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 $a_1 a_2 \dots a_n$, 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*: $L = \{a^n b^n c^n \mid n \geq 1\}$ [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*: $L_{\text{eq}} = \{a^n b^n \mid n \geq 1\}$ is stochastic but not regular. A PFA can accept this language with probability $\geq \frac{2}{3}$ for valid strings and $\leq \frac{1}{3}$ 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 $\lvert xy\rvert \leq p$
Context-Free (CFL)	PDA	Palindromes	Union, Kleene*	$uvxyz$ with $\lvert vxy\rvert \leq p$
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

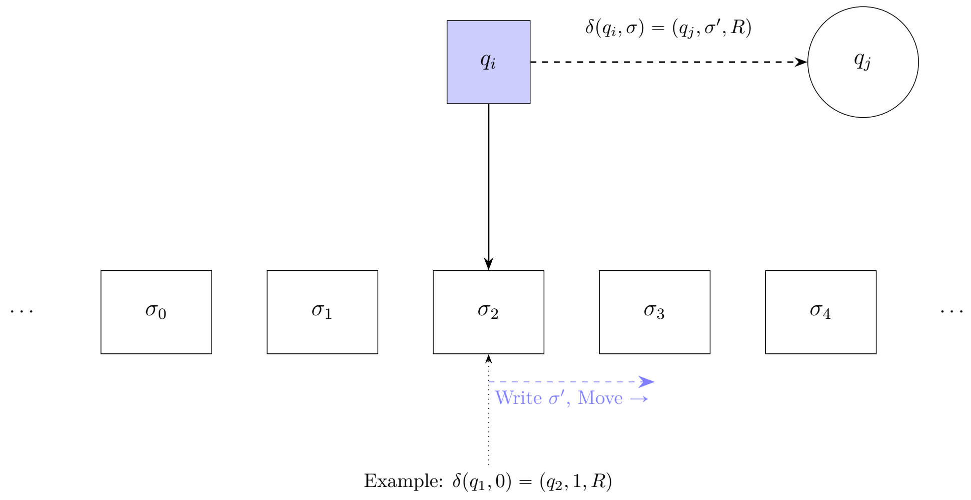

$\delta(q_i, \sigma) = (q_j, \sigma', R)$
q_i
q_j
$\cdots$
σ_0
σ_1
σ_2
σ_3
σ_4
$\cdots$
Write σ', Move $\rightarrow$
Example: $\delta(q_1, 0) = (q_2, 1, R)$

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].

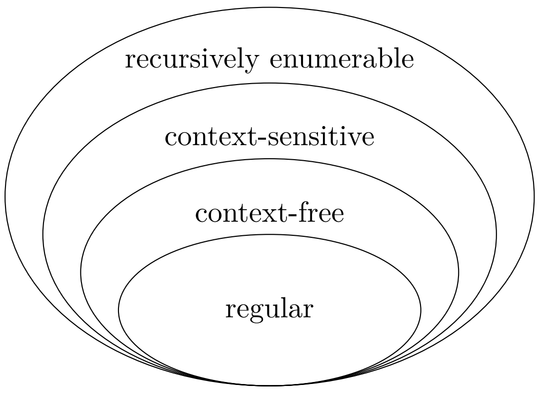


Figure 2.2: Chomsky hierarchy of formal languages

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].

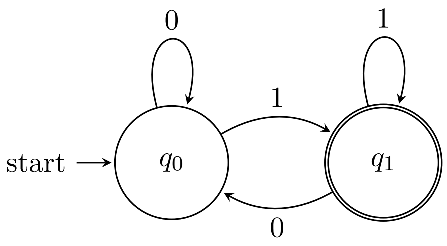


Figure 2.3: Example DFA recognizing even number of 1s

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

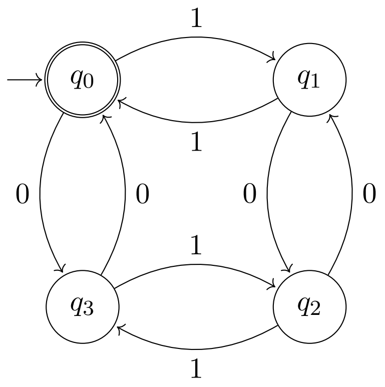
DFAs precisely characterize the class of regular languages (REG) within the Chomsky hierarchy [16]. Key properties include:

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.



$$\begin{tikzcd}
\arrow[r] & q_0 \arrow[r, bend left, "1"] \arrow[d, bend right, "0"'] & q_1 \arrow[l, bend left, "1"] \arrow[d, bend right, "0"'] \\
& q_3 \arrow[u, bend right, "0"'] \arrow[r, bend left, "1"] & q_2 \arrow[l, bend left, "1"] \arrow[u, bend right, "0"']
\end{tikzcd}$$

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:

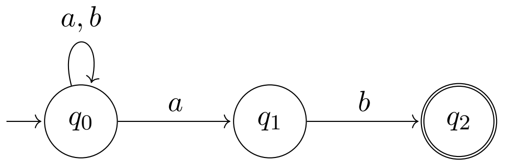


Figure 2.5: NFA recognizing $L = \Sigma^* ab$

$L = \{w \in \{a, b\}^* \mid w$ 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]

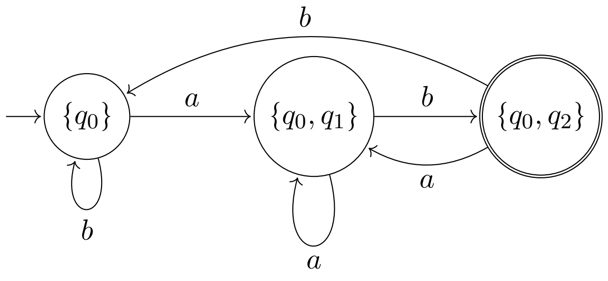


Figure 2.6: Equivalent DFA for NFA in Figure 2.5 [16]

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 $\geqslant 2/3$:

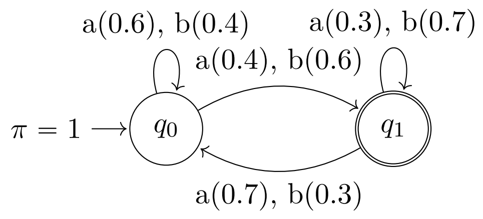


Figure 2.7: PFA for majority language with probabilistic transitions

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

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 1DFA and 2DFA 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 Automata (2DFA)

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 the following conditions:

- For every state $q \in Q$, when reading the left endmarker, 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, 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.

Operational Mechanics The two-way motion allows the automaton to make multiple passes over the input. A typical operation is to scan right until an endmarker is reached, then reverse direction to verify some property of a prefix or suffix, and so on. Although every 2DFA can be simulated by a one-way DFA, the simulation may incur an exponential increase in the number of states.

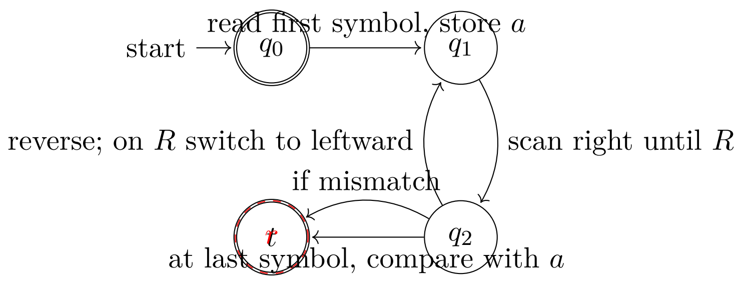
Example: First and Last Symbol Equality As a simple example, consider the language

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

A 2DFA for L works 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 to the 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 possible (simplified) state diagram is as follows:


read first symbol, store a
start
q_0
q_1
reverse; on R switch to leftward
scan right until R
if mismatch
t
q_2
at last symbol, compare with a

(Here the transitions "compare" and "store" are implemented in the finite control; note that the full formal construction would encode the stored symbol as part of the state.)

State Complexity Advantages and Conversion Algorithms For some families of regular languages, two-way machines can be much more succinct than their one-way counterparts. In fact, there exist regular languages for which a 2DFA with $O(n)$ states has an equivalent 1DFA requiring exponentially many states. (See, e.g., [39] for related discussions.) Note, however, that one must be careful when citing specific languages—languages like

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

are not regular and hence are not recognized by any DFA, two-way or one-way.

Several algorithms exist to convert a 2DFA into a 1DFA. Shepherdson's and Kozen's constructions use the notion of *crossing sequences* or *tables* that capture the information carried by the head when it moves across the boundary between portions of the input. Although these constructions are effective, they typically incur an exponential blow-up in the number of states.

Summary In summary, while two-way deterministic finite automata do not recognize more languages than one-way automata, they offer significant advantages in state complexity for certain regular languages and provide a natural framework for multi-pass verification of input properties. Algorithms for transforming a 2DFA into a 1DFA, though constructive, may lead to an exponential increase in the number of states.

Two-Way Nondeterministic Finite Automata (2NFA)

A *Two-Way Nondeterministic Finite Automaton (2NFA)* is defined similarly to the 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 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 is required to obey similar boundary conditions as for 2DFAs: when reading L the head must move right and when reading R it must move left; moreover, once the machine reaches t or r its state does not change.

Key Advantages

1. **Exponential state savings**: For certain families of regular languages there exist 2NFAs that are exponentially more succinct than any equivalent one-way DFA [39].
2. **Guess-and-verify paradigm**: Nondeterminism naturally enables the machine to *guess* important positions (for example, a split point in the input) and then *verify* the guess via bidirectional traversal.
3. **Enhanced succinctness via bidirectional motion**: Even though 2NFAs recognize only regular languages, the ability to re-read the input in both directions may yield machines with very few states compared to their one-way counterparts.

Example: Symmetry Check (Toy Version) To illustrate the nondeterministic bidirectional strategy, consider a (toy) 2NFA for the regular 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 as follows:

1. In the initial phase the machine scans right from the left endmarker L and nondeterministically *guesses* the point at which it will compare the first two symbols with the last two symbols.
2. On reading the right endmarker R it reverses direction.

3. While moving left, the machine nondeterministically *checks* that the symbol it sees at the far left (recorded during the first phase) matches the corresponding symbol at the right.

4. If both comparisons succeed, the automaton enters the accept state t; otherwise, it eventually transitions to the reject state r.

Figure 2.8 (adapted below) schematically illustrates such a mechanism. (Note that this diagram is a simplified illustration of the guess-and-check strategy rather than a complete formal construction.)

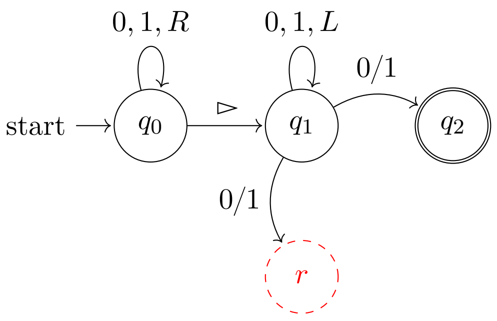


Figure 2.8: Schematic 2NFA illustrating a guess-and-check mechanism (for a toy symmetry language).

Two-Way Probabilistic Finite Automata (2PFA)

A *Two-Way Probabilistic Finite Automaton (2PFA)* extends the classical probabilistic finite automaton (PFA) by allowing the input head to move both left and right. Formally, a 2PFA is an 8-tuple

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

where the components Q, Σ, L, R, s, t, and r are as defined for 2DFAs. The transition function now is

$$\delta : Q \times (\Sigma \cup \{L, R\}) \to \mathbb{R}_{\geqslant 0}^{Q \times \{L,R\}},$$

and for every state q and symbol $a \in \Sigma \cup \{L, R\}$ it satisfies

$$\sum_{(q',d) \in Q \times \{L,R\}} \delta(q, a, q', d) = 1.$$

In other words, $\delta(q, a, q', d)$ is the probability of moving from state q to q' while moving in direction d upon reading a.

Computational Phases A typical 2PFA operates in the following phases:

1. **Initialization**: The automaton starts at the left endmarker L in state s with an initial probability distribution (typically, a point mass at s).

2. **Evolution**: The machine updates its state probabilistically according to the transition function δ while moving its head bidirectionally.

3. **Measurement and Halting**: At any step the machine may enter the accept state t or the reject state r; once one of these states is reached, the computation halts (with the head continuing in a fixed direction).
4. **Error Reduction**: Standard techniques (e.g., repeating the computation or amplitude amplification ideas) are used to achieve bounded-error probabilities.

Recognition Capabilities Some key features of 2PFAs include:

- **Recognition of some nonregular languages**: Unlike deterministic models, 2PFAs with bounded error have been shown to recognize certain nonregular languages such as $\{a^n b^n \mid n \geqslant 0\}$ in $O(n^2)$ time [**freivalds1981probabilistic**].
- **Space-Time Tradeoffs**: With $O(\log n)$ space, certain 2PFAs can achieve polynomial time recognition of specific languages [28].
- **Language Examples**: Common examples include
  - $L_{maj} = \{w \in \{a, b\}^* \mid \#a(w) > \#b(w)\}$, and
  - $L_{eq} = \{a^n b^n \mid n \geqslant 0\}$ (recognized with bounded error).

Example: Majority Language As an illustrative example, consider a 2PFA for the majority language

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

A high-level description is as follows:

1. The automaton makes a probabilistic pass from left to right over the input (bounded by endmarkers), updating a probabilistic counter.
2. It then may reverse direction and perform additional sweeps to refine its decision.
3. After a sufficient number of passes, the machine halts in t with high probability if w belongs to L_{maj} (and in r otherwise).

Figure 2.9 illustrates a schematic 2PFA with sample probabilistic transitions.

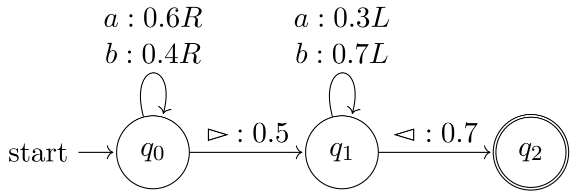


Figure 2.9: Schematic 2PFA for a majority language (with sample probabilistic transitions).

Cut-Point Variations The language recognized by a 2PFA may be defined in various ways:

Isolated Cut-Point: The automaton accepts if the probability is at least $\lambda + \epsilon$ and rejects if at most $\lambda - \epsilon$; in this case the recognized languages are exactly the regular languages [31].

Non-Isolated Cut-Point: With $\lambda = 0$ the model may recognize languages beyond the regular class [30].

Bounded Error: If the error is bounded below 1/2, techniques similar to amplitude amplification can be applied [26].

Comparative Analysis of Two-Way Models

Table 2.4 summarizes key computational parameters of classical two-way automata models. (Here "REG" denotes the class of regular languages.)

Model	**Language Class**	**Time Complexity**	**Space Complexity**	**State Comp**
2DFA	REG	$O(n^2)$	$O(1)$	May be exponentially s
2NFA	REG	$O(n)$	$O(1)$	Can be exponentially more
2PFA	REG $\subset L \subseteq$ P	$O(n^3)$	$O(\log n)$	Varies with err

Table 2.4: Comparison of classical two-way automata models (using explicit endmarkers).

These classical models provide essential baselines for more advanced variants—such as two-way quantum finite automata (2QFA) and two-way quantum-classical automata (2QCFA) [20]—demonstrating how bidirectional access can significantly reduce state complexity while maintaining strict space constraints.

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

A *qubit* is the fundamental unit of quantum information. Mathematically, a qubit is represented as a normalized vector in a two-dimensional complex Hilbert space, i.e.

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

The standard (computational) basis states are defined as

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

A general qubit state 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*. Note that global phase factors (i.e. multiplication of the state by an overall phase factor $e^{i\gamma}$) are physically irrelevant.

A convenient geometrical representation of qubit states is provided by the **Bloch sphere**. Any pure qubit state can be written as

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

with $\theta \in [0, \pi]$ and $\phi \in [0, 2\pi)$. Figure 2.10 illustrates this representation.

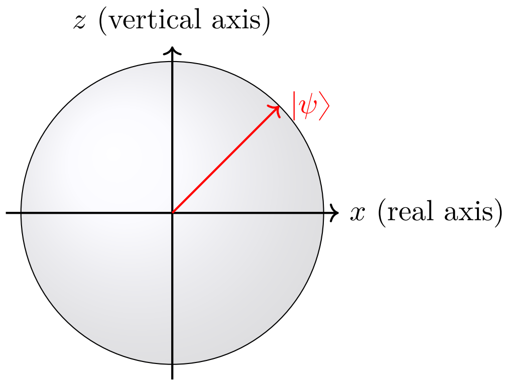


Figure 2.10: Bloch sphere representation of a qubit.

For multi-qubit systems the state space is the tensor product of individual qubit spaces. For example, 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 growth in the state space is at the heart of quantum parallelism.

2.2.2 Superposition and Entanglement

Superposition is the principle that a quantum state can exist as a linear combination of basis states. This underlies the quantum parallelism exploited by quantum algorithms. For example, applying the Hadamard gate H to $|0\rangle$ creates a uniform superposition:

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

Entanglement is a uniquely quantum correlation between subsystems. An entangled state is one that cannot be factored into a product of individual states. The **Bell states** are classic examples of maximally entangled two-qubit states:

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

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

Beyond Bell states, multipartite entangled states such as the Greenberger–Horne–Zeilinger (GHZ) state

$$|\text{GHZ}\rangle = \frac{|000\rangle + |111\rangle}{\sqrt{2}}$$

and the W state

$$|W\rangle = \frac{|001\rangle + |010\rangle + |100\rangle}{\sqrt{3}}$$

play critical roles in quantum communication and error correction.

Entanglement is essential for phenomena like quantum teleportation [5], superdense coding, and it contributes to the potential exponential speedup in algorithms such as Shor's factorization algorithm [34].

2.2.3 Quantum Gates and Circuits

Quantum gates are the building blocks of quantum circuits. They are represented by *unitary operators* U (i.e. $U^\dagger U = I$) that govern the evolution of quantum states. Key single-qubit gates include:

- **Pauli-X (bit-flip):**
$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix},$$
which acts as $X|0\rangle = |1\rangle$ and $X|1\rangle = |0\rangle$.

- **Hadamard (creating superpositions):**
$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

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

Two-qubit gates, such as the **Controlled-NOT (CNOT) gate**, introduce entanglement. The CNOT gate acts on a pair of qubits as

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

where $\oplus$ denotes addition modulo 2.

Quantum circuits decompose complex algorithms into sequences of gate operations. A universal gate set (e.g., $\{H, T, \text{CNOT}\}$) can approximate any unitary operation to arbitrary precision, forming the basis for the quantum circuit model.

Figure 2.11 shows an example quantum circuit used in the Deutsch-Jozsa algorithm, illustrating the interplay of Hadamard gates and entangling operations.

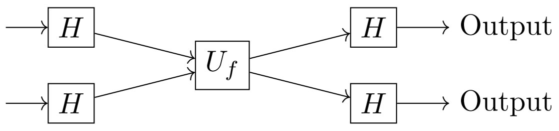


Figure 2.11: Quantum circuit for the Deutsch-Jozsa algorithm.

2.2.4 Measurement and Probabilistic Outcomes

Measurement in quantum mechanics is a fundamentally probabilistic process. When a quantum system in state

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

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

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

This process is typically described as a *projective measurement*, after which the state collapses to the observed eigenstate.

More generally, measurements can be described by **Positive Operator-Valued Measures (POVMs)**, which provide a framework for describing generalized measurements that are not necessarily projective. In a POVM, each measurement outcome is associated with a positive operator E_i satisfying $\sum_i E_i = I$. The probability of outcome i is then given by

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

For example, measuring the Bell state $|\Phi^+\rangle = \frac{|00\rangle+|11\rangle}{\sqrt{2}}$ in the computational basis yields the outcomes $|00\rangle$ or $|11\rangle$ with 50% probability each, as summarized in Table 2.5.

Table 2.5: Measurement outcomes for $|\Phi^+\rangle$.

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

Measurement is an irreversible process and plays a critical role in quantum algorithms as well as in quantum automata theory, where it provides the means to extract classical information from quantum computations.

Mixed states, which describe statistical ensembles of quantum states, are represented by density matrices:

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

with $p_i \geqslant 0$ and $\sum_i p_i = 1$. This formalism is essential when considering open systems subject to decoherence.

2.2.5 Decoherence and Open Systems

In an ideal (closed) quantum system, evolution is unitary. However, in practice, quantum systems interact with their environments, leading to **decoherence**. Decoherence describes the loss of quantum coherence (i.e. the decay of off-diagonal elements in the density matrix) and causes the system to behave more classically.

The dynamics of an open quantum system are often modeled 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 of the system, H is the system Hamiltonian, and L_k are the Lindblad (noise) operators [11]. Typical noise models include:

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

Decoherence is a central challenge in quantum computation and communication. To mitigate its effects, one employs **quantum error correction** codes (e.g., Shor code [35]) and develops decoherence–free subspaces.

2.2.6 Additional Remarks: Unitary Evolution and Quantum Dynamics

In a closed quantum system the evolution is governed by the Schrödinger equation,

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

where H is the Hamiltonian of the system. The formal solution is given by

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

and $U(t)$ is a unitary operator. In quantum circuits, the discrete analog of this continuous evolution is realized by sequences of quantum gates.

The **no-cloning theorem** is another foundational result stating that it is impossible to create an identical copy of an arbitrary unknown quantum state. This principle has deep implications for quantum information processing and quantum cryptography.

2.2.7 Summary

The foundations of quantum mechanics—qubits, superposition, entanglement, unitary evolution, measurement, and decoherence—form the bedrock upon which quantum automata theory is built. Understanding these principles is crucial for grasping how quantum automata differ from their classical counterparts and how quantum effects can be harnessed to achieve computational advantages.

3. Literature Review

This chapter reviews the literature on quantum finite automata Quantum Finite Automaton (QFA), juxtaposing the quantum models with their classical counterparts. We discuss one-way and two-way QFA 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 Quantum Finite Automata (QFAs)

One-way QFA 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 QFA (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

MO-1QFA (Measure-Once QFA)

Definition 2 (MO-1QFA)**.** *An Measure-Once Quantum Finite Automaton (MOQFA) 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 MOQFA 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 [22, 6].

Language Acceptance: Due to the requirement that all transformations be unitary and the measurement occurs only once at the end, MOQFA 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 MOQFA 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.

MM-1QFA (Measure-Many QFA)

Definition 3 (MM-1QFA). *An Measure-Many Quantum Finite Automaton (MMQFA) 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 MMQFA 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

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

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 [20, 3].

Language Acceptance: MMQFA have been shown to accept certain languages with bounded error that are not accepted by MOQFA. For instance:

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

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

  under bounded-error acceptance conditions, although this comes with restrictions on error bounds [20].

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

However, due to the reversibility imposed by unitarity and the intermediate measurement process, MMQFA 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 MOQFA.

- **Space Efficiency:** In certain cases, MMQFA 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 (Latvian QFA)

Definition 4 (LQFA)**.** *A Latvian QFA (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 MMQFA. 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 MMQFA.
- **Expressive Power:** While LQFA can sometimes simulate MMQFA 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:

- **MOQFA** perform a single measurement at the end of computation, resulting in a simple operational model that recognizes exactly the reversible regular languages.
- **MMQFA** 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.

1QFAC (One-Way QFA with Classical States)

Definition 5 (1QFAC)**.** *A one-way QFA with classical states (One-way Quantum Finite Automaton with Classical States (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 [41]. 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 [10].

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.

CL-1QFA (Control Language QFA)

Definition 6 (CL-1QFA)**.** *A Control Language QFA (One-way Quantum Finite Automaton with Classical States (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.

EQFA (Enhanced QFA)

Definition 7 (EQFA). *An enhanced quantum finite automaton (Enhanced Quantum Finite Automaton (EQFA)) is defined as*

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

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.*
- $Q_{acc}, Q_{rej}, Q_{non}$ *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 [29].

Language Acceptance: EQFA can simulate classical finite automata and, under unbounded error, can even recognize certain non-regular languages [24]. 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.

OT-QFA (Open-Time Evolution QFA)

Definition 8 (OT-QFA)**.** *A quantum finite automaton with open time evolution (Open Time Evolution Quantum Finite Automaton (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.*
- *$\delta : \Sigma \to CPTP(\mathcal{H}_{|Q|})$ 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

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

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 [15].

Language Acceptance: OT-QFA 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.

A-QFA (Ancilla-Based QFA)

Definition 9 (A-QFA)**.** *An ancilla-based QFA (Ancilla-Based Quantum Finite Automaton (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

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

allowing the automaton to simulate additional quantum resources or classical nondeterminism [29].

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: Ancilla-based 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.
- **OT-QFA** 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 two-way QFA models by incorporating restricted bidirectional motion or processing of multiple symbols at once. These models aim to bridge the gap between the expressive power of one-way QFA and that of full two-way QFA, often achieving enhanced recognition capabilities with bounded error while retaining some of the simplicity of one-way motion.

1.5QFA (One-and-a-Half Way QFA)

Definition 10 (1.5QFA)**.** *A 1.5-way QFA 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, allows the tape head to move primarily rightward but also permits 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 permitted limited backward (leftward) motion or to remain stationary on a given symbol, thereby providing a restricted form of bidirectional scanning. This additional flexibility allows for an enhanced control over state transitions, which can improve the automaton's ability to recognize certain non-regular patterns under bounded error [20].

Key Features:

- Enhanced computational power compared to strictly one-way QFA.
- Capable of recognizing certain non-regular languages with bounded error.
- More expressive than MO-1QFA while remaining less powerful than full two-way QFA.

Limitations:

- The restricted backward motion, while beneficial, still limits the overall expressive power relative to fully two-way models.
- The complexity of the head motion may lead to increased implementation challenges.

ML-QFA (Multi-Letter QFA)

Definition 11 (ML-QFA)**.** *An ML-QFA 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 individual symbols; that is, 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: In an ML-QFA, the input is partitioned into consecutive blocks of k symbols. For each block, a unitary operator corresponding to the entire k-letter substring is applied, thereby simulating a form of multi-head or parallel processing. This block-wise processing allows the automaton to capture dependencies over longer segments of the input, which is particularly useful for recognizing patterns beyond the capabilities of single-letter transitions [4].

Key Features:

- Simulates classical multi-head automata by processing multiple symbols simultaneously.
- Can recognize some context-sensitive languages with bounded error.

Limitations:

- The state complexity of the automaton tends to grow exponentially with the block size k, which can limit practical implementations.
- Increased computational overhead in managing and designing unitary operators for blocks rather than individual symbols.

Summary of Advanced Variants

The advanced QFA variants discussed exhibit unique trade-offs that aim to enhance the computational power of standard one-way QFA models:

- **1.5QFA** extend the traditional one-way model by allowing restricted backward moves or stationary steps. This limited bidirectional motion improves the recognition of certain non-regular languages with bounded error, yet it remains less powerful than full two-way QFA.
- **ML-QFA** process input in blocks of k symbols, effectively simulating a multi-head behavior. This capability allows ML-QFA to recognize more complex language patterns (including some context-sensitive languages), but at the cost of exponential growth in state complexity with increasing block size.

Overall, advanced QFA variants offer enhanced recognition capabilities and greater expressiveness than standard one-way QFA models, although they do so with increased implementation complexity and resource requirements.

Detailed Hierarchy of One-Way QFA Models

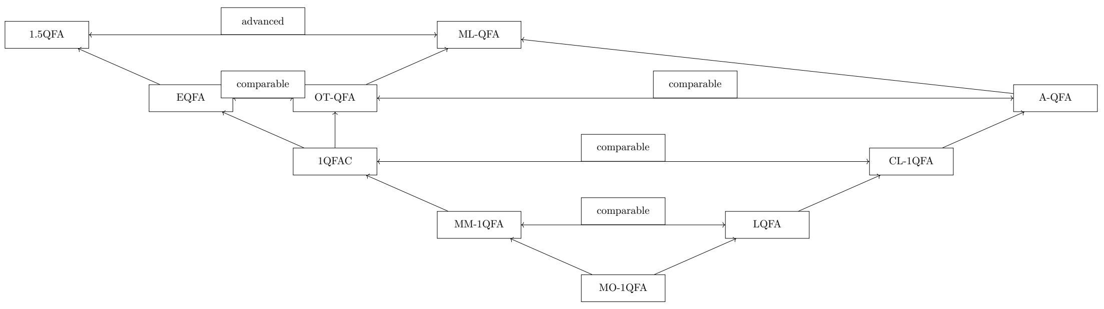


Figure 3.1: Detailed hierarchy of one-way QFA models by expressive power. Vertical arrows indicate a strict increase in expressive power from lower to higher levels, while horizontal arrows denote models with comparable expressive capabilities.

3.2 Two-way Quantum Finite Automata (QFAs)

Two-way QFAs extend one-way models by permitting bidirectional head movement, thereby enhancing their computational power. In two-way QFA models the head may move both left and right (or remain stationary), which allows the automaton to exploit quantum interference over multiple passes of the input. This section discusses standard two-way QFAs, hybrid variants that incorporate classical control, and multihead/tape extensions.

3.2.1 Standard Models

2QFA (Two-Way QFA)

Definition 12 (2QFA)**.** *A two-way quantum finite automaton (Two-way Quantum Finite Automaton (2QFA)) is defined as*

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

where:

- *Q is a finite set of quantum states, partitioned into accepting states Q_{acc}, rejecting states Q_{rej}, and non-halting states $Q_{non} = Q \backslash (Q_{acc} \cup Q_{rej})$.*
- *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 $w = \sigma_1\sigma_2 \dots \sigma_n$, the automaton repeatedly applies the corresponding unitary operators while moving its head in a left, right, or stationary direction as dictated by δ. The use of intermediate measurements (or deferred measurement techniques) allows the 2QFA to decide acceptance based on quantum interference effects. This capability enables 2QFA to recognize certain non-regular languages – for instance, the language

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

has been shown to be recognizable with bounded error in linear time [20, 39].

Key Features:

- Recognizes non-regular languages such as L_{eq} with bounded error.
- Exhibits quantum interference that can yield exponential state savings over classical two-way automata.

Limitations:

- Requires quantum registers whose size may scale with the input.
- Error management and decoherence remain significant challenges.

3.2.2 Hybrid Models

2QCFA (Two-Way QFA with Classical Control)

Definition 13 (2QCFA)**.** *A two-way quantum finite automaton with classical control (Two-way Quantum Classical Finite Automaton (2QCFA)) is defined as*

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

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 (either 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 controls the head movement and decides when to invoke quantum operations as dictated by Θ. Adaptive measurements are performed based on the classical state, allowing the automaton to decide acceptance using a small quantum register. This model is powerful enough to recognize languages such as L_{eq} and palindromes in polynomial time while using a constant number of quantum states [2].

Key Features:

- Can recognize certain non-regular languages in polynomial time.
- Combines the reliability of classical control with the parallelism of quantum computation.

Limitations:

- Synchronization between the classical and quantum components adds complexity.
- The decidability of the equivalence problem for 2QCFA remains an open issue.

3.2.3 Multihead/Tape Extensions

2TQCFA (Two-Tape QCFA)

Definition 14 (2TQCFA)**.** *A two-tape quantum finite automaton with classical control (Two-way Two-Tape Quantum Classical Finite Automaton (2TQCFA)) is defined as*

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

where:

- Σ_1 *and* Σ_2 *are the input alphabets for the two tapes.*
- *The remaining components 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, which allows it to verify language properties involving correlated substrings. For example, languages such as

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

can be recognized efficiently by comparing symbols from the two tapes [43].

Key Features:

- Enhanced recognition power compared to single-tape 2QCFA.
- Can recognize some languages beyond context-free languages in polynomial time.

Limitations:

- Increased synchronization complexity between tape heads.
- Higher error-correction overhead.

kTQCFA (k-Tape QCFA)

Definition 15 (kTQCFA)**.** *A k-tape quantum finite automaton with classical control (k-Tape Quantum Classical Finite Automaton (KTQCFA)) generalizes 2TQCFA to k tapes. It is defined as*

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

where:

- *Each tape has its own alphabet Σ_i.*
- *The transition function coordinates head movements across all k tapes.*

Operation: The automaton processes input from k tapes simultaneously. This multi-tape configuration enables efficient recognition of languages with complex structural dependencies – for example, languages like

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

can be recognized more effectively through parallel comparisons [43].

Key Features:

- Potential for exponential state savings compared to classical multi-tape automata.
- Greater expressive power for languages with intricate dependencies.

Limitations:

- Complexity and overhead increase with the number of tapes k.
- Synchronizing multiple tape heads becomes progressively challenging.

Summary of Two-Way QFA 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 can 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. Such models enhance recognition power further—capable of processing languages with complex structural dependencies—but at the cost of increased synchronization and error-correction complexity.

Detailed Hierarchy of Two-Way QFA Models

3.3 Interactive Quantum Automata

Interactive quantum automata integrate communication between a resource-limited quantum verifier and an (often unbounded) prover.

Quantum Interactive Proof Systems (QIP)

Definition 16 (Quantum Interactive Proof (QIP) System)**.** *A Quantum Interactive Proof (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) [27, 42].*

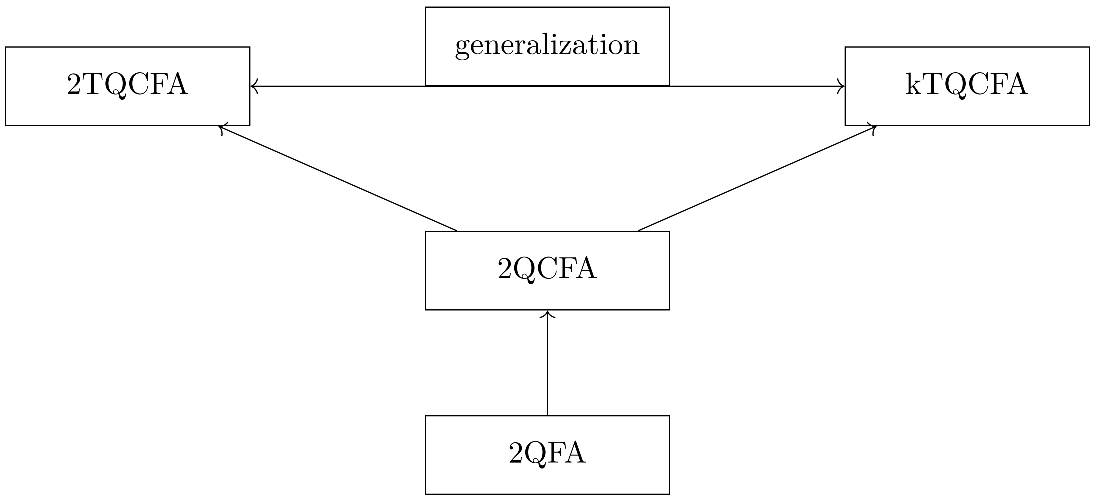


Figure 3.2: Detailed hierarchy of two-way QFA models by expressive power. Vertical arrows indicate an increase in expressive power from standard to hybrid and then to multi-tape extensions, while the horizontal arrow denotes that kTQCFA generalize 2TQCFA.

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. Acceptance is determined by measuring the verifier's final state after the prescribed rounds [42].

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:

- Requires precise control of quantum communication channels.
- The verifier's state complexity may scale with the input for complex languages.

Quantum Merlin-Arthur (QMIP) Systems

Definition 17 (QMIP System)**.** *Quantum Merlin-Arthur (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 [33, 40].*

Operation: The verifier (modeled as a 2QFA-based machine) receives quantum witness states from the provers and processes them using a quantum circuit. The final state is measured to decide acceptance. The provers are not allowed to communicate with each other, and the verification is non-adaptive [33].

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 multiprover interactive proofs.

Limitations:

- Managing entanglement among multiple provers complicates the verification process.
- Practical implementations are highly sensitive to noise and decoherence.

3.4 Summary and Comparative Analysis

In summary, the quantum finite automata literature comprises a rich variety of models, each balancing quantum resources with computational power:

- **One-way QFAs**:
  - *MO-1QFA* employs a single measurement at the end, yielding a model with relatively simple dynamics but limited language recognition power.
  - *MM-1QFA* interweaves unitary evolution with measurements after each symbol, thus recognizing a larger subset of regular languages (and even some non-regular languages) with bounded error.
  - *LQFA* integrate unitary operations with immediate measurements, offering a trade-off between measurement overhead and expressiveness.
- **Hybrid Models**:
  - *1QFAC* merge classical control with quantum processing to recognize all regular languages and sometimes even non-regular languages with significant state savings.
  - *CL-1QFA* use control languages to guide measurement outcomes, ensuring closure under Boolean operations.
- **Enhanced Models**:
  - *EQFA* and *OT-QFA* extend the basic models by allowing non-unitary evolution (via superoperators or CPTP maps) and open system dynamics, thereby simulating classical automata and, in some cases, recognizing non-regular languages.
  - *A-QFA* leverage ancilla qubits to simulate additional quantum resources.
- **Two-way QFAs**:
  - *2QFA* allow bidirectional head movement and can recognize some non-regular languages (e.g., $\{a^n b^n\}$) with bounded error in linear time.
  - *2QCFA* hybridize classical control with two-way quantum evolution, enabling recognition of complex languages with limited quantum resources.

- **Multihead/Tape Extensions**:
  - *2TQCFA* and *kTQCFA* extend two-way QFAs to multiple tapes, further increasing their computational power and efficiency for languages with intricate structure.
- **Interactive Models**:
  - *QIP* systems incorporate interactive proofs with quantum verifiers, linking QFA models with complexity classes like PSPACE.
  - *QMIP* systems extend this framework to multiple provers, opening avenues for recognizing languages beyond the reach of single-prover systems.

A central theme across these models is the trade-off between expressive power and resource efficiency. While many one-way QFA models recognize only proper subsets of the regular languages, hybrid and enhanced models have been developed to bridge the gap with classical finite automata. Two-way models, on the other hand, leverage bidirectional motion to achieve recognition of non-regular languages under bounded error. Interactive models further extend the capabilities by incorporating communication protocols.

This taxonomy not only resolves inconsistencies in earlier literature but also highlights open research problems in equivalence checking, minimization, and error management. As quantum hardware progresses, these theoretical models will be crucial for guiding the design of efficient quantum algorithms and automata implementations.

4. 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 deterministic finite automata (DFAs), nondeterministic finite automata (NFAs), and probabilistic finite automata (PFAs), and then extended these concepts to a variety of quantum finite automata (QFA) models. In doing so, a systematic taxonomy was developed that organizes one-way models (e.g., MO-1QFA, MM-1QFA, 1QFAC) and two-way models (e.g., 2QFA, 2QCFA) 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.

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