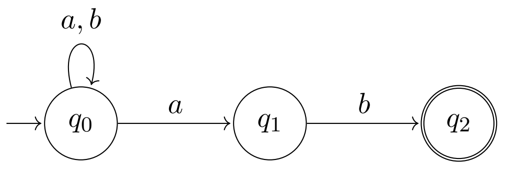

Figure 2.5: NFA recognizing $L = \Sigma^* ab$

$L = \{w \in \{a, b\}^* \mid w \text{ contains the substring } ab\}$ is recognized by this NFA. Key features include:

- Nondeterministic branching: q_0 has two transitions on a
- Implicit epsilon transitions via loopback on q_0
- Acceptance via any path reaching q_2

Equivalent DFA Construction

Using the subset construction method [16], we derive the equivalent DFA shown in Figure 2.6. The conversion process involves:

1. **State creation**: Each DFA state represents a subset of NFA states
2. **Transition calculation**: For each subset $S \subseteq Q$ and input σ, $\delta_{DFA}(S, \sigma) = \bigcup_{q \in S} \delta_{NFA}(q, \sigma)$
3. **Acceptance condition**: A DFA state is accepting if it contains any NFA accepting state [14]

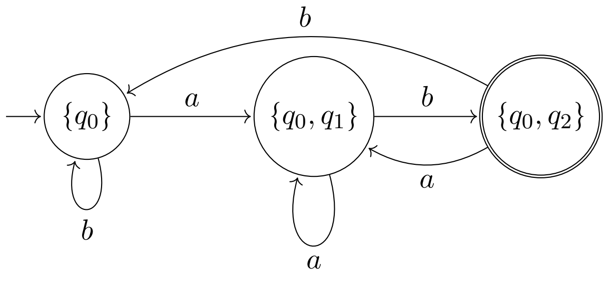
$$\begin{tikzcd}
\arrow[r] & \{q_0\} \arrow[r, "a"] \arrow[loop below, "b"] & \{q_0, q_1\} \arrow[r, "b"] \arrow[loop below, "a"] & \{q_0, q_2\} \arrow[ll, bend right, "b"'] \arrow[l, bend left, "a"]
\end{tikzcd}$$

Figure 2.6: Equivalent DFA for NFA in Figure 2.5 [16]

Key observations:

- The DFA contains only 3 states instead of the theoretical maximum $2^3 = 8$ [16]
- State $\{q_0, q_2\}$ is accepting as it contains q_2
- Transitions preserve the language $L = \{w \in \{a, b\}^* \mid w \text{ contains } ab\}$

Key Features

1. **Expressive equivalence**: NFAs and DFAs have equal computational power [16]
2. **Space-time tradeoff**: NFAs can express certain languages (e.g., $\Sigma^* ab \Sigma^*$) more succinctly than DFAs [10]
3. **Decision algorithms**: Emptiness testing and membership verification have polynomial complexity [31]

Closure Properties and Complementation

All regular language closure properties hold for NFAs *indirectly* through their equivalence to DFAs [16]. However, when directly manipulating NFAs, some operations like complementation require special handling due to nondeterminism:

1. **Standard Operations**: NFAs naturally support closure under *union*, *concatenation*, and *Kleene star* through ϵ-transitions and state duplication [14]. These operations can be implemented without determinization.
2. **Complementation Challenge**: Unlike DFAs, direct state-swapping in NFAs does *not* yield valid complementation. This limitation arises because:

- Nondeterministic paths may allow acceptance through alternate routes even after state inversion [31]
- Epsilon transitions can create implicit acceptance paths unaffected by state swaps [16]

3. **Proper Complementation Method**: To compute $\overline{L(N)}$ for an NFA N:

   (a) Convert N to equivalent DFA D via subset construction [16]

   (b) Swap accepting/non-accepting states in D to get $\overline{D}$ [14]

   This two-step process ensures correctness but may incur exponential state blowup [10]

4. **Equivalence Preservation**: The subset construction guarantees language equivalence between NFAs and DFAs [16], forming the basis for all NFA complementation proofs

5. **Intersection Handling**: While DFAs allow direct product-construction for intersection [16], NFAs require:

   (a) Conversion to DFAs for both languages

   (b) Standard product construction on DFAs

The nondeterministic paradigm serves as a conceptual bridge to quantum automata models, particularly in handling superposition-like state transitions [3].

2.1.4 Probabilistic Finite Automata (PFA)

As a generalization of the classical finite automaton framework 1 established in Section 2.1.1, a Probabilistic Finite Automaton (PFA)[31] is formally defined as a quintuple $M = (Q, \Sigma, \delta, \pi, F)$ where:

- Q: Finite set of states
- Σ: Finite input alphabet
- $\delta : Q \times \Sigma \times Q \to [0, 1]$: Probabilistic transition function satisfying $\sum_{q' \in Q} \delta(q, \sigma, q') = 1$ for all $q \in Q$, $\sigma \in \Sigma$
- $\pi \in \mathbb{R}^{|Q|}$: Initial state distribution vector with $\sum_{q \in Q} \pi_q = 1$
- $F \subseteq Q$: Set of accepting states

Language Recognition and Expressive Power

PFAs recognize stochastic languages through probabilistic acceptance criteria. The language recognized by a PFA M with cut-point $\lambda \in [0, 1)$ is:

$$L(M, \lambda) = \{w \in \Sigma^* \mid \Pr[M \text{ accepts } w] > \lambda\}$$

Key variants and their properties include:

1. **Isolated Cut-Point (λ with margin $\epsilon > 0$):**

$$\Pr[M \text{ accepts } w] \begin{cases} \geqslant \lambda + \epsilon & \text{if } w \in L \\ \leqslant \lambda - \epsilon & \text{if } w \notin L \end{cases}$$

   Recognizes exactly the regular languages [31]

2. **Non-Isolated Cut-Point** ($\lambda = 0$):

$$L(M, 0) = \{w \mid \Pr[M \text{ accepts } w] > 0\}$$

   Recognizes languages beyond regular, including context-sensitive [30]

3. **Strict Cut-Point** ($\lambda = 1$):

$$L(M, 1) = \{w \mid \Pr[M \text{ accepts } w] = 1\}$$

   Equivalent to deterministic finite automata [32]

Closure Properties

PFAs exhibit nuanced closure characteristics compared to classical models:

- Closed under union, intersection, and reversal [30]
- **Not closed** under complementation unless using isolated cut-points [9]
- Kleene star closure requires additional constraints on transition probabilities [17]

Graphical Representation

Consider a PFA recognizing $L_{\text{maj}} = \{w \in \{a, b\}^* \mid |w|_a > |w|_b\}$ with probability $\geqslant 2/3$:

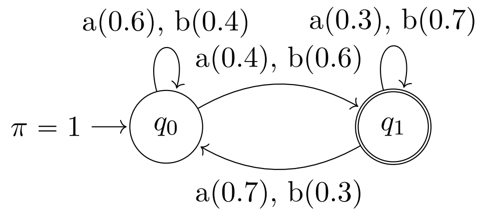

Figure 2.7: PFA for majority language with probabilistic transitions

This PFA maintains a probability distribution across states, with transitions weighted by input symbol probabilities. Acceptance occurs when the cumulative probability in F exceeds the cut-point after processing the entire input.

Key Theoretical Results

1. **Rabin's Theorem**: PFAs with isolated cut-points recognize exactly REG [31]
2. **Pumping Lemma**: Probabilistic version requires probability bounds on loop iterations [30]
3. **Equivalence Problem**: Undecidable for PFAs with non-isolated cut-points [38]

2.1.5 Two-Way Finite Automata Variants

This section examines bidirectional automata models that extend one-way finite automata through tape head reversibility. These classical models provide foundational insights for complexity analysis in quantum computing contexts [16]. We analyze their computational mechanics, state complexity advantages, and language recognition capabilities while emphasizing structural parallels with quantum counterparts.

Two-Way Deterministic Finite Automata (2DFA)

As an extension of the deterministic framework established in Section 2.1.2, a Two-Way Deterministic Finite Automaton (2DFA) [20] is formally defined as a septuple $M = (Q, \Sigma, \delta, q_0, F, \lhd, \rhd)$ where:

- $\lhd, \rhd$: Left/right triangular endmarkers delimiting input
- $\delta : Q \times (\Sigma \cup \{\lhd, \rhd\}) \to Q \times \{L, R, S\}$
- S: "Stay" operation maintaining head position

Operational Mechanics The bidirectional capability enables:

1. **Cross-verification**: Rechecking input symbols for patterns like palindromes
2. **Multi-pass validation**: Repeated scans with $O(1)$ space complexity
3. **Deterministic transitions**: $\delta(q, a) = (q', d)$ with $d \in \{L, R, S\}$
4. **Boundary handling**: Special transitions at $\lhd$ and $\rhd$ markers

State Complexity Advantages 2DFAs demonstrate significant savings over 1DFAs:

- Recognize $L_{eq} = \{a^n b^n | n \geqslant 0\}$ with $O(n)$ states vs. 1DFA's exponential requirements [39]
- Implement context-free-like validations through bidirectional sweeps [16]
- Achieve $O(\sqrt{n})$ state efficiency for languages with periodic structures [20]

Example: Even 01 Substrings Figure 2.8 shows a 2DFA recognizing $L = \{w \in \{0, 1\}^* \mid \text{even number of 01 substrings}\}$. The states q_0-q_3 track substring counts using bidirectional verification:

- q_0: Initial state scanning right
- q_1: Reverse scan after detecting '0'
- q_2: Validate '1' completion
- q_3: Boundary error handling

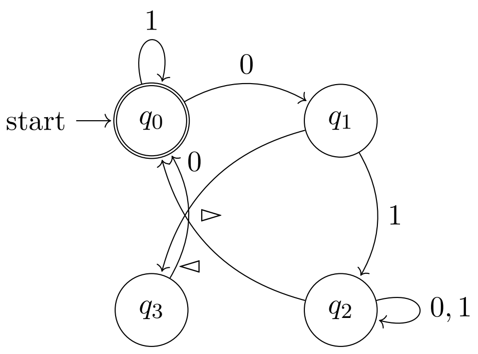

Figure 2.8: 2DFA with triangular endmarkers recognizing even 01 substrings

Two-Way Nondeterministic Finite Automata (2NFA)

Generalizing the NFA framework from Section 2.1.3, a Two-Way Nondeterministic Finite Automaton (2NFA) [16] has transition function:

$$\delta : Q \times (\Sigma \cup \{\lhd, \rhd\}) \to 2^{Q \times \{L,R,S\}}$$

Key Advantages

1. **Exponential state savings**: Recognizes $\{a^n b^{2n} | n \geqslant 0\}$ with $O(1)$ states [39]
2. **Guess-and-verify paradigm**:
   - Nondeterministically "guess" split points (e.g., a^k in $a^k b^{2k}$)
   - Verify guesses through bidirectional traversal
3. **CFL recognition**: Simulates context-free validations in $O(n)$ time [17]

Example: Palindrome Recognition Figure 2.9 demonstrates a 2NFA for $L_{pal} = \{ww^R | w \in \{0,1\}^*\}$:

- Nondeterministically guess midpoint at $\rhd$
- Match symbols bidirectionally from guessed position
- Accept if all symbol pairs match

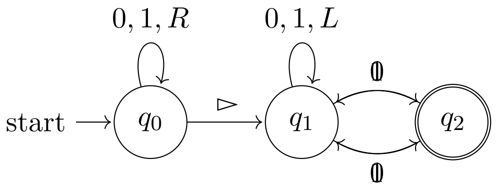

Figure 2.9: 2NFA for palindrome recognition with triangular endmarkers

Two-Way Probabilistic Finite Automata (2PFA)

Extending the PFA framework from Section 2.1.4, a Two-Way Probabilistic Finite Automaton (2PFA) [**freivalds1981probabilistic**] is defined with:

$$\delta(q, a, q', d) = \begin{cases} p \in [0, 1] & \text{probability of transition} \\ 0 & \text{otherwise} \end{cases}$$

satisfying $\sum_{q',d} \delta(q, a, q', d) = 1$.

Computational Phases

1. **Initialization**: Start at $\lhd$ with distribution $\delta(q_0)$
2. **Evolution**: Apply transition matrix to current state distribution
3. **Measurement**: Check for acceptance after each step with possible halting
4. **Error reduction**: Use amplitude amplification for bounded error [26]

Recognition Capabilities

- **Non-Regular Languages**: Recognizes $\{a^n b^n\}$ with bounded error $\epsilon < 1/2$ in $O(n^2)$ time [**freivalds1981probabilistic**]
- **Space-Time Tradeoff**: $O(\log n)$ space requires $O(n^3)$ time [28]
- **Language examples**:
  - $L_{maj} = \{w \in \{a, b\}^* || w|_a > |w|_b\}$
  - $L_{eq} = \{a^n b^n | n \geqslant 0\}$

Example: Majority Language Figure 2.10 shows a 2PFA for L_{maj}:

- Uses probabilistic counters with bidirectional sweeps
- Maintains $O(\log n)$ states through quantum-like interference patterns
- Achieves 2/3 correctness probability through repeated passes

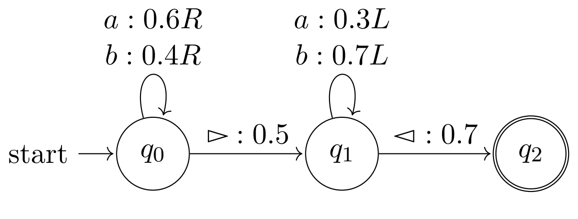

Figure 2.10: 2PFA for majority language with probabilistic transitions

Cut-Point Variations

Isolated Cut-Point $\lambda \pm \epsilon$: Recognizes exactly regular languages [31]

Non-Isolated Cut-Point $\lambda = 0$: Recognizes languages beyond regular [30]

Bounded Error $\epsilon < 1/2$: Enables quantum-inspired error tolerance [26]

Comparative Analysis

Table 2.4 summarizes capabilities:

Model	Language Class	Time Complexity	Space Complexity	State Complexity	Key Re
2DFA	REG	$O(n^2)$	$O(1)$	$O(\sqrt{n})$	[1
2NFA	REG	$O(n)$	$O(1)$	$O(1)$ (CFLs)	[3
2PFA	REG $\subset L \subseteq$ P	$O(n^3)$	$O(\log n)$	$O(n)$	[**freivalds1981**

Table 2.4: Classical two-way automata capabilities with triangular endmarkers

These classical models establish crucial baselines for quantum counterparts like 2QCFA and 2QFA [2], particularly in demonstrating how bidirectional access enhances computational power while maintaining space constraints. The triangular endmarkers ($\lhd, \rhd$) enable precise boundary handling critical for quantum tape operations [20].

2.2 Quantum Mechanics Foundations

This section establishes the quantum mechanical principles underpinning quantum automata theory, emphasizing mathematical formalism and conceptual distinctions from classical systems. The discussion focuses on foundational postulates and their computational implications.

2.2.1 Qubits and Quantum States

A *qubit* is the fundamental unit of quantum information, represented as a normalized vector in a two-dimensional complex Hilbert space $\mathcal{H} = \mathbb{C}^2$. The standard basis states are:

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

A general qubit state is a superposition:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \quad |\alpha|^2 + |\beta|^2 = 1,$$

where $\alpha, \beta \in \mathbb{C}$ are probability amplitudes [26]. This state can be visualized on the **Bloch sphere** (Figure 2.11):

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

with $\theta \in [0, \pi]$ and $\phi \in [0, 2\pi)$.

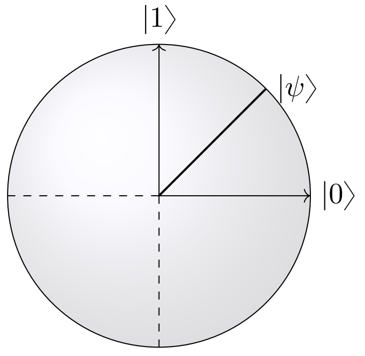

Figure 2.11: Bloch sphere representation of a qubit.

For multi-qubit systems, the state space grows exponentially. A two-qubit system is described by:

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

Table 2.5: Comparison of classical and quantum bits.

Property	**Classical Bit**	**Qubit**
State space	$\{0, 1\}$	$\mathbb{C}^2$ (superposition)
Measurement	Deterministic	Probabilistic (Born rule)
Entanglement	N/A	Non-classical correlations

2.2.2 Superposition and Entanglement

Superposition enables parallel computation. The Hadamard gate (H) creates uniform superpositions:

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

Entanglement arises in multi-qubit states that cannot be factored. The Bell states are maximally entangled:

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

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

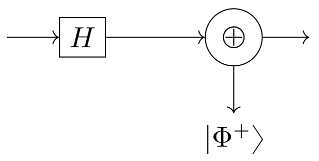

Figure 2.12: Quantum circuit generating the Bell state $|\Phi^+\rangle$.

Entanglement is critical for quantum teleportation [5] and enables exponential speedups in algorithms like Shor's [34].

2.2.3 Quantum Gates and Circuits

Quantum gates are unitary operators ($U^\dagger U = I$). Key single-qubit gates include:

- **Pauli-X**: Bit flip, $X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$.
- **Hadamard**: Creates superposition, $H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$.
- **Phase shift**: $R_\phi = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix}$.

The **CNOT gate** entangles qubits:

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

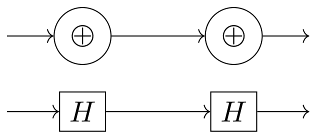

Figure 2.13: Quantum circuit for the Deutsch-Jozsa algorithm.

Quantum circuits decompose algorithms into gate sequences. The **Quantum Fourier Transform** (QFT) is pivotal in Shor's algorithm [34].

2.2.4 Measurement and Probabilistic Outcomes

Measurement collapses a state to a basis vector. For $|\psi\rangle = \sum_i \alpha_i |i\rangle$, the probability of outcome i is $|\alpha_i|^2$ (Born rule). For example, measuring $|\Phi^+\rangle$ in the computational basis yields $|00\rangle$ or $|11\rangle$ with 50% probability each.

Table 2.6: Measurement outcomes for $|\Phi^+\rangle$.

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

Mixed states are described by density matrices $\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|$. This formalism is essential for open quantum systems [11].

2.2.5 Decoherence and Open Systems

Decoherence arises from environment interactions, modeled by the Lindblad 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 L_k are Lindblad operators [11]. Common noise models include: - **Amplitude damping**: Energy loss to the environment. - **Phase damping**: Loss of phase coherence.

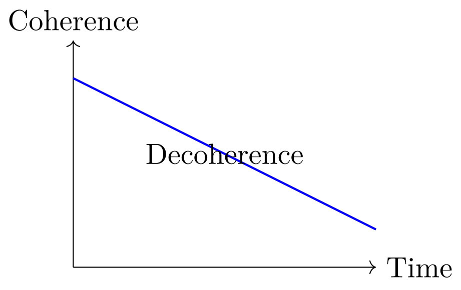

Figure 2.14: Decoherence reduces quantum state fidelity over time.

Quantum error correction codes, like the Shor code [35], mitigate decoherence by encoding logical qubits into multiple physical qubits.

3. Literature Review

3.1 One-way QFAs

3.1.1 Standard Models

MO-1QFA (Measure-Once)

Definition: An MO-1QFA is defined as $M = (Q, \Sigma, \delta, q_0, F)$, where: - Q: Finite set of quantum basis states. - Σ: Input alphabet with end-marker $\#$. - δ: Transition function inducing unitary operators U_σ for $\sigma \in \Sigma$. - $q_0 \in Q$: Initial state. - $F \subseteq Q$: Accepting states [22].

Operation: Processes input sequentially with a single measurement at the end. The state evolves as $|\psi_i\rangle = U_{\sigma_i}|\psi_{i-1}\rangle$. Acceptance is determined by projecting onto F [22].

Key Features: - Recognizes a strict subset of regular languages (e.g., periodic languages) [6]. - Requires $O(n)$ states for languages like $L_{\text{mod}} = \{w \mid |w| \equiv 0 \mod p\}$.

Limitations: - Cannot recognize non-regular languages like $L_{\text{eq}} = \{a^n b^n\}$.

MM-1QFA (Measure-Many)

Definition: An MM-1QFA is $M = (Q, \Sigma, \delta, q_0, Q_{\text{acc}}, Q_{\text{rej}}, Q_{\text{non}})$, where: - $Q_{\text{acc}}, Q_{\text{rej}}, Q_{\text{non}}$: Partitioned states for halting and continuation [20].

Operation: Measures after each symbol, halting if Q_{acc} or Q_{rej} is observed [20].

Key Features: - Recognizes non-regular languages (e.g., L_{eq}) with bounded error [20]. - Exponential state advantage over DFAs for certain languages [3].

Limitations: - Strictly less powerful than two-way QFAs [3].

LQFA (Latvian)

Definition: Combines unitary operations and projective measurements. Defined as $M = (Q, \Sigma, \delta, q_0, F)$, where transitions include measurements [2].

Operation: Applies unitary transformations followed by projective measurements at each step [2].

Key Features: - Recognizes a proper subset of MM-1QFA languages [2]. - Fails to recognize regular languages like $a\Sigma^*$.

Limitations: - Weaker closure properties compared to MM-1QFA [2].

3.1.2 Hybrid Models

1QFAC (Classical States)

Definition: $M = (S, Q, \Sigma, \delta, \mu, s_0, q_0, F)$, combining classical control S and quantum states Q [41].

Operation: Classical state s_i selects quantum operator $\mu(s_i, \sigma)$. Measurement occurs only at the end [41].

Key Features: - Recognizes all regular languages and some non-regular languages (e.g., L_{eq}) [41]. - Exponentially more succinct than DFAs for certain languages [10].

Limitations: - Requires careful error correction due to quantum-classical interaction [41].

CL-1QFA (Control Languages)

Definition: Uses control languages to guide measurements. Defined as $M = (Q, \Sigma, \delta, q_0, \mathcal{L})$, where $\mathcal{L}$ specifies allowed measurement outcomes [8].

Operation: Applies unitary operations and projects onto subspaces dictated by $\mathcal{L}$ [8].

Key Features: - Closed under Boolean operations [8]. - Recognizes regular languages with bounded error [8].

Limitations: - Complex control logic increases implementation overhead [8].

3.1.3 Enhanced Models

EQFA (Enhanced)

Definition: Uses ancilla qubits and arbitrary measurements. Defined as $M = (\Sigma, Q, \{U_\sigma\}, Q_{\text{acc}}, Q_{\text{rej}}, Q_{\text{non}}, q_0$ [29].

Operation: Employs mixed states and non-unitary transitions for enhanced expressiveness [29].

Key Features: - Simulates all classical finite automata [29]. - Recognizes non-regular languages with unbounded error [24].

Limitations: - Irreversible operations complicate error analysis [24].

OT-QFA (Open-Time Evolution)

Definition: Incorporates environmental noise via Lindblad dynamics. Defined as $M = (\Sigma, Q, \mathcal{L}, q_0, F)$, where $\mathcal{L}$ models decoherence [15].

Operation: State evolution governed by the Lindblad equation, with measurement at the end [15].

Key Features: - Generalizes MO-1QFA and MM-1QFA [15]. - Models realistic noisy systems [11].

Limitations: - Undecidable properties due to open-system dynamics [15].

A-QFA (Ancilla-Based)

Definition: Extends MO-1QFA with ancilla qubits. Defined as $M = (Q, \Sigma, \delta, q_0, F)$ with an expanded Hilbert space [29].

Operation: Uses ancillae to simulate classical nondeterminism via quantum interference [29].

Key Features: - Recognizes all regular languages with certainty [29]. - Handles non-regular languages with one-sided error [29].

Limitations: - Ancilla management increases resource overhead [29].

3.1.4 Advanced Variants

1.5QFA (1.5-Way)

Definition: Allows limited head movement. Defined as $M = (Q, \Sigma, \delta, q_0, F)$, where δ restricts leftward motion [20].

Operation: Head moves right or remains stationary but cannot backtrack fully [20].

Key Features: - Recognizes non-regular languages with bounded error [20]. - Strictly more powerful than MO-1QFA [20].

Limitations: - Less powerful than 2QFA [20].

ML-QFA (Multi-Letter)

Definition: Reads k-symbol blocks. Defined as $M = (Q, \Sigma, \delta, q_0, F)$, where δ depends on k-length substrings [4].

Operation: Processes input in chunks, applying unitary operators for each block [4].

Key Features: - Simulates multi-head classical automata [4]. - Recognizes context-sensitive languages with bounded error [4].

Limitations: - State complexity grows exponentially with k [4].

3.2 Two-way QFAs

3.2.1 Standard Models

2QFA (Two-Way)

Definition: A 2QFA is defined as $M = (Q, \Sigma, \delta, q_0, Q_{\text{acc}}, Q_{\text{rej}})$, where: - Q: Finite set of quantum states partitioned into Q_{acc}, Q_{rej}, and Q_{non}. - Σ: Input alphabet with end-markers # (left) and \$ (right). - δ: Transition function defining unitary operators and head movements $\{\leftarrow, \rightarrow, \downarrow\}$ [20].

Operation: The head moves bidirectionally over the input tape. For input $w = \sigma_1\sigma_2 \ldots \sigma_n$, the state evolves as $|\psi_i\rangle = U_{\sigma_i}|\psi_{i-1}\rangle$, with intermediate measurements allowed [20].

Key Features: - Recognizes non-regular languages (e.g., $L_{\text{eq}} = \{a^n b^n\}$) with bounded error in linear time [20]. - Solves the word problem for finitely generated groups [2].

Limitations: - Requires quantum registers scaling with input length, complicating physical implementation [2].

3.2.2 Hybrid Models

2QCFA (Classical States)

Definition: Combines classical control and quantum states. Defined as $M = (S, Q, \Sigma, \Theta, \delta, s_0, q_0, S_{\text{acc}}, S_{\text{rej}})$, where: - S: Classical states controlling transitions. - Q: Quantum states for superposition/mixed states [2].

Operation: Classical states S select quantum operations Θ, while δ governs head movement. Measurements occur adaptively based on classical control [2].

Key Features: - Recognizes L_{eq} and palindromes $L_{\text{pal}} = \{ww^R\}$ in polynomial time with constant quantum states [2]. - Simulates classical 2PFAs while recognizing non-regular languages [41].

Limitations: - Decidability of equivalence between 2QCFAs remains open [41].

3.2.3 Multihead/Tape Extensions

2TQCFA (Two-Tape)

Definition: Extends 2QCFA with two tapes. 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, Σ_2: Input alphabets for two tapes. - δ: Governs synchronized head movements on both tapes [43].

Operation: Processes inputs on two tapes simultaneously, enabling verification of relationships like $L = \{a^n b^n c^n\}$ [43].

Key Features: - Recognizes languages beyond the capabilities of single-tape 2QFAs [43]. - Verifies non-context-free languages in polynomial time [43].

Limitations: - Increased complexity in synchronization and error correction [43].

kTQCFA (k-Tape)

Definition: Generalizes 2TQCFA to k tapes. Defined as $M = (S, Q, \times_{i=1}^{k} \Sigma_i, \Theta, \delta, s_0, q_0, S_{\text{acc}}, S_{\text{rej}})$ [43].

Operation: Coordinates k independent tapes for parallel processing, useful for multi-variable language recognition [43].

Key Features: - Recognizes $L = \{a^n b^{n^2}\}$ with $O(\log n)$ quantum states [43]. - Subsumes classical multi-tape automata in efficiency [43].

Limitations: - Practical implementation constrained by tape synchronization overhead [43].

3.3 Interactive Quantum Automata

QIP (Quantum Interactive Proofs)

Definition: A *Quantum Interactive Proof* (QIP) system involves a polynomial-time quantum verifier V interacting with an unbounded quantum prover P via a shared quantum channel. The verifier is modeled as a quantum finite automaton (QFA) with limited memory [42]. Formally, QIP(k) denotes systems with k rounds of interaction [27].

Operation: The verifier processes input w through alternating rounds of quantum communication with the prover. For 1QFA/2QFA verifiers, transitions are governed by:

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

where Γ is the communication alphabet [42]. Acceptance is determined by measuring the verifier's final state.

Key Features: - QIP = PSPACE [[1], [2]], demonstrating equivalence to classical interactive proofs. - QFA-based verifiers (e.g., 2QFA) recognize languages beyond regular classes with bounded error [27]. - Two-message QIP systems (QIP(2)) are contained in PSPACE <button class="citation-flag" data-index="1">.

Limitations: - Requires precise control over quantum communication channels [42]. - Verifier's state complexity scales with input length for non-regular languages [27].

QMIP (Quantum Merlin-Arthur)

Definition: *Quantum Merlin-Arthur* (QMIP) extends QIP to multiple quantum provers ($k \geqslant 2$) who cannot communicate. Defined as QMIP(k), it allows entangled provers but restricts collusion [33].

Operation: Merlin (prover) sends a quantum witness state $|\psi\rangle$ to Arthur (verifier). For 2QFA verifiers, the transition function validates $|\psi\rangle$ via:

$$\delta : Q \times \Sigma \times \mathcal{H}_{\text{wit}} \to \mathbb{C}^{Q \times Q},$$

where $\mathcal{H}_{\text{wit}}$ is the witness Hilbert space [40].

Key Features: - QMIP = MIP*, enabling recognition of languages beyond QIP <button class="citation-flag" data-index="10">. - Recognizes the palindrome language $L_{\text{pal}} = \{ww^R\}$ with entangled provers [33]. - 2QFA verifiers with QMIP achieve exponential state savings over classical MIP systems [42].

Limitations: - Entanglement between provers introduces verification complexity [40]. - QMIP(1QFA) $\neq$ QIP(1QFA) in polynomial time [27].

4. Conclusion

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] Alberto Bertoni et al. "On the closure of languages under convex-expressions". In: *Theoretical Computer Science* 123.1 (1994), pp. 1–17.

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

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

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

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

[14] GeeksforGeeks. *Introduction of Finite Automata.* `https://www.geeksforgeeks.org/introduction-of-finite-automata/`. 2024.

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

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

[17] Juraj Hromkovič et al. "Probabilistic and nondeterministic unary automata". In: *Mathematical Foundations of Computer Science* (2000), pp. 445–454.

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

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

[20] 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.

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

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

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

[24] 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.

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

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

[27] 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.

[28] Christos H Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994.

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

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

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

[32] Arto Salomaa. "Probabilistic and weighted grammars". In: *Information and Control* 15.1 (1969), pp. 52–63.

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

[34] 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.

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

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

[37] 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.

[38] Christos Tzelepis et al. “Undecidable problems for probabilistic automata of fixed dimension”. In: *Theory of Computing Systems* 65.3 (2021), pp. 573–593.

[39] 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.

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

[41] 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.

[42] 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.

[43] 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