Università degli Studi di Camerino

Scuola di Scienze e Tecnologie

Corso di Laurea in Informatica (Classe L-31)

Towards a Unified Taxonomy and Circuit Compilation Framework for Quantum Finite Automata

Candidate
Marta Musso

Student ID 122360

Advisors
Prof. Dr. Michele Loreti
Prof. Dr. Marcello Bonsangue

A.A. 2024/2025

Abstract

As quantum computing matures, concise models are needed to connect its theoretical foundations with practical implementations. Quantum finite automata fulfil this role by extending the well-known framework of classical finite automata into the quantum domain, offering a compact setting in which to study finite-memory quantum behaviour.

This thesis first revisits the essential background on both quantum information and classical automata, establishing a common language for readers from either field. It then delivers a comprehensive literature review that consolidates the many quantum finite automata variants introduced over the last three decades and arranges them in a unified, consistently named taxonomy. Building on that organisation, the work presents a compilation algorithm that translates the widely studied measure-once and measure-many quantum finite automata models into architecture-independent quantum-circuit templates, thereby bridging abstract automaton descriptions with executable gate-level designs.

Viewed more broadly, this study contributes a computer-science perspective on the quantum landscape and offers an accessible entry point for further research in quantum software by promoting circuit-level abstraction and enabling systematic comparison of different designs.

Keywords: Quantum finite automata, automata theory, quantum information, quantum computing, quantum circuits, quantum compilation.

Contents

1. Introduction

2. Background

2.1 Classical Finite Automata and Languages

2.1.1 Formal Languages and Grammars

2.1.2 Classical Finite Automata Definition Fundamentals

2.1.3 Deterministic Finite Automaton

2.1.4 Nondeterministic Finite Automaton

2.1.5 Probabilistic Finite Automaton

2.1.6 Two-Way Variants

2.2 Quantum Mechanics Foundations

2.2.1 Qubits and Quantum States

2.2.2 Superposition and Entanglement

2.2.3 Measurement and Probabilistic Outcomes

2.2.4 Decoherence and Open Systems

2.2.5 Unitary Evolution and Quantum Dynamics

2.3 Quantum Gates and Circuits

3. Quantum Finite Automata

3.1 Detailed Models of Quantum Finite Automata

Among the numerous variants of Quantum Finite Automata (QFAs), the measure-once and measure-many models stand out as two of the most foundational and widely studied frameworks. Respectively known as Measure Once One-Way Quantum Finite Automaton (MO-1QFA) and Measure Many One-Way Quantum Finite Automaton (MM-1QFA), these automata provide a minimalistic yet powerful theoretical playground for investigating the principles of quantum computation with finite memory. Their significance lies not only in their historical development as some of the earliest quantum models for language recognition [36, 25], but also in their role as archetypal examples in the study of quantum-classical computational boundaries.

A MO-1QFA performs unitary operations throughout the input reading process and conducts a single measurement only at the end of the computation. This model, introduced by Moore and Crutchfield [36], can recognize only a restricted class of regular languages, such as the so-called group languages [15]. In contrast, the MM-1QFA, introduced by Kondacs and Watrous [25], allows measurements after each symbol is processed, enabling it to recognize a strictly larger class of regular languages, though still not the full class.

The primary importance of MO-1QFA and MM-1QFA is not just theoretical. These models are particularly suitable for demonstrating techniques for the compilation of quantum automata into quantum circuits, as we will explore in Chapter 4. Due to their simpler architecture—one-way movement, discrete time steps, and finite-dimensional Hilbert spaces—these automata provide an ideal framework for illustrating how abstract automaton transitions can be implemented using quantum gates and projective measurements. In this thesis, they will serve as canonical models to illustrate the compilation strategy, setting a baseline for comparison with more advanced or generalized QFA variants.

3.1.1 Measure Once One-Way Quantum Finite Automaton

Introduction

MO-1QFAs represent one of the simplest models of quantum computation in the realm of automata theory. Introduced by Moore and Crutchfield in 2000 [36], MO-1QFAs evolve solely through unitary transformations corresponding to the input symbols and perform a single measurement at the end of the computation. This model has been further characterized by Brodsky and Pippenger [15], and it is known for its conceptual simplicity as well as for its limitations. Notably, when restricted to bounded error, MO-1QFAs recognize exactly the class of group languages—a proper subset of the regular languages. In contrast, the measure-many variant [25] employs intermediate measurements and exhibits different acceptance capabilities.

Formal Definition

An MO-1QFA is formally defined as a 5-tuple

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

where:

- Q is a finite set of states,
- Σ is a finite input alphabet, typically augmented with a designated end-marker (e.g., \$),
- $\delta : Q \times \Sigma \times Q \to \mathbb{C}$ is the transition function such that, for every symbol $\sigma \in \Sigma$, the matrix
$$U_\sigma, \quad \text{with} \quad (U_\sigma)_{q,q'} = \delta(q, \sigma, q'),$$
is unitary [36],
- $q_0 \in Q$ is the initial state, and
- $F \subseteq Q$ is the set of accepting states.

For an input string $x = x_1 x_2 \cdots x_n$, the computation proceeds by applying the corresponding unitary matrices sequentially:

$$|\Psi_x\rangle = U_{x_n} U_{x_{n-1}} \cdots U_{x_1} |q_0\rangle .$$

After reading the entire input, a measurement is performed using the projection operator

$$P = \sum_{q \in F} |q\rangle\langle q| ,$$

so that the acceptance probability is defined as

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

Alternative characterizations, including formulations using the Heisenberg picture, have been discussed in [44, 41].

Strings Acceptance

A string x is accepted by an MO-1QFA if the acceptance probability $p_M(x)$ exceeds a predetermined cut-point λ. In a bounded error setting, there exists a margin $\epsilon > 0$ such that:

$$\forall x \in L : \quad p_M(x) \geqslant \lambda + \epsilon,$$
$$\forall x \notin L : \quad p_M(x) \leqslant \lambda - \epsilon.$$

Under the unbounded error regime, MO-1QFAs can accept some nonregular languages (for instance, solving the word problem over the free group) [15]. The precise acceptance behavior thus depends on whether a cut-point or a bounded error framework is adopted.

Set of Languages Accepted

When MO-1QFAs are restricted to bounded error acceptance, they recognize exactly the class of *group languages*—languages whose syntactic semigroups form groups [15]. This class forms a strict subset of the regular languages, emphasizing the inherent limitation of the MO-1QFA model. In contrast, by relaxing the error bounds, one may design MO-1QFAs that accept a broader range of languages, albeit often at the cost of increased computational complexity.

Closure Properties

The class of languages accepted by MO-1QFAs under bounded error exhibits robust closure properties. Specifically, this class is closed under:

- Inverse Homomorphisms [15],
- Word Quotients [15],
- Boolean Operations (union, intersection, and complement) [22, 7].

Additional algebraic properties, including aspects related to the pumping lemma and the structure of the accepting probabilities, have been further elaborated in works such as Ambainis et al. [5] and Xi et al. [53].

Summary of Advantages and Limitations

MO-1QFAs are praised for their simplicity. Since the quantum state evolves unitarily until a single measurement is made, the model avoids the complications associated with intermediate state collapses. This simplicity has practical implications; for example, recent experimental work has shown that custom control pulses can significantly reduce error rates in IBM-Q implementations [31], and photonic implementations have further demonstrated the feasibility of MO-1QFAs in optical setups [16]. On the downside, the acceptance power of MO-1QFAs is limited—when operating under bounded error, they recognize only the group languages, which form a strict subset of the regular languages. In contrast, MM-1QFAs offer greater acceptance power but at the cost of increased model complexity [25, 9].

Example

Consider a simple MO-1QFA defined over the unary alphabet $\Sigma = \{a\}$ with state set $Q = \{q_0, q_1\}$, initial state q_0, and accepting state set $F = \{q_1\}$. Let the unitary operator corresponding to the symbol a be defined by the rotation matrix:

$$U_a = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix},$$

for a fixed angle θ. For an input string a^n, the state evolves as:

$$|\Psi_{a^n}\rangle = U_a^n |q_0\rangle.$$

The acceptance probability is computed as:

$$p_M(a^n) = \|P\,|\Psi_{a^n}\rangle\|^2,$$

where the projection operator P is given by $P = |q_1\rangle\langle q_1|$. By appropriately choosing θ, the automaton can be tuned so that $p_M(a^n)$ exceeds the cut-point λ (for example, $\lambda = \frac{1}{2}$) if and only if a^n belongs to the target language. This example illustrates the essential mechanism of MO-1QFAs, as described in [36, 15].

Additional Topics

Learning and Optimization: Recent work by Chu et al. [18] has introduced methods that combine active learning with non-linear optimization to approximately learn the parameters of MO-1QFAs. These techniques provide insights into how one can recover the unitary transformations and state structure from observed data.

Complexity and Minimization: The problem of minimizing the number of states in an MO-1QFA was originally posed by Moore and Crutchfield [36]. Subsequent work by Mateus, Qiu, and Li [32] has established an EXPSPACE upper bound for the minimization problem, framing it as a challenge in solving systems of algebraic polynomial (in)equations.

Experimental Implementations: Experimental realizations of MO-1QFAs have also been explored. Lussi et al. [31] demonstrated an implementation on IBM-Q devices using custom control pulses, while photonic approaches have been reported in [16]. These works highlight both the practical challenges and the potential advantages of implementing MO-1QFAs on current quantum hardware.

Future Directions: Future research may focus on further enhancing experimental implementations, developing more robust learning algorithms for MO-1QFAs, and exploring new minimization techniques that could lead to more efficient automata. Extensions that combine features of MO-1QFAs and MM-1QFAs may also provide richer language recognition capabilities and deepen our understanding of quantum computational models.

3.1.2 Measure Many One-Way Quantum Finite Automaton

Introduction

MM-1QFAs are a variant of quantum finite automata in which a measurement is performed after reading each input symbol. Introduced by Kondacs and Watrous in 1997 [25], MM-1QFAs allow the automaton to collapse its quantum state at intermediate steps, thereby potentially influencing the computation dynamically. Although this mechanism can enhance the detection of accepting or rejecting conditions during the run, under the bounded error regime MM-1QFAs are known to recognize only a proper subset of the regular languages [15]. Recent work, such as by Lin [30], has provided elegant methods for addressing the equivalence problem of MM-1QFAs, further enriching our understanding of their computational properties.

Formal Definition

A Measure Many One-Way Quantum Finite Automaton (MM-1QFA) is defined as a 6-tuple

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

where:

- Q is a finite set of states,
- Σ is a finite input alphabet, typically augmented with an end-marker (e.g., \$),
- $\delta : Q \times \Sigma \times Q \to \mathbb{C}$ is the transition function, where for each symbol $\sigma \in \Sigma$ the corresponding matrix

  $$U_\sigma, \quad \text{with} \quad (U_\sigma)_{q,q'} = \delta(q, \sigma, q'),$$

  is unitary [25],
- $q_0 \in Q$ is the initial state,
- $Q_{acc} \subseteq Q$ is the set of accepting (halting) states, and
- $Q_{rej} \subseteq Q$ is the set of rejecting (halting) states.

After each symbol is read, the automaton's current state is measured with respect to the decomposition

$$E_{acc} = \text{span}\{|q\rangle : q \in Q_{acc}\}, \quad E_{rej} = \text{span}\{|q\rangle : q \in Q_{rej}\}, \quad E_{non} = \text{span}\{|q\rangle : q \in Q \backslash (Q_{acc} \cup Q_{rej})\}.$$

If the measurement outcome lies in E_{acc} or E_{rej}, the computation halts immediately with acceptance or rejection, respectively. This definition, adapted from Kondacs and Watrous [25] and refined in Lin [30], forms the basis of the MM-1QFA model.

Strings Acceptance

For an input string $x = x_1 x_2 \cdots x_n$, the MM-1QFA processes each symbol sequentially. At each step i, the unitary operator U_{x_i} is applied, followed by a measurement:

- If the measurement result falls in E_{acc}, the automaton immediately accepts x.
- If it falls in E_{rej}, the automaton rejects x.
- If the result lies in E_{non}, the computation continues with the next symbol.

The overall acceptance probability of x is the cumulative probability of all computation paths that eventually lead to an accepting state. In a bounded error framework, there exists a margin $\epsilon > 0$ such that for every $x \in L$, the acceptance probability satisfies

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

and for every $x \notin L$,

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

where λ is a predetermined cut-point (commonly set to $\frac{1}{2}$) [25, 15].

Set of Languages Accepted

Under the bounded error constraint, MM-1QFAs recognize a proper subset of the regular languages. In particular, the languages accepted by MM-1QFAs must satisfy specific algebraic properties that restrict their expressive power. Although MM-1QFAs can, in some cases, recognize nonregular languages when allowed unbounded error, the bounded error condition confines them to a class that is comparable to that of group languages [15, 25]. This limitation underscores the trade-off between the increased measurement frequency and the resultant reduction in language recognition capability.

Closure Properties

The language class recognized by MM-1QFAs with bounded error is known to enjoy several closure properties:

- It is closed under complement and inverse homomorphisms [15].
- It is closed under word quotients [15].
- However, the class is not closed under arbitrary homomorphisms [25, 7].

Recent work by Lin [30] further refines our understanding of these closure properties by addressing the equivalence problem for MM-1QFAs, thereby linking the structural properties of the recognized languages to the underlying automata.

Summary of Advantages and Limitations

MM-1QFAs offer notable advantages:

- The use of intermediate measurements can enable earlier detection of acceptance or rejection, potentially reducing the average computation time.
- The dynamic collapse of the quantum state provides a different balance between quantum coherence and classical decision-making.

Nevertheless, there are significant limitations:

- The frequent measurements interrupt the quantum evolution, which can limit the automaton's ability to harness quantum interference effectively.
- As a result, under bounded error conditions, MM-1QFAs recognize only a restricted subset of the regular languages.
- The complexity of analyzing and minimizing MM-1QFAs remains high, with state minimization posing an EXPSPACE challenge [32] and lower bound results highlighting the inherent state complexity [2].

Moreover, when compared to MO-1QFAs, MM-1QFAs may offer greater recognition power in some unbounded error scenarios but at the cost of increased computational and implementation complexity [25, 9].

Example

Consider an MM-1QFA defined over the alphabet $\Sigma = \{a\}$ with the state set $Q = \{q_0, q_1, q_2\}$, where q_0 is the initial state, $Q_{acc} = \{q_2\}$, and $Q_{rej} = \{q_1\}$. Let the unitary operator for the symbol a be given by:

$$U_a = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

The MM-1QFA processes an input string such as aa as follows:

1. Starting in state $|q\rangle$, the operator U_a is applied and a measurement is performed. The measurement may collapse the state into:
   - E_{acc} (state q_2) – leading to immediate acceptance,
   - E_{rej} (state q_1) – leading to immediate rejection, or
   - E_{non} (state q_0, in this example) – allowing the computation to continupe.
2. If the first measurement yields a non-halting result, the second symbol is processed in a similar manner. The overall acceptance probability is the sum of the probabilities of all computation paths that eventually result in an outcome within E_{acc}.

This example demonstrates the stepwise measurement process that characterizes MM-1QFAs [25, 30].

Additional Topics

Equivalence and Decision Problems: Lin [30] presents a simplified approach for deciding the equivalence of two MM-1QFAs by reducing the problem to comparing initial vectors, thereby streamlining the decision process.

State Complexity and Lower Bounds: Lower bound results for One-Way Quantum Finite Automaton (1QFA), such as those by Ablayev and Gainutdinova [2], provide insights into the inherent state complexity challenges that also impact MM-1QFAs.

Experimental Considerations: While experimental implementations have predominantly focused on MO-1QFAs due to their relative simplicity, future work may explore the adaptation of techniques (e.g., custom pulse shaping as demonstrated in [31]) to the more complex MM-1QFA framework.

3.2 Main Models of Quantum Finite Automata

3.2.1 One-Way Quantum Finite Automaton

1QFA are the quantum analog of classical one-way finite automata where the input tape is read from left to right without revisiting symbols. They provide a model for finite quantum computation that is more restricted than two-way quantum finite automata but often simpler to implement and analyze.

Among the most studied variants of 1QFA are the MO-1QFA and the MM-1QFA, which are addressed in detail in Section 3.1. These automata differ primarily in the timing of their measurements: MO-1QFA perform a single measurement at the end of the computation, while MM-1QFA perform measurements after reading each input symbol.

Beyond these foundational models, several other types of 1QFA have been proposed, each offering unique computational perspectives or enhancements. We present a comprehensive overview of such models, each in its own subsubsection.

Measure-Only One-Way Quantum Finite Automaton (MON-1QFA)

The MON-1QFA is a special model in which only measurement operations are used for computation, with no intermediate unitary evolutions. This model simplifies quantum computation by relying solely on projective measurements.

Formal Definition A MON-1QFA is defined as a tuple $A = (Q, \Sigma, \rho_0, \{P_\sigma\}_{\sigma\in\Sigma}, Q_{acc})$ where:

- Q is a finite set of states,
- Σ is a finite input alphabet,
- ρ_0 is the initial quantum state (density matrix),
- P_σ is the measurement operator for each input symbol σ,
- $Q_{acc} \subseteq Q$ is the set of accepting states.

Strings Acceptance Acceptance is determined by applying the appropriate projective measurement after each symbol and measuring the final state. Acceptance can be defined with a bounded-error threshold.

Sets of Languages Accepted The class of languages accepted by MON-1QFA is strictly less powerful than the class of regular languages. It corresponds to a particular class of regular languages known as literally idempotent piecewise testable languages [8].

Closure Properties The languages recognized by MON-1QFA are not closed under union or complementation [8].

Advantages and Limitations They offer a hardware-friendly model due to the absence of unitaries, but are strictly less powerful than MO-1QFA and MM-1QFA.

Comparison Compared to MO-1QFA and MM-1QFA, MON-1QFA are less powerful due to the absence of unitary evolution.

Example An example is the language of all strings over {a, b} with an even number of a's. It can be recognized by a suitable choice of projective measurements.

Additional Topics Measure-only models relate to trace monoids with idempotent generators and have been used in language algebraic characterizations [20, 8].

One-Way Quantum Finite Automata with Two Observables (1QFA(2))

Introduction The One-Way Quantum Finite Automaton with Two Observables (1QFA(2)) model introduces a second observable to the MO-1QFA framework, enhancing the capacity to distinguish input strings. It can be seen as an intermediate enhancement over classical MO-1QFA.

Formal Definition A 1QFA(2) is defined similarly to a standard MO-1QFA, but it includes two projective measurements applied in alternation during computation:

$$A = (Q, \Sigma, \rho_0, \{U_\sigma\}, \{P_1, P_2\}, Q_{acc})$$

where the two measurements P_1 and P_2 alternate throughout the computation.

Strings Acceptance A string is accepted based on the final outcome after alternating between the two measurements. Acceptance is typically defined with bounded error.

Sets of Languages Accepted The class of languages recognized is still a proper subset of regular languages, although strictly more than MO-1QFA.

Closure Properties These automata do not have known closure under union or intersection.

Advantages and Limitations The addition of a second observable enables more refined discrimination of inputs, albeit still with limited computational power.

Comparison The model is more expressive than MO-1QFA but remains less powerful than general MM-1QFA.

Additional Topics This line of work aligns with the broader research aim of incrementally extending the expressive power of 1QFA models, as seen in the approach of [19].

Two-Tape One-Way Quantum Finite Automata with Two Heads (2tQFA(2))

Introduction The Two-Tape One-Way Quantum Finite Automaton with Two Heads (2T1QFA(2)) model incorporates two tapes and two heads, enabling cross-comparison between symbols of the input and reference tape, enhancing the recognition of complex patterns.

Formal Definition Formally, a 2T1QFA(2) is a 7-tuple $A = (Q, \Sigma, \delta, q_0, Q_{acc}, Q_{rej}, \mathbb{T})$, where $\mathbb{T}$ denotes the tape set. The quantum evolution occurs in tandem over both tapes with corresponding heads.

Strings Acceptance Acceptance is determined through measurement at halting, typically with bounded error.

Sets of Languages Accepted 2T1QFA(2) can recognize some nonregular languages, exceeding the capabilities of classical one-way automata and standard 1QFA models [23].

Closure Properties Closure properties remain largely unexplored, with no known results under union or complement.

Advantages and Limitations The addition of a second tape expands the computational power substantially, though it also introduces practical complexity.

Comparison Outperforms classical and most one-way quantum models in expressive power.

Additional Topics This model exemplifies a practical direction in extending 1QFA expressivity by architectural enhancement.

Nondeterministic Quantum Finite Automata (NQFA)

Introduction Non-Deterministic Quantum Finite Automaton (NQFA) introduce nondeterminism in the quantum setting. Unlike probabilistic nondeterminism, here the nondeterminism arises from quantum measurement outcomes and amplitude branches.

Formal Definition An NQFA is a 5-tuple $A = (Q, \Sigma, \psi_0, \{U_\sigma\}, Q_{acc})$, and uses a cutpoint acceptance mode, where a string x is accepted if $\mathbb{P}(x) > \lambda$ for some threshold λ.

Strings Acceptance NQFA recognize strings based on acceptance with nonzero amplitude, allowing acceptance of nonregular languages with bounded error [55].

Sets of Languages Accepted This model strictly recognizes more than regular languages, offering power comparable to or exceeding classical nondeterministic finite automata.

Closure Properties NQFA are not closed under complement or union.

Advantages and Limitations The model is powerful yet lacks constructive methods for deterministic acceptance, making some verification tasks harder.

Comparison More expressive than MO-1QFA, MM-1QFA, and even probabilistic finite automata in some cases.

Reversible Quantum Finite Automata (RevQFA)

Introduction Reversible One-Way Quantum Finite Automaton (RevQFA) enforce reversibility in state transitions, in line with quantum mechanics principles, where computation steps are invertible.

Formal Definition Defined similarly to MO-1QFA, but all transitions are reversible, and state evolution is enforced to be unitary across all paths. Measurement occurs only at the end.

Strings Acceptance A string is accepted based on the final state post unitary evolution.

Sets of Languages Accepted Recognizes all regular languages [56], differing from many other 1QFA models.

Closure Properties Closed under Boolean operations due to equivalence with classical deterministic automata.

Advantages and Limitations They enjoy a strong correspondence to classical reversible automata with quantum efficiency benefits, though do not surpass regular languages.

Comparison Unlike most 1QFA, RevQFA can simulate any DFA, thus closing the gap between quantum and classical finite automata in expressiveness.

Additional Topics The interest in RevQFA stems from the desire to harness quantum reversibility for computation, a core direction explored by [19].

3.2.2 Two-way Quantum Finite Automata

QFAs can be classified based on the measurement policy and head movement direction. In this subsection, we focus exclusively on Two-Way Quantum Finite Automata (2QFAs) models that operate under pure quantum evolution: namely, the Measure Once Two-Way Quantum Finite Automaton (MO-2QFA) and Measure Many Two-Way Quantum Finite Automaton (MM-2QFA). These automata extend the capabilities of their one-way counterparts by allowing bidirectional movement of the tape head, while differing in how often measurements are performed during computation.

Measure-Once Two-Way Quantum Finite Automata (MO-2QFA)

The MO-2QFA was formally introduced by Xi et al. [53] as the natural two-way extension of the (MO-1QFA). It performs unitary transformations while scanning the input in both directions but conducts a projective measurement only once—at the end of the input.

Formal Definition. An MO-2QFA is defined as a 5-tuple $M = (Q, \Sigma, \delta, q_0, Q_a)$ where:

- Q is a finite set of quantum states,
- Σ is a finite input alphabet,
- $\delta : Q \times \Gamma \times Q \times D \to \mathbb{C}$ is the transition function with $\Gamma = \Sigma \cup \{\#, \$\}$ and $D = \{-1, 0, +1\}$ indicating tape head movement,
- $q_0 \in Q$ is the initial state,
- $Q_a \subseteq Q$ is the set of accepting states.

The unitary evolution is enforced through conditions of orthogonality and separability as outlined in [53].

Strings Acceptance. Acceptance is defined via a single projective measurement after the entire input string, delimited by endmarkers, is processed. A string is accepted with a probability computed from the projection onto the accepting subspace. The model supports acceptance with bounded error, cutpoint, and exact acceptance depending on configuration [53].

Set of Languages Recognised. The class of languages recognized by MO-2QFA strictly contains the languages recognized by MO-1QFA, and it includes some nonregular languages. In fact, MO-2QFA can recognize proper supersets of group languages and supports complex operations such as intersection and reversal [53].

Closure Properties. The languages recognized by MO-2QFA are closed under union, intersection, complement, and reversal. Notably, the closure under intersection and union can be achieved by direct sum and tensor product constructions, respectively [53].

Advantages and Limitations. The main advantage of MO-2QFA lies in its improved recognition power over one-way models. However, the measurement at the end implies potentially high computational complexity for simulating classical behavior. Its limitation includes the reliance on exact unitary constraints and nontrivial implementation challenges [53].

Comparison. Compared to MO-1QFA, the MO-2QFA is strictly more powerful due to its bidirectional scanning ability. It is less expressive than MM-2QFA in general, since the latter allows more frequent measurements and thus a richer computational structure.

Example. An example from [53] shows that MO-2QFA can recognize the language $L = \{a^n b^n \mid n \geqslant 1\}$ with bounded error—something not possible with any one-way QFA model.

Additional Topics. The authors suggest future directions include studying the closure under concatenation in more general terms and optimizing state complexity for specific regular operations [53].

Measure-Many Two-Way Quantum Finite Automata (MM-2QFA)

Introduction. The MM-2QFA was introduced by Kondacs and Watrous [25] and represents the most powerful pure QFA model in terms of language recognition. It performs a measurement at each computational step and allows head movement in both directions.

Formal Definition. An MM-2QFA is a 6-tuple $M = (Q, \Sigma, \delta, q_0, Q_a, Q_r)$ where:

- Q is the finite set of states partitioned into Q_n (non-halting), Q_a (accepting), and Q_r (rejecting) states,
- $\delta : Q \times \Gamma \times Q \times D \to \mathbb{C}$ is the transition function,
- The machine performs a projective measurement at each step to determine whether to accept, reject, or continue.

Well-formedness constraints are necessary to ensure unitarity and validity of transition amplitudes [25].

Strings Acceptance. A string is accepted if, during any computational step, the machine transitions into an accepting state. This supports acceptance with bounded error, cutpoint, and exact modes depending on the configuration of the measurement [25].

Set of Languages Recognised. The MM-2QFA can recognize nonregular languages such as $L_{eq} = \{a^n b^n \mid n \geqslant 0\}$ with bounded error in linear time. This is a significant enhancement over any classical 2PFA or 1QFA model [25].

Closure Properties. The class of languages recognized by MM-2QFA is not known to be closed under union or intersection. This is a key limitation of the model despite its high expressive power.

Advantages and Limitations. The model can recognize nonregular languages efficiently, offering exponential advantages in space over classical automata. However, it is more complex to analyze due to frequent measurements and lacks closure under basic operations [25, 43].

Comparison. MM-2QFA is strictly more powerful than all one-way models (MO-1QFA, MM-1QFA) and more powerful than MO-2QFA in general. However, its structure is harder to simulate and analyze, limiting its practical implementation.

Example. Kondacs and Watrous [25] present a MM-2QFA that recognizes L_{eq} in linear time with bounded error—a feat requiring exponential time for classical two-way probabilistic automata.

Additional Topics. Extensions to hybrid models such as Two-Way Quantum Finite Automata with Classical States (2QCFAs) and Quantum Interactive Proof (QIP) with 2QFA verifiers have been proposed to harness the strengths of MM-2QFA while addressing its limitations [40, 43].

3.2.3 Hybrid Quantum Finite Automata

Hybrid quantum finite automata (HQFA) are finite-state machines that combine a quantum state component with a classical state component. In essence, an HQFA consists of a quantum system and a classical finite automaton that operate together, communicating information between them during the computation [27].

In all these models, the goal is to leverage a small quantum memory together with classical states to recognize languages, potentially with far fewer states than a purely classical automaton would require [60].

Several important hybrid QFA models have been introduced. Ambainis and Watrous [4] first proposed the two-way quantum finite automaton with classical states (2QCFA), which augments a two-way deterministic finite automaton with a constant-size quantum register.

More advanced hybrids include multi-tape extensions like the Two-Tape Quantum Finite Automaton with Classical States (2TQCFA) and k-Tape Quantum Finite Automaton with Classical States (kTQCFA), which increase computational power by leveraging multiple input tapes [58]. All one-way hybrid QFA models (One-Way Quantum Finite Automaton with Classical States (1QCFA), 1QFAC, One-Way Quantum Finite Automaton with Control Language (CL-1QFA)) recognize exactly the regular languages [27, 60]. However, they can be significantly more *succinct* in terms of state complexity than classical models [54].

One-Way Quantum Finite Automaton with Classical States (1QCFA)

The 1QCFA model, introduced by Zheng, Qiu, Li, and Gruska [60], augments a classical one-way finite automaton with a quantum component. It can be seen as a one-way restriction of the 2QCFA model. The 1QCFA features two-way communication between classical and quantum parts: the classical state influences the quantum operation applied, and the quantum measurement outcome affects the next classical state.

Formal Definition: A 1QCFA is a 9-tuple:

$$A = (Q, S, \Sigma, C, q_1, s_1, \{\Theta_{s,\sigma}\}, \delta, S_a)$$

where:

- Q: finite set of quantum basis states,
- S: finite set of classical states,
- Σ: input alphabet,
- C: measurement outcomes,
- $q_1 \in Q$, $s_1 \in S$: initial quantum and classical states,
- $\Theta_{s,\sigma}$: quantum operation with outcomes in C,
- $\delta : S \times \Sigma \times C \to S$: classical transition function,
- $S_a \subseteq S$: accepting states.

Computation begins in $(s_1, |q_n\rangle)$, proceeds symbol-by-symbol. On input $x = x_1x_2 \cdots x_n$, the machine updates classical and quantum states based on outcomes $c_i \in C$ generated by each Θ_{s_i,x_i}, applying $\delta(s_i, x_i, c_i)$ at every step. Acceptance is determined by whether the final classical state is in S_a [27].

Strings Acceptance: A 1QCFA accepts string x with probability based on outcome paths $c_1c_2 \cdots c_n$. If the classical state ends in S_a, it accepts. Languages are recognized with bounded error $\varepsilon < 1/2$ if acceptance probability is $\geqslant 1 - \varepsilon$ for all $x \in L$ and $\leqslant \varepsilon$ for all $x \notin L$ [27, 60].

Sets of Languages Recognized: 1QCFA recognize exactly the class of regular languages [60]. Any regular language can be recognized by some 1QCFA with certainty. Moreover, they can be exponentially more succinct than DFA for some regular languages [54].

Closure Properties: The class of languages recognized by 1QCFA is closed under union, intersection, and complement, as it coincides with the regular languages [27].

Advantages and Limitations: While 1QCFA do not exceed DFA in language power, they are often exponentially more state-efficient. For instance, certain periodic languages can be recognized with a single qubit: rotating the quantum state by $2\pi/p$ for each a and measuring at the end to detect if the state returned to its initial position (full rotation) [54, 13].

Comparison Between Models: 1QCFA generalize CL-1QFA and 1QFAC. The main difference lies in bidirectional communication. In comparison to 2QCFA, 1QCFA are weaker, since they cannot move backward on the input or recognize non-regular languages [27].

Example: A 1QCFA can recognize the language $L = \{a^n : n \equiv 0 \mod p\}$ using a single qubit: rotating the quantum state by $2\pi/p$ for each a and measuring at the end to detect if the state returned to its initial position (full rotation) [12].

Additional Topics: 1QCFA equivalence is decidable [27]. State trade-offs between classical and quantum resources have been studied extensively [48, 54]. Future work includes refining state complexity bounds and exploring minimal configurations.

One-Way Quantum Finite Automaton with Control Language (CL-1QFA)

Introduction: The CL-1QFA (Quantum Finite Automata with Control Language) model, introduced by Bertoni, Mereghetti, and Palano [34], consists of a quantum component responsible for unitary operations and measurements, and a classical DFA that processes the sequence of measurement outcomes. The role of the control language is to guide acceptance: a word is accepted if and only if the sequence of measurement outcomes belongs to a regular language defined by the control automaton.

Formal Definition: A CL-1QFA is a 6-tuple:

$$A = (Q, \Sigma, \{U_\sigma\}_{\sigma \in \Sigma}, q_0, M, L)$$

where:

- Q: finite set of quantum basis states,
- Σ: input alphabet,
- U_σ: unitary transformation applied upon reading symbol σ,
- $q_0 \in Q$: initial quantum state,
- M: projective measurement with outcomes in a finite set Γ,
- $L \subseteq \Gamma^*$: regular control language recognized by a classical DFA.

On input $x = x_1 x_2 \cdots x_n$, the automaton applies U_{x_i} and measures after each step, producing an output string $y \in \Gamma^n$. The word x is accepted if $y \in L$.

Strings Acceptance: Acceptance depends entirely on whether the sequence of quantum measurement results belongs to the control language L. This model generally uses bounded-error acceptance. However, exact acceptance is possible for certain languages, depending on the control DFA and quantum measurements [34].

Sets of Languages Recognized: CL-1QFA recognize exactly the class of regular languages [27]. Though they do not surpass the regular class in power, their structure allows for different expressive strategies by decoupling quantum operations from classical verification.

Closure Properties: The class of languages recognized is closed under union, intersection, and complement because the control language is regular and the measurement outcomes are deterministic modulo quantum probabilities [27].

Advantages and Limitations: Advantages include clear separation between quantum processing and classical control, which simplifies modular design and analysis. The main limitation is that the model cannot recognize non-regular languages and does not allow dynamic feedback between quantum and classical components [27].

Comparison Between Models: CL-1QFA can be simulated by 1QCFA [27], but not vice versa. Unlike 1QCFA, CL-1QFA do not allow two-way communication: measurement outcomes do not affect the ongoing quantum state. This makes CL-1QFA structurally simpler, but less expressive in practice.

Example: To recognize $L = (ab)^*$, a CL-1QFA can measure each input symbol's quantum effect and produce a binary output: '0' for a, '1' for b. The control DFA can then accept only if the output string alternates properly and has even length [34].

Additional Topics: Variants of CL-1QFA using unary control languages have been recently proposed to explore succinctness and unary acceptance conditions [35]. These help analyze minimal state configurations and potential hardware implementations.

Two-Way Quantum Finite Automaton with Classical States (2QCFA)

Introduction: The 2QCFA model, introduced by Ambainis and Watrous [4], is a hybrid automaton that consists of a classical two-way deterministic finite control and a constant-size quantum register. It was designed to exploit quantum computation while maintaining classical control over input movement. This makes the model both powerful and physically realizable, allowing the classical part to handle head movement and state tracking, while the quantum component processes information probabilistically.

Formal Definition: A 2QCFA is a 9-tuple:

$$A = (Q, S, \Sigma, \Theta, \delta, q_0, s_0, S_a, S_r)$$

where:

- Q: finite set of quantum basis states,
- S: finite set of classical states,
- Σ: input alphabet,
- Θ: quantum transition function defining unitary operators or measurements,
- δ: classical transition function based on current state, symbol, and measurement outcome,
- $q_0 \in Q$, $s_0 \in S$: initial quantum and classical states,
- S_a, S_r: sets of accepting and rejecting classical states.

The classical control can move the tape head both left and right. The quantum state is manipulated via unitary transformations or measurements, which are determined by the classical state and scanned symbol. Decisions are made based on both the classical and quantum information.

Strings Acceptance: 2QCFA accept strings using bounded error or with one-sided error. For example, languages like $L_{eq} = \{a^n b^n \mid n \geqslant 1\}$ can be accepted with one-sided bounded error in expected polynomial time [4].

Sets of Languages Recognized: 2QCFA can recognize certain non-regular languages, including L_{eq} and palindromes over unary alphabets, which makes them strictly more powerful than classical DFA or one-way QFA [4, 27].

Closure Properties: The class of languages recognized by 2QCFA is not closed under union or intersection, due to the constraints of the probabilistic error bounds and two-way head movement. However, they maintain closure under reversal and concatenation in specific cases [27].

Advantages and Limitations: Advantages include greater recognition power than 1QCFA and succinctness for certain problems. For example, 2QCFA can recognize L_{eq} with only a constant-size quantum register and logarithmic classical states [51]. However, they are generally limited to languages where probabilistic techniques suffice, and their runtime is often polynomial in the worst case [50].

Comparison Between Models: 2QCFA generalize 1QCFA by allowing two-way head movement, which significantly increases computational power. In contrast to CL-1QFA, they use dynamic feedback from the quantum measurements to the classical state transitions. 2QCFA are more expressive but harder to analyze due to interaction complexity [61].

Example: The language $L_{eq} = \{a^n b^n \mid n \geqslant 1\}$ can be recognized by a 2QCFA by using the quantum register to randomly check positions and probabilistically verify balance between a's and b's through repeated subroutines [4].

Additional Topics: Future work includes better understanding the time complexity of 2QCFA algorithms and developing minimization techniques. Variants include alternating 2QCFA and state-succinct encodings [61, 50].

Two-Tape Quantum Finite Automaton with Classical States (2TQCFA)

Introduction: The 2TQCFA model (Two-Tape Quantum-Classical Finite Automata) extends the 2QCFA by using two input tapes instead of one. Introduced by Zheng, Li, and Qiu [58], this model enhances computational power by enabling comparisons and synchronized traversal of two input strings. The quantum component remains fixed in size, while the classical controller can move the heads on both tapes and perform transitions based on measurements.

Formal Definition: A 2TQCFA is formally a tuple similar to a 2QCFA but with two input tapes:

$$A = (Q, S, \Sigma, \Theta, \delta, q_0, s_0, S_a, S_r)$$

with the following distinctions:

- Two input heads, each reading a separate string from Σ^*,
- Classical state $s \in S$ determines movement and operation on each tape head,
- Quantum operations Θ depend on the symbols scanned by both heads and classical state.

As in 2QCFA, the automaton evolves through interactions between classical and quantum transitions, but the two-tape structure allows for cross-input comparisons.

Strings Acceptance: 2TQCFA can accept languages using bounded-error acceptance, typically with one-sided error. A notable example includes the language $L = \{w\#w \mid w \in \{a, b\}^*\}$, which is non-regular and not recognizable by 2QCFA, but accepted by 2TQCFA using synchronous traversal of both input halves [58].

Sets of Languages Recognized: The language recognition power of 2TQCFA includes certain context-free and non-regular languages not recognizable by 2QCFA. Thus, 2TQCFA strictly extends the power of 2QCFA under bounded-error acceptance [58].

Closure Properties: Due to the added complexity of two-tape processing, closure properties are less well-defined. However, the model is still limited by finite memory and cannot recognize arbitrary context-free languages [27].

Advantages and Limitations: The primary advantage is an increased ability to perform input comparisons, useful for palindromes or equality checks. The main limitations include increased implementation complexity and difficulties in analyzing language classes and performance bounds.

Comparison Between Models: 2TQCFA extend 2QCFA in power by enabling comparisons across two tapes. Unlike kTQCFA (which generalize even further), 2TQCFA remain practical for checking mirrored or related substrings. Compared to 1QCFA and CL-1QFA, they are significantly more powerful in terms of language recognition.

Example: To recognize $L = \{w\#w\}$, a 2TQCFA reads w on the first tape and stores information in the quantum register. It then compares this with the second half of the input on the second tape. Probabilistic subroutines are used to ensure correctness with bounded error [58].

Additional Topics: Variants of multi-tape quantum automata have been studied to explore even richer classes. Open problems include characterizing all non-regular languages recognizable by 2TQCFA with polynomial expected runtime.

k-Tape Quantum Finite Automaton with Classical States (kTQCFA)

Introduction: The kTQCFA model generalizes the two-tape quantum-classical automaton to an arbitrary finite number k of input tapes. This model, proposed in subsequent works building upon the 2TQCFA model [58], enhances the automaton's ability to process complex language patterns by allowing simultaneous access to multiple strings. Each tape is read by an independent head, all coordinated by a classical control unit and a constant-size quantum register.

Formal Definition: A kTQCFA is defined similarly to a 2TQCFA but with k tapes and k input heads. The formal components include:

$$A = (Q, S, \Sigma, \Theta, \delta, q_0, s_0, S_a, S_r)$$

with modifications:

- k input tapes, each with its own head,
- Classical state transitions δ depend on the symbols read from all k heads and outcomes of quantum operations,

- Quantum transitions Θ may vary based on any combination of input symbols and classical state.

The automaton reads the tapes simultaneously and updates its classical and quantum states accordingly, with acceptance determined by reaching a state in S_a.

Strings Acceptance: Acceptance is based on bounded-error criteria. This model can accept languages requiring coordinated comparisons across multiple strings, such as interleaving or mirror structures, which are beyond the capability of 2QCFA or 2TQCFA.

Sets of Languages Recognized: kTQCFA can recognize certain languages that lie outside the class of regular and some context-free languages. It provides a hierarchical extension in power with increasing k, where $k = 1$ corresponds to 1QCFA and $k = 2$ to 2TQCFA [27].

Closure Properties: Due to increasing complexity with larger k, closure properties are less explored. They inherit the limited closure of 2TQCFA but allow more expressive constructions for language families.

Advantages and Limitations: The main advantage of kTQCFA is scalability of pattern comparison and cross-tape logic. However, this comes at a cost: managing multiple heads and quantum-classical interactions becomes increasingly complex, both analytically and in potential physical realization.

Comparison Between Models: kTQCFA generalize all previously discussed models. While more powerful, they are also less practical for current quantum computing technologies. Unlike 1QCFA or CL-1QFA which are implementable with simpler setups, kTQCFA require sophisticated synchronization mechanisms.

Example: A 3TQCFA can accept a language like $L = \{(x, y, z) \mid x = y = z\}$ by comparing the three inputs simultaneously, performing probabilistic checks using quantum subroutines and classical tracking over each input position.

Additional Topics: Future work may include classification of languages based on minimal k required, complexity of simulations by smaller models, and physical feasibility of multi-tape implementations in quantum automata.

3.2.4 Quantum Finite Automata with Counters

Quantum finite automata with counters extend the computational power of QFA by incorporating classical or quantum counters into the system. These models provide hybrid computational capabilities where quantum state transitions are influenced by the counter value and vice versa, enabling the recognition of certain non-regular languages with bounded error which classical counterparts fail to recognize.

In the literature, various models of QFA with counters have been proposed. This subsection explores the prominent models including Quantum Finite One-Counter Automaton (QF1CA), Two-Way Quantum Finite One-Counter Automaton (2QF1CA), Quantum Finite k-Counter Automaton (1QFkCA), and Real-Time Quantum One-Counter Automaton (RTQ1CA), detailing their structure, properties, capabilities, and limitations based on foundational work such as [14, 26, 40, 17].

Quantum Finite One-Counter Automata (QF1CA)

A QF1CA is a one-way QFA that uses a classical counter, capable of incrementing or decrementing its value and testing for zero. This model merges quantum transitions with classical counter logic, providing a new pathway for recognizing languages beyond the regular set [26].

Formal Definition Formally, a QF1CA consists of a finite set of states Q, an input alphabet Σ, a classical counter with values in $\mathbb{Z}$, and a transition function $\delta : Q \times \Sigma \times \{0, 1\} \times Q \times \{-1, 0, 1\} \to \mathbb{C}$ where $\{0, 1\}$ indicates whether the counter is zero or not. The system operates unitarily, with counter updates contingent on the current state and symbol read.

Strings Acceptance QF1CA can accept strings using bounded-error probabilistic acceptance. They can recognize non-regular languages such as $L_1 = \{w \in \Sigma^* :$ equal number of 0's and 1's in $x\}$ when augmented with additional structure in the input [14].

Sets of Languages Accepted The class of languages accepted by QF1CA with bounded error properly includes the class of languages accepted by classical deterministic and probabilistic one-counter automata [14].

Closure Properties Closure properties are limited and not thoroughly investigated; however, QF1CA do not maintain closure under union or intersection due to non-closure in the classical probabilistic case.

Advantages and Limitations A notable advantage is the ability to recognize certain context-free languages with bounded error. However, limitations stem from counter-based non-reversibility and measurement-induced collapses which reduce robustness.

Comparison Compared to 1QFA or MO-1QFA, QF1CA exhibit significantly higher computational power due to the counter's memory augmentation.

Example A QF1CA recognizing the language L_1 as shown above was constructed in [14], showing correct acceptance probabilities distinguishing it from deterministic models.

Additional Topics Current research investigates the influence of quantum control on counter updates and the simulation of classical pushdown automata using counters in hybrid quantum settings.

Two-Way Quantum Finite One-Counter Automata (2QF1CA)

Introduction 2QF1CA enhances the QF1CA model by allowing two-way head movement on the input tape, significantly expanding computational capabilities. This flexibility enables the automaton to reprocess information with context, analogous to two-way classical finite automata but equipped with quantum transitions and a counter.

Formal Definition A 2QF1CA is defined by a tuple $(Q, \Sigma, \delta, q_0, Q_a, Q_r)$, where δ maps configurations including direction: $\delta : Q \times \Sigma \times \{0, 1\} \times Q \times \{-1, 0, 1\} \times \{-1, 0, 1\} \to \mathbb{C}$. The last component indicates the head movement (-1 for left, 0 for stay, 1 for right), and the counter updates accordingly.

Strings Acceptance 2QF1CA can recognize more complex languages such as L_2 from [14], composed of multiple L_1 segments demarcated by control symbols. These languages are not recognizable by 1QF1CA or classical probabilistic variants.

Sets of Languages Accepted These automata can accept languages outside deterministic and probabilistic one-counter automata capabilities, establishing a broader language class, including some context-sensitive languages under bounded error.

Closure Properties Closure under complement and intersection is not generally guaranteed due to quantum nondeterminism and measurement dependencies. Formal closure results remain limited.

Advantages and Limitations The key strength of 2QF1CA lies in its bidirectional input scanning which provides significant advantages in language parsing. However, unitarity and interference management become more complex.

Comparison Compared to QF1CA, the two-way model shows enhanced language recognition at the cost of more complex design and verification.

Example Recognition of L_2 involving interleaved structures demonstrates the superiority of 2QF1CA over classical and one-way quantum models as outlined in [14].

Additional Topics Further topics include automaton minimization, real-time simulation constraints, and efficient quantum algorithm implementation.

One-Way Quantum Finite k-Counter Automata (1QFkCA)

Introduction The 1QFkCA model generalizes the QF1CA by including k classical counters. Each counter is independently incremented, decremented, or checked against zero, enabling multi-dimensional memory augmentation in the quantum control logic [17].

Formal Definition Formally, a 1QFkCA is given by a transition function $\delta : Q \times \Sigma \times \{0, 1\}^k \times Q \times \{-1, 0, 1\}^k \to \mathbb{C}$ with the counter vector defining current zero/non-zero statuses and updates.

Strings Acceptance Languages involving multiple numeric relationships, such as $L = \{a^n b^n c^n \mid n \geqslant 1\}$, can be recognized in bounded error by appropriately configured 1QFkCA.

Sets of Languages Accepted These automata recognize a subset of context-sensitive languages and are more powerful than all one-counter automata, quantum or classical.

Closure Properties Closure under intersection and union becomes feasible with k counters, particularly when structured synchronization is used in parallel counters.

Advantages and Limitations Their capability to recognize complex dependencies is advantageous, but the exponential state complexity and entangled counter management are practical limitations.

Comparison Compared to QF1CA, this model is exponentially more powerful but with higher operational complexity.

Example In [17], a 1QFkCA was shown to recognize the language $a^n b^n c^n$ via three synchronized counters incremented and decremented according to the current segment of the input.

Additional Topics Potential topics include quantum counter compression, fault tolerance in counters, and counter sharing protocols in hybrid quantum-classical systems.

Realtime Quantum One-Counter Automata (rtQ1CA)

Introduction RTQ1CA represents a restricted subclass of QF1CA in which the input head moves strictly right at each step, processing the input in real-time. This model explores trade-offs between real-time operation and computational power [17].

Formal Definition Defined similarly to QF1CA but with a strict constraint on the transition direction (right only). The transition function thus omits head direction: $\delta : Q \times \Sigma \times \{0, 1\} \times Q \times \{-1, 0, 1\} \to \mathbb{C}$.

Strings Acceptance Though more limited, RTQ1CA can still recognize several non-regular languages with carefully crafted transition amplitudes and counter updates.

Sets of Languages Accepted Their accepted languages lie strictly between those of MO-1QFA and QF1CA due to the real-time restriction.

Closure Properties Due to strict real-time behavior and interference effects, closure properties are even more restricted.

Advantages and Limitations The main advantage is speed and simplicity in implementation, but at a significant cost to recognition power compared to QF1CA or 2QF1CA.

Comparison RTQ1CA are less powerful than general QF1CA, but more powerful than classical real-time automata due to quantum parallelism.

Example An RTQ1CA can probabilistically accept strings with a balanced number of 0's and 1's using only real-time passes and interference.

Additional Topics Real-time simulation fidelity and circuit-based implementations of RTQ1CA models are open areas of study.

3.2.5 Generalised Quantum Finite Automata

Generalised Quantum Finite Automata (1gQFAs) extend the standard QFAs by replacing the usual unitary-based state transitions with the most general physically admissible maps—namely, trace-preserving quantum operations. This modification permits non-unitary evolution, allowing the automata to simulate probabilistic and classical automata while still operating with finite memory. Nevertheless, it has been shown that both the measure-once and measure-many versions of 1gQFA recognize exactly the regular languages (with bounded error) [28].

Measure Once Generalised Quantum Finite Automaton

A Measure Once Generalised Quantum Finite Automaton (MO-1gQFA) generalizes the traditional MO-1QFA by allowing each input symbol to trigger a trace-preserving quantum operation (instead of a unitary transformation) on the system. In this model, no measurement is performed during the reading of the input; a single projective measurement is executed only at the end to decide acceptance or rejection [28].

Formal Definition. An MO-1gQFA is defined as the quintuple

$$M = \{\mathcal{H}, \Sigma, \rho_0, \{\mathcal{E}_\sigma\}_{\sigma\in\Sigma}, P_{acc}\},$$

where

- $\mathcal{H}$ is a finite-dimensional Hilbert space,
- Σ is a finite input alphabet,
- $\rho_0 \in D(\mathcal{H})$ is the initial density operator,
- For each $\sigma \in \Sigma$, the state transition is given by the trace-preserving quantum operation
$$\mathcal{E}_\sigma(\rho) = \sum_k \mathcal{E}_{\sigma,k}\,\rho\,\mathcal{E}_{\sigma,k}^\dagger, \quad \text{with } \sum_k \mathcal{E}_{\sigma,k}^\dagger \mathcal{E}_{\sigma,k} = I,$$
- P_{acc} is a projector on the accepting subspace of $\mathcal{H}$ (with the complementary projector $P_{rej} = I - P_{acc}$).

On an input string $x = \sigma_1\sigma_2\cdots\sigma_n \in \Sigma^*$ the automaton evolves as

$$\rho_x = \mathcal{E}_{\sigma_n} \circ \mathcal{E}_{\sigma_{n-1}} \circ \cdots \circ \mathcal{E}_{\sigma_1}(\rho_0),$$

and a final measurement in the basis $\{P_{acc}, P_{rej}\}$ is performed. The acceptance probability is defined by

$$f_M(x) = \mathrm{Tr}\big(P_{acc}\,\rho_x\big).$$

Strings Acceptance. A string $x \in \Sigma^*$ is accepted by M if the acceptance probability meets the specified criterion. Common acceptance criteria include:

1. **Bounded Error:** There exist a threshold $\lambda \in (0,1]$ and an error margin $\epsilon > 0$ such that
$$f_M(x) \geqslant \lambda + \epsilon \quad \text{if } x \in L,$$
$$f_M(x) \leqslant \lambda - \epsilon \quad \text{if } x \notin L.$$

2. **Cutpoint Acceptance:** x is accepted if $f_M(x) > \lambda$, where λ is an isolated cutpoint.

3. **Exact Acceptance:** In certain constructions (e.g., when simulating a deterministic finite automaton) one has $f_M(x) = 1$ for accepted strings and $f_M(x) = 0$ for rejected strings.

Set of Languages Accepted. It has been proved that under the bounded error criterion, MO-1gQFA recognize precisely the class of regular languages. That is, for every regular language there exists an MO-1gQFA recognizing it, and every language recognized by an MO-1gQFA is regular [28].

Closure Properties. The class of languages recognized by MO-1gQFA is closed under several standard operations:

- **Union and Intersection:** By suitable constructions (e.g., via direct sums and tensor products), if L_1 and L_2 are recognized by MO-1gQFA then so are $L_1 \cup L_2$ and $L_1 \cap L_2$.
- **Complementation:** Replacing P_{acc} with its complement $I - P_{acc}$ yields an automaton for the complement language.
- **Inverse Homomorphism and Concatenation with Regular Languages:** These operations preserve the regularity of the language.

Summary of Advantages and Limitations. The MO-1gQFA model is advantageous due to its structural simplicity—requiring only a final measurement—and its ability to simulate classical probabilistic automata exactly via general trace-preserving operations. However, despite the broadened operational framework, its computational power remains confined to recognizing regular languages (with bounded error). Furthermore, the state minimization problem for MO-1gQFA is known to be EXPSPACE-hard [32].

Example. An example is provided by the simulation of a deterministic finite automaton (DFA) for the language
$$L = a^* b^*.$$
Here, one chooses
$$\mathcal{H} = \operatorname{span}\{|q_n\rangle, |q_n\rangle, \ldots, |q_n\rangle\},$$
sets the initial state as $\rho_0 = \sum_i \pi_i \, |q_n\rangle\langle q_i|$ (with $\{\pi_i\}$ given by the DFA's initial distribution), and defines each operation $\mathcal{E}_\sigma$ so that for each basis state $|q_n\rangle$,
$$\mathcal{E}_\sigma\big(|q_n\rangle\langle q_i|\big) = \sum_j A(\sigma)_{ij} \, |q_n\rangle\langle q_j| \, ,$$

where $A(\sigma)$ is the stochastic matrix corresponding to the DFA's transition function. The final measurement is performed using

$$P_{acc} = \sum_{q_i \in F} |q_n\rangle\langle q_i| \,,$$

where F is the set of accepting states. This construction ensures that the acceptance probability $f_M(x)$ replicates the behavior of the DFA [28].

Additional Topics. Further research on MO-1gQFA includes the equivalence problem, where necessary and sufficient conditions are derived based on the linear span of the reachable density operators. In addition, advanced state minimization techniques have been developed, reducing the minimization problem to solving systems of polynomial inequalities with an EXPSPACE upper bound [33].

Measure Many Generalised Quantum Finite Automaton

Introduction. Measure Many Generalised Quantum Finite Automaton (MM-1gQFA) extend the MO-1gQFA model by performing a measurement after processing each input symbol. In this model, after each trace-preserving quantum operation corresponding to a symbol, a projective measurement is executed that partitions the state space into three mutually orthogonal subspaces—namely, the accepting subspace, the rejecting subspace, and the non-halting subspace. If the outcome lies in the accepting or rejecting subspace, the computation halts immediately; otherwise, it continues with the next symbol [28].

Formal Definition. An MM-1gQFA is defined as the 6-tuple

$$M = \{\mathcal{H}, \Sigma, \rho_0, \{\mathcal{E}_\sigma\}_{\sigma \in \Sigma \cup \{¢, \$\}}, \mathcal{H}_{acc}, \mathcal{H}_{rej}\},$$

where

- $\mathcal{H}$ is a finite-dimensional Hilbert space that decomposes as

  $$\mathcal{H} = \mathcal{H}_{acc} \oplus \mathcal{H}_{rej} \oplus \mathcal{H}_{non},$$

  with $\mathcal{H}_{acc}$ and $\mathcal{H}_{rej}$ denoting the accepting and rejecting subspaces, and $\mathcal{H}_{non}$ the non-halting subspace;
- Σ is a finite input alphabet, and the symbols ¢ and \$ serve as the left and right end-markers, respectively;
- $\rho_0 \in D(\mathcal{H})$ is the initial state with $\text{supp}(\rho_0) \subseteq \mathcal{H}_{non}$;
- For each $\sigma \in \Sigma \cup \{¢, \$\}$, the state transition is given by the trace-preserving quantum operation

  $$\mathcal{E}_\sigma(\rho) = \sum_k \mathcal{E}_{\sigma,k}\, \rho\, \mathcal{E}_{\sigma,k}^\dagger, \quad \text{with } \sum_k \mathcal{E}_{\sigma,k}^\dagger \mathcal{E}_{\sigma,k} = I;$$
- After each $\mathcal{E}_\sigma$, a projective measurement is performed with respect to the orthogonal projectors $\{P_{non}, P_{acc}, P_{rej}\}$ onto $\mathcal{H}_{non}$, $\mathcal{H}_{acc}$, and $\mathcal{H}_{rej}$, respectively.

For an input string $x \in \Sigma^*$ (presented as $¢\, x\, \$$), the automaton processes each symbol sequentially. If, at any step, the measurement projects onto $\mathcal{H}_{acc}$ (or $\mathcal{H}_{rej}$), the computation halts with acceptance (or rejection). Otherwise, if the outcome is in $\mathcal{H}_{non}$, the automaton continues processing the next symbol.

Strings Acceptance. The acceptance of an input string x is defined by the cumulative probability that the automaton halts in an accepting configuration. The common acceptance criteria include:

1. **Bounded Error:** There exist $\lambda \in (0, 1]$ and $\epsilon > 0$ such that

$$\begin{aligned} &\text{if } x \in L, \quad \text{cumulative acceptance probability} \geqslant \lambda + \epsilon, \\ &\text{if } x \notin L, \quad \text{cumulative acceptance probability} \leqslant \lambda - \epsilon. \end{aligned}$$

2. **Cutpoint/Exact Acceptance:** As in the MO-1gQFA model, acceptance may also be defined via an isolated cutpoint or by requiring exact acceptance.

Set of Languages Accepted. It has been established that MM-1gQFA, despite the intermediate measurements after each symbol, recognize exactly the class of regular languages (with bounded error). Thus, the frequency of measurements does not extend the language recognition power beyond that of MO-1gQFA [28].

Closure Properties. MM-1gQFA are closed under standard operations. In particular, if L_1 and L_2 are recognized by MM-1gQFA then:

- They are closed under *union* and *intersection* (by appropriate constructions using direct sums or tensor products),
- They are closed under *complementation* (by swapping the roles of $\mathcal{H}_{acc}$ and $\mathcal{H}_{rej}$),
- And they are closed under other operations such as inverse homomorphism.

Summary of Advantages and Limitations. The MM-1gQFA model offers the flexibility of making intermediate measurements, which may simplify the design of some automata. However, like MO-1gQFA, its computational power remains limited to regular languages under the bounded error regime. Furthermore, the state minimization problem for MM-1gQFA is EXPSPACE-hard [32].

Example. For example, consider an MM-1gQFA designed to recognize

$$L = \{w \in \{a, b\}^* \mid \text{the last symbol of } w \text{ is } a\}.$$

In this automaton, after each input symbol the machine performs a measurement. If a measurement outcome projects onto $\mathcal{H}_{acc}$ (indicating that the current configuration is accepting) and no previous measurement forced a rejection, the automaton eventually halts with acceptance. This construction guarantees that the cumulative acceptance probability meets the bounded error condition exactly when the input ends with an a [28].

Additional Topics. Recent research on 1gQFA has addressed the equivalence problem, providing necessary and sufficient conditions based on the linear span of the reachable density operators. Moreover, advanced state minimization techniques have been developed, reducing the minimization problem to solving systems of polynomial inequalities with an EXPSPACE upper bound [33]. Future directions include exploring further generalizations and their potential applications in modeling noisy quantum systems.

3.2.6 Interactive Automata Based on Quantum Interactive Proof Systems

Interactive automata based on quantum interactive proof systems offer a striking demonstration of how even extremely resource-limited verifiers—modeled by quantum finite automata (qfa's)—can, through interaction with a powerful prover, recognize nontrivial languages. Two principal models have been developed in this area:

- **Quantum Interactive Proof (Quantum Interactive Proof (QIP)) systems**, in which the verifier's internal moves remain hidden (private-coin), and
- **Quantum Arthur–Merlin (Quantum Arthur-Merlin (QAM)) systems**, where the verifier publicly announces his next move (public-coin).

In these models, the verifier is typically a two-way qfa, though one-way variants have also been considered. The seminal works by Nishimura and Yamakami [38, 39] and Zheng, Qiu, and Gruska [59] have established detailed protocols and complexity separations that reveal the potential of interactive proofs even when the verifier possesses only finite-dimensional quantum memory.

Formal Definition. A general QIP system with a qfa verifier is defined as a pair (P, V), where:

Verifier. The verifier V is given by

$$V = \big(Q,\, \Sigma \cup \{¢, \$\},\, \Gamma,\, \delta,\, q_0,\, Q_{acc},\, Q_{rej}\big),$$

with the following components:

- Q is a finite set of inner states partitioned as $Q = Q_{non} \cup Q_{acc} \cup Q_{rej}$;
- Σ is the input alphabet, and ¢ and \$ denote the left and right endmarkers, respectively;
- Γ is the communication alphabet;
- δ is the transition function. For each configuration (q, σ, γ), the verifier changes its state, updates the tape head position (with moves in $\{-1, 0, 1\}$), and writes a new symbol in the communication cell according to complex amplitudes given by $\delta(q, \sigma, \gamma, q', \gamma', d)$;
- $q_0 \in Q$ is the initial state;
- Q_{acc} and Q_{rej} are the sets of halting (accepting and rejecting) states.

The verifier's overall Hilbert space, denoted by $\mathcal{H}_V$, is spanned by basis states of the form

$$|q, k, \gamma\rangle, \quad q \in Q,\ k \in \mathbb{Z},\ \gamma \in \Gamma.$$

Prover. The prover P is specified by a family of unitary operators

$$\{U_{x,P,i}\}_{i\geqslant 1},$$

acting on the prover's private Hilbert space $\mathcal{H}_P$. In some variants (denoted by the restriction ⟨c-prover⟩), the prover's unitaries are required to have only 0–1 entries, effectively making the prover deterministic.

QAM Systems. In the QAM variant, the verifier is additionally required to announce his next move via the communication cell, rendering the system a public-coin protocol. Thus, while the basic structure of (P, V) remains the same, a QAM system is denoted as

$$\text{QAM}(\langle\text{restriction}\rangle) \quad \text{or} \quad \text{QIP(public)},$$

and the transition function δ is designed so that, after each move, the pair (q', γ', d) is revealed to the prover.

Strings Acceptance. An interactive proof system (P, V) accepts an input string $x \in \Sigma^*$ if, after a prescribed sequence of interaction rounds, the verifier eventually performs a halting measurement that yields an accepting configuration with high probability. Formally, the system recognizes a language $L \subseteq \Sigma^*$ if the following conditions hold:

- **Completeness:** For every $x \in L$, there exists a prover strategy P such that the verifier accepts x with probability at least $1 - \epsilon$, where $\epsilon < 1/2$.
- **Soundness:** For every $x \notin L$, for every prover strategy P^*, the verifier rejects x with probability at least $1 - \epsilon$.

Variants of acceptance include definitions via an isolated cutpoint or exact acceptance (i.e., acceptance with probability 1), but the bounded-error model is standard.

Set of Languages Accepted. The language recognition power of these interactive systems depends on the verifier's model and the nature of the interaction:

- When the verifier is a **1qfa** (one-way qfa), it has been shown that

  $$\text{QIP(1qfa)} = \text{REG},$$

  meaning that even with interaction the system recognizes only the regular languages [38].
- In contrast, when the verifier is a **2qfa** (two-way qfa), the interactive proof system can recognize languages that are not regular. For example, several protocols with 2qfa verifiers operating in expected polynomial time have been shown to outperform classical AM systems with 2pfa verifiers [59, 39].
- In the public-coin (QAM) variant, where the verifier reveals its next move, the additional information sometimes further enhances the system's power, and comparisons with classical Arthur–Merlin systems have been established.

Closure Properties. The language classes defined by QIP and QAM systems exhibit robust closure properties:

- They are closed under *union* and *intersection*, typically via parallel composition (using direct sums or tensor products).
- They are closed under *complementation* (by exchanging the roles of Q_{acc} and Q_{rej} in the verifier's design).
- They are also closed under other operations such as inverse homomorphism.

These properties are established through constructions that combine multiple protocols while preserving the bounded-error guarantees.

Summary of Advantages and Limitations. Interactive proof systems with qfa verifiers offer several compelling advantages:

- **Finite Quantum Resources:** The verifier operates with a finite-dimensional quantum system, making the model realistic for devices with limited quantum memory.
- **Enhanced Recognition via Interaction:** Even though a standalone qfa (especially a 1qfa) may recognize only regular languages, interaction with a powerful prover can significantly boost the verifier's ability, particularly when using two-way qfa verifiers.
- **Flexibility through Protocol Variants:** By varying whether the system is a QIP (private-coin) or QAM (public-coin) system, and by imposing restrictions on the prover (quantum vs. classical), one can fine-tune the computational power and compare with classical interactive proof systems.

However, there are also limitations:

- **Limited Power of One-Way Verifiers:** When restricted to one-way qfa verifiers, the system's power is confined to the regular languages.
- **Potentially High Interaction Complexity:** Protocols with two-way qfa verifiers can require a large (sometimes exponential) number of rounds or running time.
- **Technical Complexity:** The design and analysis of these interactive protocols are intricate, involving careful balancing of quantum and classical information.

Example. An illustrative example is the QIP protocol for the language

$$\texttt{Pal\#} = \{x\#x^R \mid x \in \{0,1\}^*\},$$

which comprises even-length palindromes separated by a delimiter. In the protocol described in [39], the verifier (modeled as a 2qfa) interacts with a quantum prover in the following way:

1. The verifier scans the input (framed by the endmarkers ¢ and $) and, based on its transition function δ, generates a superposition reflecting potential midpoints.
2. Through a sequence of rounds, the verifier requests the prover to indicate the position of the center. In the QAM variant, the verifier publicly announces his next move to assist the prover.
3. Finally, the verifier applies a Quantum Fourier Transform (QFT) to consolidate the information and performs a measurement. If the input is indeed of the form $x\#x^R$, the verifier accepts with high probability; otherwise, it rejects.

This example clearly demonstrates how interaction compensates for the verifier's limited memory, enabling recognition of a nontrivial language.

Additional Topics. Several open problems and future research directions emerge from this line of work:

- **Round Complexity:** How does limiting the number of interaction rounds (e.g., as in QIP$\#(k)$) affect the recognition power and efficiency?
- **Prover Restrictions:** What are the precise differences in computational power when the prover is restricted to classical behavior ($\langle$c-prover$\rangle$) versus full quantum capability?
- **Public vs. Private Protocols:** Further analysis is needed to understand the trade-offs between QIP (private-coin) and QAM (public-coin) systems.
- **Resource-Bounded Protocols:** Tightening the upper and lower bounds on running time and state complexity for these systems remains a challenging task.

These issues continue to be central to the ongoing exploration of the interplay between interaction and quantum finite automata.

- **Limited-Round Interactive Systems (QIP$\#(k)$):** In some works (e.g., by Nishimura and Yamakami), the number of interaction rounds is explicitly bounded. These models, often denoted by QIP$\#(k)$ (with k indicating the maximum number of rounds), allow a more refined complexity classification of interactive protocols.
- **Interactive Proof Systems with Semi-Quantum Verifiers:** Another significant model is the one in which the verifier is not a full-fledged quantum finite automaton but a *semi-quantum* two-way finite automaton (2QCFA). In such systems—as studied, for instance, by Zheng, Qiu, and Gruska—the verifier possesses both classical and quantum states, using limited quantum resources alongside classical processing. These systems (sometimes denoted QAM(2QCFA) in the public-coin setting) have been shown to recognize languages beyond those recognizable by two-way probabilistic finite automata.
- **Variants Based on Prover Restrictions:** Some works also examine the effect of restricting the prover to *classical* behavior (i.e., using only 0–1 unitary operators, sometimes denoted by the restriction $\langle$c-prover$\rangle$). This yields interactive models that can be compared with their fully quantum counterparts.

3.2.7 Multi-letter Models

Multiletter Quantum Finite Automata (Multi-Letter Quantum Finite Automaton (ML-QFA)) are a generalization of traditional quantum finite automata, where transitions depend on multiple letters read from the input, rather than a single letter. This allows them to capture more complex patterns in the input string.

Multi-Letter Quantum Finite Automaton (ML-QFA)

Multiletter QFA were introduced to extend the capability of classical and quantum models by applying unitary operations based on the last k letters read rather than just one. This enables them to recognize languages outside the reach of traditional measure-once or measure-many 1QFA models [6].

Formal Definition A k-letter ML-QFA is a 5-tuple $A = (Q, Q_{acc}, |\psi_0\rangle, \Sigma, \mu')$, where:

- Q is a finite set of states,
- $Q_{acc} \subseteq Q$ is the set of accepting states,
- $|\psi_0\rangle$ is the initial superposition of states with unit norm,
- Σ is the input alphabet,
- $\mu' : (\Sigma \cup \{\Lambda\})^k \to U(\mathbb{C}^n)$ is a function assigning a unitary operator to every k-tuple of symbols.

The transition applies the unitary associated with the last k letters read. For input $\omega = x_1 x_2 \dots x_n$, the computation evolves through unitary applications as specified in Eq. (1) and acceptance is determined using a projection operator P_{acc} [45].

Strings Acceptance Acceptance is defined by exact probability, cutpoint (strict or non-strict), or bounded-error depending on how $P_A(\omega) = \|P_{acc} U_\omega |\psi_0\rangle\|^2$ compares to a threshold λ.

Sets of Languages Accepted ML-QFA can recognize a proper superset of regular languages compared to MO-1QFA and MM-1QFA. Notably, they can recognize the language $(a + b)^* a$ which is not recognizable by MO-1QFA or MM-1QFA [6].

Closure Properties The class of languages recognized by ML-QFA is not closed under union, intersection, or complement, especially under non-strict cutpoint semantics [45].

Advantages and Limitations ML-QFA demonstrate higher computational power with fewer states in certain scenarios. However, equivalence and minimization are complex and computationally hard. For non-strict cutpoints, the emptiness problem is undecidable [46].

Comparison ML-QFA are more expressive than MO-1QFA and MM-1QFA under the same acceptance criteria, but less so than two-way or general quantum automata with additional memory models [47].

Example The language $(a + b)^* a$ is a canonical example recognized by 2-letter ML-QFA, using a transition function dependent on the last two characters read [6].

Additional Topics Research is ongoing in determining exact hierarchies, equivalence testing, and applying ML-QFA in quantum protocol verification [29, 46].

Multiletter Measure-Many Quantum Finite Automata (ML-MMQFA)

Introduction Multi-Letter Measure Many Quantum Finite Automaton (ML-MMQFA) combine the power of measure-many acceptance strategies with multiletter transitions, extending both classical MM-1QFA and ML-QFA. In this model, a measurement is performed after each quantum evolution step, but the evolution itself depends on the last k letters read. This hybrid structure allows ML-MMQFA to accept more complex languages than ML-QFA or MM-1QFA alone [29].

Formal Definition A k-letter ML-MMQFA is a 7-tuple $A = (Q, Q_{acc}, Q_{rej}, |\psi_0\rangle, \Sigma, \mu', \mathcal{O})$, where:

- Q is the finite set of states,
- $Q_{acc} \subset Q$ is the set of accepting states,
- $Q_{rej} \subset Q$ is the set of rejecting states with $Q_{acc} \cap Q_{rej} = \varnothing$,
- $|\psi_0\rangle$ is the initial state,
- Σ is the finite input alphabet,
- $\mu' : (\Sigma \cup \{\Lambda, £, \$\})^k \to U(\mathbb{C}^n)$ assigns a unitary matrix to each k-letter word,
- $\mathcal{O} = \{P_{acc}, P_{rej}, P_{non}\}$ is a projective measurement partitioning the Hilbert space based on Q_{acc}, Q_{rej}, and the non-halting subspace.

Computation starts with the end-marked input $£x_1x_2 \ldots x_n\$$, and proceeds by interleaving unitary evolutions with projective measurements.

Strings Acceptance For a word $\omega = x_1x_2 \ldots x_n$, the acceptance probability is:

$$P_A(\omega) = \sum_{i=0}^{n+1} \left\| P_{acc} U_{x_i} \left(\prod_{j=i-1}^{0} P_{non} U_{x_j} \right) |\psi_0\rangle \right\|^2$$

A string is accepted if this probability exceeds a cutpoint (for probabilistic acceptance), or is exactly 1 (for exact acceptance). Both strict and non-strict cutpoints are considered [29].

Sets of Languages Accepted ML-MMQFA can recognize languages beyond the regular class. They accept some languages that are not accepted by classical QFA or even standard MM-1QFA, especially under non-strict cutpoint semantics [45, 29].

Closure Properties The set of languages recognized by ML-MMQFA is not closed under union, intersection, or complementation for general acceptance modes. Notably, these properties depend heavily on the acceptance criteria used (bounded-error, cutpoint, etc.) [45].

Advantages and Limitations ML-MMQFA are more powerful than both ML-QFA and MM-1QFA. However, they inherit the undecidability of the emptiness and equivalence problems for non-strict and strict cutpoint semantics [46, 29]. Their expressive power comes at the cost of analytical and implementation complexity.

Comparison ML-MMQFA strictly subsume ML-QFA under the same input size and acceptance criteria. They are not comparable in power with general QFA models that allow additional memory or two-way movement. Compared to MM-1QFA, ML-MMQFA can accept non-stochastic languages [45].

Example An ML-MMQFA with 2-letter transitions and intermediate measurements can accept the language $(a + b)^*a$ with higher robustness to probabilistic acceptance thresholds than an ML-QFA [6].

Additional Topics Further work focuses on state complexity, succinctness, and simulation algorithms between different classes. The diagonal sum construction plays a key role in analyzing equivalence and decidability [29].

Multi-Letter Reversible Quantum Finite Automaton (ML-RevQFA)

Introduction Multi-Letter Reversible Quantum Finite Automaton (ML-RevQFA) extend the multiletter framework by imposing reversibility constraints on the quantum evolution. These automata are designed so that each computation step is invertible, maintaining the core principle of reversibility from quantum mechanics and enhancing coherence preservation [6].

Formal Definition A k-letter ML-RevQFA is a special case of k-letter ML-QFA where each unitary transformation $\mu'(\omega)$ satisfies the additional constraint that $\mu'(\omega)^{-1} = \mu'(\omega)^\dagger$ for all $\omega \in (\Sigma \cup \{\Lambda\})^k$, and the set of such transformations forms a group under composition.

The automaton is defined as a 5-tuple $A = (Q, Q_{acc}, |\psi_0\rangle, \Sigma, \mu')$, with μ' restricted to reversible unitary matrices.

Strings Acceptance Acceptance is defined as in ML-QFA using a projector P_{acc}. For an input string ω, the acceptance probability is:

$$P_A(\omega) = \|P_{acc} U_\omega |\psi_0\rangle\|^2$$

with U_ω computed from the sequence of reversible unitaries associated with the k-letter substrings [6].

Sets of Languages Accepted ML-revQFA are strictly more limited than general ML-QFA due to their reversibility constraint. They can recognize a subset of regular languages and do not accept all regular languages with bounded error, particularly under exact acceptance criteria.

Closure Properties The class of languages recognized by ML-revQFA is not closed under union or complement. Reversibility limits the computational power of these models in comparison with more general multiletter QFA models [6].

Advantages and Limitations Reversible automata are appealing for quantum computing implementations due to better coherence and energy efficiency. However, their expressiveness is constrained. ML-revQFA cannot recognize some simple regular languages that non-reversible ML-QFA can handle [6].

Comparison ML-revQFA are less powerful than both ML-QFA and ML-MMQFA. They are closely related to reversible classical automata and group automata. Compared to non-reversible models, they generally require more states or cannot recognize the same languages under equivalent semantics [6].

Example It was shown that ML-revQFA cannot accept the language $(a + b)^*a$ even when using multiletter transitions, while a general ML-QFA can accept it with bounded error [6].

Additional Topics Future research may explore connections with quantum error correction, fault-tolerant reversible computing, and applications in energy-efficient quantum hardware design.

3.3 Other Models of Quantum Finite Automata

Beyond the core models of quantum finite automata discussed in the previous sections, the literature also presents several alternative models that explore different computational paradigms, theoretical extensions, or enhancements. While these models are less prominent or less widely used, they offer valuable insights into the boundaries and variations of quantum automata theory.

In this section, we provide a concise overview of some notable variants. Each model is briefly introduced with its main characteristics and distinguishing features, along with references to the original works in which they were proposed. Readers interested in further details are encouraged to consult the cited articles.

3.3.1 Quantum Turing Machines

The Quantum Turing Machine (QTM) is the quantum analog of a classical Turing machine, featuring an infinite tape and a moving head with quantum states and unitary transitions. It was first proposed by Deutsch in 1985 as a general model of quantum computation [21]. A QTM can implement any quantum algorithm and is computationally equivalent to the quantum circuit model (Yao proved that any QTM can be efficiently simulated by quantum circuits and vice versa [57]). Unlike finite automata models, the QTM is not limited to regular languages – it has unbounded memory and can recognize non-regular languages – but this generality comes at the cost of a much more complex machine description. In practice, QTMs serve mostly as a theoretical cornerstone since simpler models (like quantum circuits) are used for designing algorithms, yet the QTM remains important for defining quantum complexity classes and formalizing the Church–Turing principle in the quantum realm.

3.3.2 Latvian Quantum Finite Automata

The term Latvian Quantum Finite Automaton (LQFA) refers to the one-way quantum finite automaton model introduced by Ambainis and Freivalds (who are Latvian) in 1998 [3]. This model is essentially the *measure-once* 1QFA: the machine's state evolves unitarily as it reads the input, and only after reaching the end of the input is a single projective measurement performed to decide acceptance. (In contrast, the earlier QFA model by Kondacs and Watrous allowed measurements after each step.) The Latvian 1QFA demonstrated that even with a single end-of-input measurement, a quantum automaton can recognize certain regular languages with exponentially fewer states than any equivalent deterministic automaton. However, like other 1QFAs, it cannot recognize all regular languages. The LQFA is historically significant as one of the first quantum automata models, and its state-efficiency advantages and limitations were studied in subsequent works.

3.3.3 l-valued Finite Automata

An l-valued Finite Automaton (l-VFA) is an automaton model based on multi-valued logic (in particular, on quantum logic), rather than probabilistic or binary state transitions. This model was explored by Ying (2000) and was later formalized and extended by Qiu in 2007 as a "logical" approach to quantum computation [42]. In an l-VFA, the transition function is not strictly deterministic or probabilistic – instead, each transition from a state p to a state q on an input symbol σ is assigned a truth-value from a complete orthomodular lattice L. Intuitively, $\delta(p, \sigma, q)$ may be 0, 1, or some intermediate truth-value in L. A string is accepted by an l-VFA if the aggregated truth-value of all paths leading to an accepting state evaluates to 1 in the lattice sense. This construction generalizes classical finite automata and provides a way to apply quantum logic to automata theory.

3.3.4 l-valued Pushdown Automata

The l-valued Pushdown Automaton (l-VPDA) extends the idea of an l-VFA by adding a pushdown stack, thus enabling recognition of some non-regular languages within the l-valued logic framework. This model was introduced alongside l-VFAs by Qiu in 2007 [42] as part of the effort to build automata theory on quantum logic. An l-VPDA operates similarly to a classical pushdown automaton, but its state transitions and stack operations carry truth-values in a lattice L instead of deterministic outcomes.

3.3.5 Quantum Automata with Advice

Quantum Finite Automaton with Advice are variants of 1QFA that are supplemented with an additional input - an advice string or quantum state - that depends only on the input length n and is provided to the automaton to improve its computation. This idea was studied by Yamakami (2014) [56]. In his model, the machine can utilize a pre-prepared quantum advice state during its computation, allowing for potentially improved computational power while still remaining weaker than full quantum Turing machines.

3.3.6 Enhanced Quantum Finite Automata

Enhanced One-Way Quantum Finite Automaton (E-1QFA) is a variant of the one-way QFA where the machine's state can be measured after each symbol is read, rather than restricting measurement to occur only at the end of the input. This model was introduced by Nayak [37] and studied further by Lin [30]. It allows the computation to dynamically adapt based on partial measurement outcomes, making it slightly more powerful than traditional one-way QFAs in certain contexts.

3.3.7 Postselection Quantum Finite Automata

Postselection Quantum Finite Automaton (PQFA) is a theoretical model that augments a quantum finite automaton with the power of *postselection* - the ability to conditionally proceed based on a desired measurement outcome. This powerful but unphysical feature was used to explore computational limits, and the model was studied in depth by Scegulnaja-Dubrovska et al. [52] and originally proposed in the context of quantum complexity by Aaronson [1].

3.3.8 ω Quantum Finite Automata

ω-Quantum Finite Automaton (ω-QFA) extend quantum finite automata to operate on infinite input strings. Bhatia and Kumar (2019) introduced several formal models with different acceptance conditions like Büchi, Rabin, and Streett [10]. These models are important for exploring quantum computation over streams or continuous inputs and show intriguing differences from their classical counterparts.

3.3.9 Promise Problems and Quantum Finite Automata

Promise problems are a generalization of language recognition where an automaton is required to correctly classify inputs from two disjoint sets: the “yes” instances and the “no” instances. This relaxed setting provides a useful framework for analyzing subtle distinctions in computational power, especially when comparing classical and quantum models.

QFA have demonstrated significant advantages in the context of promise problems. These models are often more state-efficient or capable of solving problems that classical automata cannot handle with bounded error. One notable study by Zheng et al. [61] investigates the 2QCFA model and demonstrates its exponential state succinctness over classical counterparts for families of promise problems. For example, they construct a 2QCFA that solves a problem with constant quantum memory and logarithmic classical memory, whereas equivalent classical automata require exponentially more states.

Other works explore theoretical implications of quantum advantages under promises. Rashid and Yakaryilmaz [49] analyze how quantum automata solving promise problems can relate to foundational concepts like contextuality in quantum theory. Bianchi et al. [11] examine the computational complexity of promise problems across classical and quantum finite automata, identifying specific contexts where quantum models are strictly more efficient. Gruska et al. [24] further study promise problems under exact acceptance and show that QFA can solve certain structured promise problems with significantly fewer states than their classical counterparts.

Overall, the study of promise problems has emerged as a rich area to highlight the computational advantages of quantum models, often revealing separations that are not observable in standard language recognition settings.

4. Automata to Circuits

QFAs furnish a concise, mathematically transparent model of finite-memory computation, yet practical algorithms must ultimately be recast as quantum circuits that manipulate qubits through finite sequences of gates and measurements. The purpose of this chapter is to articulate, in a systematic manner, how a quantum automaton defined at the symbolic level is translated into a concrete circuit description suitable for compilation on Noisy Intermediate-Scale Quantums (NISQs) hardware. By mapping each automaton primitive onto circuit counterparts we obtain designs that are executable on present devices, support quantitative resource accounting and admit gate-level formal verification.

Section 4.1 outlines the compilation workflow for the MO-1QFA, illustrating how its fundamental components can be encoded within a quantum circuit model. The translation process preserves the computational semantics of the original automaton while making it compatible with standard circuit synthesis techniques.

Section 4.2 extends the methodology to the MM-1QFA, whose intermediate measurements create early-halt branches and classical control flow. Particular attention is devoted to expressing the three-outcome measurement paradigm with standard two-outcome projective tests, to limiting ancilla overhead when discarding rejected branches, and to maintaining language-recognition semantics in the presence of realistic noise.

A template-first compilation philosophy is retained throughout: for an input word of length L the compiler emits a parameterised skeleton in which each placeholder gate is later instantiated with the concrete operator U_{σ_i} attached to the i-th symbol. This separation between structural aspects fixed by the automaton and numerical parameters dictated by the input encourages component reuse across multiple words and eases the deployment of QFAs as high-speed recognisers within larger quantum applications. Two complementary instantiation strategies are considered in Section 4.3. The first, an offline synthesis approach, compiles every operator U_σ ahead of execution and stores the resulting gate sequences as reusable fragments. The second adopts a parameter-loading paradigm in which a generic template containing analytic Euler-angle rotations is populated at runtime with classically computed angles that depend on the input word, thereby reducing memory overhead and enabling just-in-time adaptation to specific problem instances.

Upon completing the chapter the reader will possess a reproducible method for converting any MO-1QFA or MM-1QFA into an architecture-independent gate-level description, together with practical criteria for choosing state encodings, measurement decompositions and synthesis back-ends. These results pave the way for future research on two-way and hybrid models and represent a decisive step toward a unified tool-chain for automata-driven quantum software engineering.

4.1 Measure-Once One-Way Quantum Finite Automaton to Circuit

This section presents the compilation of the MO-1QFA model into an executable quantum circuit, elucidating the precise correspondence between automaton-level abstractions and gate-level constructs. The internal state set Q is represented using $\lceil \log_2 |Q| \rceil$ qubits, encoding each classical state $q \in Q$ into a computational basis vector of the quantum register. Each symbol $\sigma \in \Sigma$ is associated with a unitary matrix U_σ, which governs the evolution of the state vector upon reading σ; these operators are later decomposed into sequences of elementary gates drawn from a universal set (e.g., Clifford+T or $\{\mathrm{CNOT}, R_z, H\}$). The initial state preparation maps the all-zero register to the automaton's start state via minimal gate operations. Finally, the accepting condition is enforced via a projective measurement onto a subspace defined by the set of accepting states F, implemented as a single multi-controlled rotation followed by a standard measurement. This mapping supports systematic synthesis of circuits from automaton descriptions while preserving the semantics of quantum language recognition.

4.1.1 Mapping Automaton Components to Circuit Elements

A MO-1QFA is defined as a tuple

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

as introduced in Section 3.1.1. The goal is to construct, for any such automaton, a quantum circuit that faithfully reproduces its evolution and acceptance behaviour. This is achieved by mapping each formal component to a corresponding physical construct within the circuit model, preserving the semantics of quantum language recognition.

State Register and Qubit Allocation

The set of internal states Q is encoded over an n-qubit register, where $n = \lceil \log_2 |Q| \rceil$. Each state $q \in Q$ corresponds to a computational basis vector $|q\rangle \in (\mathbb{C}^2)^{\otimes n}$ under a fixed encoding. This representation ensures compatibility with standard gate decompositions and measurement procedures, and facilitates reversible indexing of automaton transitions.

Symbol-Dependent Unitary Evolution

Each symbol $\sigma \in \Sigma$ induces a unitary transformation U_σ defined by the transition function δ. These matrices are assumed to be unitary by definition of the model, and are compiled into native gate sequences using a fault-tolerant universal basis, such as Clifford+T or $\{\mathsf{CNOT}, R_z, H\}$. This decomposition is performed either ahead of time or via parameterised template instantiation, depending on the compilation strategy adopted.

Initialisation Procedure

The computation starts in the automaton's designated initial state q_0, represented as $|q_0\rangle$ on the n-qubit register. Physical initialisation begins with the zero state $|0\rangle^{\otimes n}$, which is then mapped to $|q_0\rangle$ using a preparation circuit. When q_0 is a basis vector in the encoding, this step reduces to applying a sequence of Pauli-X gates.

Measurement and Acceptance

Upon completion of the input traversal, the quantum state is measured against the accepting subspace defined by the set $F \subseteq Q$. The corresponding projector is

$$P_{\text{acc}} = \sum_{q \in F} |q\rangle\langle q| \,,$$

and the final two-outcome measurement $\{P_{\text{acc}}, I - P_{\text{acc}}\}$ determines acceptance (output 1) or rejection (output 0). In circuit terms, this is realised via a controlled operation targeting an ancilla qubit, followed by a standard measurement. Efficient implementations leverage multi-controlled rotations and ancilla reuse to minimise overhead.

Automaton part	**Circuit realisation**	**Explanation**
Q	n-qubit basis	Encode each $q \in Q$ as $\|q\rangle$, with $n = \lceil \log_2 \|Q\| \rceil$.
Σ	unitary U_σ	Reading σ applies U_σ.
δ	set $\{U_\sigma\}$	Transition matrices later decomposed into elementary gates.
q_0	state $\|q_0\rangle$	Prepare register from $\|0\rangle^{\otimes n}$ to $\|q_0\rangle$.
F	projector P_{acc}	Measure $\{P_{\text{acc}}, I - P_{\text{acc}}\}$ for accept/reject.

Table 4.1: Mapping MO-1QFA components to quantum-circuit constructs.

4.1.2 General Compilation Algorithm

The compilation of a MO-1QFA into an executable quantum circuit is divided into two conceptually distinct steps:

1. **Template Generation:** Construct a symbolic circuit template that encodes the automaton's structure using placeholder gates. This template depends only on the automaton and the input word length.

2. **Instantiation:** For a specific input word, substitute each symbolic placeholder with the actual unitary operator defined by the automaton and decompose it into elementary gates.

Algorithm 1 Template Generation for a MO-1QFA Circuit

Require: Automaton $A = (Q, \Sigma, \delta, q_0, F)$, input length L
Ensure: Parametric circuit template with symbolic placeholders
1: $n \leftarrow \lceil \log_2 |Q| \rceil$
2: Initialise n qubits in state $|q_0\rangle$
3: **for** $i = 1$ **to** L **do**
4: Insert symbolic gate $\boxed{U_{x_i}}$
5: **end for**
6: Append projective measurement $\{P_{\text{acc}}, I - P_{\text{acc}}\}$

Algorithm 1 creates a circuit that is independent of the actual input string. Each gate $\boxed{U_{x_i}}$ is a placeholder symbolically representing the unitary matrix to be applied upon reading the i-th symbol. This structure is fully determined by the automaton and remains fixed across all input words of the same length.

To execute the template on a specific word $x = x_1 x_2 \dots x_L \in \Sigma^L$, the placeholders $\boxed{U_{x_i}}$ must be instantiated as concrete gates derived from the automaton's transition matrices.

Algorithm 2 Instantiation and Execution of a Compiled MO-1QFA Circuit

---

Require: Input $x = x_1 x_2 \dots x_L$, circuit template, gate library or decomposition scheme for each U_σ

Ensure: Acceptance probability $p_A(x)$

1: **for** $i = 1$ **to** L **do**

2: Replace $\boxed{U_{x_i}}$ with a gate decomposition implementing U_{x_i}

3: **end for**

4: Apply $U_{x_L} \cdots U_{x_1}$ to $|q_0\rangle$

5: Perform measurement $\{P_{\text{acc}}, I - P_{\text{acc}}\}$

6: **return** $p_A(x) = \|P_{\text{acc}} U_{x_L} \cdots U_{x_1} |q_0\rangle\|^2$

The two-phase compilation strategy promotes modularity: the symbolic template can be reused across many inputs, while the instantiation adapts to specific data. This design aligns with scalable quantum software practices and is compatible with both pre-synthesised libraries and dynamic, runtime parameter loading (see Section 4.3).

4.1.3 Step-by-Step Examples

To concretise the abstract compilation scheme described above, we now present explicit examples illustrating the end-to-end translation of MO-1QFA instances into executable quantum circuits. Each example begins with a formal automaton specification and proceeds through template generation, gate instantiation for a specific input word, and—when appropriate—optimised decomposition into native gates. These case studies demonstrate the generality of the approach and clarify how structural automaton features influence circuit depth, gate choice, and measurement configuration.

Single-Letter Alphabet. Consider the MO-1QFA defined by

$$Q = \{q_0, q_1\}, \quad \Sigma = \{a\}, \quad q_0 \text{ initial}, \quad F = \{q_1\},$$

and let the transition unitary associated with the only input symbol be the Hadamard gate

$$U_a = H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

We compile this automaton for words of length $L = 1$ using Algorithm 1. The process proceeds as follows:

1. **Qubit allocation.** Since $|Q| = 2$, we require $n = \lceil \log_2 2 \rceil = 1$ qubit. The state set is encoded as $|q_0\rangle = |0\rangle$ and $|q_1\rangle = |1\rangle$, so the circuit will consist of a single wire.

2. **Initialisation.** The register is initialised in state $|0\rangle = |q_0\rangle$, which matches the default zero state on most quantum backends.

3. **Unitary evolution (template).** The compiler inserts one symbolic placeholder $\boxed{U_{x_1}}$ representing the unitary corresponding to the input symbol at position 1. The result is the circuit shown in subfigure 4.1a.

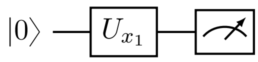
4. **Instantiation.** For the specific input $x = a$, we substitute $U_{x_1} = H$, yielding the circuit in subfigure 4.1b.

5. **Measurement.** The accepting projector is $P_{\text{acc}} = |1\rangle\langle 1|$. A measurement in the computational basis is applied at the output. The final synthesised circuit—requiring no decomposition since H is native—is shown in subfigure 4.1c. The output state is

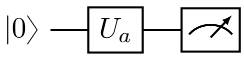
$$H|0\rangle = \tfrac{1}{\sqrt{2}}(|0\rangle + |1\rangle),$$

so the acceptance probability is

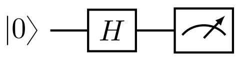
$$p_M(a) = \|P_{\text{acc}} H |0\rangle\|^2 = \left|\tfrac{1}{\sqrt{2}}\right|^2 = \tfrac{1}{2}.$$



(a) Step 1: Template circuit



(b) Step 2: Instantiation



(c) Step 2: Gate-level circuit

Figure 4.1: Compilation stages for Example 4.1.3. The first row shows the symbolic template; the second row shows instantiation and final gate decomposition.

Two-Symbol Word of Length $L = 2$. Consider a MO-1QFA defined by:

$$Q = \{q_0, q_1, q_2\}, \quad \Sigma = \{a, b\}, \quad q_0 \text{ initial}, \quad F = \{q_2\},$$

with transitions specified as follows:

- U_a performs a rotation between $|q_0\rangle$ and $|q_1\rangle$,
- U_b swaps $|q_1\rangle$ and $|q_2\rangle$.

We compile the automaton on the input word $x = ab$, of length $L = 2$.

1. **Qubit allocation.** Since $|Q| = 3$, we require

$$n = \lceil \log_2 3 \rceil = 2 \text{ qubits}.$$

The states q_0, q_1, and q_2 are encoded as $|00\rangle$, $|01\rangle$, and $|10\rangle$ respectively.

2. **Initialisation.** The register is prepared in state $|q_0\rangle = |00\rangle$. This is typically the default zero state and requires no preparation gates.

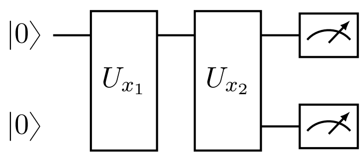
3. **Unitary evolution (template).** For a word of length $L = 2$, the compiler inserts two placeholders $\boxed{U_{x_1}}$ and $\boxed{U_{x_2}}$, representing the unitaries to be applied at each step. The resulting template is shown in subfigure 4.2a.
4. **Instantiation.** For the specific input $x = ab$, we substitute $U_{x_1} = U_a$ and $U_{x_2} = U_b$, yielding the circuit in subfigure 4.2b.
5. **Measurement.** The accepting subspace corresponds to $F = \{q_2\}$, i.e., the basis state $|10\rangle$. The measurement is performed in the computational basis, and the projector is

$$P_{\text{acc}} = |10\rangle\langle 10| \,.$$

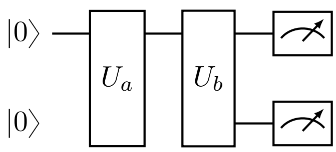
The final synthesised circuit, shown in subfigure 4.2c, assumes a possible decomposition where U_a is implemented as a rotation $\mathcal{R}_X(\theta)$ acting on the subspace $\text{span}\{|00\rangle, |01\rangle\}$, and U_b is realised by a SWAP gate between $|01\rangle$ and $|10\rangle$.

The acceptance probability depends on the choice of rotation angle θ. For instance, if $\theta = \pi/2$, the sequence $U_b U_a |00\rangle$ results in state $|10\rangle$ with probability 1, yielding

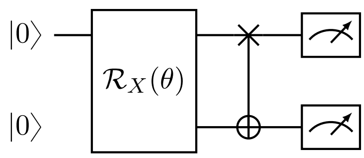
$$p_M(ab) = \|P_{\text{acc}} U_b U_a |00\rangle\|^2 = 1.$$



(a) Step 1: Template circuit



(b) Step 2: After instantiation



(c) Step 2: Gate-level circuit

Figure 4.2: Compilation stages for Example 4.1.3. The first row shows the symbolic template; the second row shows the instantiated and decomposed circuit for input *ab*.

Example 3: Cyclic Automaton, $L = 3$. Consider a MO-1QFA with

$$Q = \{q_0, q_1, q_2\}, \quad \Sigma = \{a\}, \quad q_0 \text{ initial}, \quad F = \{q_0\}.$$

Let the unitary U_a implement a 3-cycle over the basis states:

$$U_a |q_i\rangle = |q_{i+1 \bmod 3}\rangle \,, \quad \text{for } i \in \{0, 1, 2\}.$$

We compile the automaton on the input word $x = aaa$, of length $L = 3$.

1. **Qubit allocation.** The state space has cardinality $|Q| = 3$, so we require

$$n = \lceil \log_2 3 \rceil = 2 \text{ qubits}.$$

The states q_0, q_1, and q_2 can be encoded as basis vectors $|00\rangle$, $|01\rangle$, and $|10\rangle$ respectively.

2. **Initialisation.** The register is initialised in state $|q_0\rangle = |00\rangle$.

3. **Unitary evolution (template).** The compiler inserts three placeholders, one for each input symbol. These gates are labelled $\boxed{U_{x_1}}$, $\boxed{U_{x_2}}$, $\boxed{U_{x_3}}$ and will later be replaced with actual unitaries. The resulting circuit structure is shown in subfigure 4.3a.

4. **Instantiation.** For the input $x = aaa$, all placeholders are replaced by U_a, as every symbol is a. The resulting sequence $U_aU_aU_a$ is shown in subfigure 4.3b.

5. **Cycle recognition and optimisation.** Since U_a implements a perfect 3-cycle, we have

$$U_a^3 = \mathbb{I}.$$

   The composition of three applications of U_a returns the system to its original state. Recognising this cyclic structure, the compiler simplifies the circuit by removing all three gates, as shown in subfigure 4.3c. The net result is that the final state equals the initial state, i.e.,

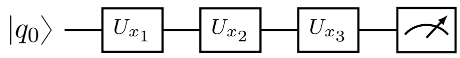
$$|\Psi_{aaa}\rangle = |q_0\rangle,$$

   which lies in the accepting subspace defined by $F = \{q_0\}$. Thus, the input aaa is accepted with probability

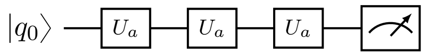
$$p_M(aaa) = \| P_{\text{acc}} |q_0\rangle \|^2 = 1.$$

In the context of quantum circuit compilation, automata that exhibit cyclic behaviour—such as k-cycles over the state set—allow for a key optimisation: repeated applications of a unitary operator implementing the cycle can often be collapsed. Specifically, if a unitary U satisfies $U^k = \mathbb{I}$, then any sequence of k consecutive applications yields the identity transformation. In such cases, the compiler can detect the cyclic structure statically and eliminate the redundant operations from the circuit. This not only reduces the gate count but also preserves the automaton's transition semantics exactly. Such cycle-aware optimisation plays a crucial role in minimising depth and improving interpretability of automaton-derived circuits.

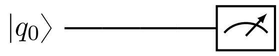
When the number of repetitions is not a multiple of the cycle length, or when intermediate cyclic states influence acceptance, the compiler cannot eliminate the corresponding gates. In such cases, the unitary operator implementing the cycle must be applied explicitly at each relevant input position. These cycle operations are realised as permutations over the encoded state space, typically constructed from SWAP gates or controlled Pauli operations. Although they may introduce repetition, this explicit representation ensures that the circuit precisely mirrors the automaton's behaviour, including partial traversals of cyclic transitions.



(a) Step 1: Template circuit



(b) Step 2: Instantiation



(c) Step 2: Optimised circuit

Figure 4.3: Compilation stages for Example 4.1.3. The first row shows the symbolic template; the second row illustrates the instantiation and the final optimised circuit after cycle detection.

Partial Cycle Without Optimisation. We now consider a variation of Example 4.1.3 in which the cycle is only partially traversed. Let the automaton be defined as

$$Q = \{q_0, q_1, q_2\}, \quad \Sigma = \{a\}, \quad q_0 \text{ initial}, \quad F = \{q_1\},$$

and let U_a be the same 3-cycle unitary:

$$U_a |q_i\rangle = |q_{(i+1) \bmod 3}\rangle, \quad \text{for } i \in \{0, 1, 2\}.$$

The input word is now $x = aa$, i.e., only two applications of U_a.

1. **Qubit allocation.** As before, we require

   $$n = \lceil \log_2 3 \rceil = 2 \text{ qubits},$$

   with states encoded as $|00\rangle$, $|01\rangle$, and $|10\rangle$.

2. **Initialisation.** The register is prepared in $|00\rangle$, representing q_0.

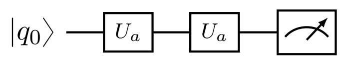
3. **Unitary evolution.** The compiler inserts two unitaries: $U_{x_1} = U_{x_2} = U_a$, as shown in subfigure 4.4a. Since the number of applications is less than the cycle length, the system does not return to the initial state. Therefore, the cycle identity $U_a^3 = \mathbb{I}$ does not apply here, and no optimisation is possible. The complete sequence must be preserved.

4. **Measurement.** After applying U_a twice, the final state is

   $$|\Psi_{aa}\rangle = U_a^2 |q_0\rangle = |q_2\rangle .$$

   The acceptance condition is $F = \{q_1\}$, corresponding to $|01\rangle$. Since the final state $|10\rangle$ is orthogonal to the accepting subspace, the acceptance probability is

   $$p_M(aa) = \|P_{\text{acc}} |\Psi_{aa}\rangle\|^2 = 0.$$



(a) Circuit for $x = aa$ (no optimisation)

Figure 4.4: Example 4.1.3: when the number of cycle steps does not complete the full orbit, the compiler must retain all gates to preserve the automaton's semantics. The input is rejected with probability 1.

4.2 Measure-Many One-Way Quantum Finite Automaton to Circuit

The circuit-level translation of a MM-1QFA follows many of the same principles used for the measure-once case, with a crucial difference: instead of deferring measurement until the end of the computation, a projective measurement is performed after each input symbol is processed. This repeated measurement model introduces non-unitary branches into the computation, enabling the automaton to halt early—either accepting or rejecting—based on intermediate outcomes. As a result, the circuit must incorporate mid-computation measurements, conditional halting logic, and branching structure, reflecting the MM-1QFA's hybrid quantum-classical behaviour.

This section describes how to compile a MM-1QFA into a quantum circuit that correctly reproduces its acceptance semantics, with attention to managing measurement timing, decomposing the three-outcome measurement structure, and ensuring termination guarantees for all input strings.

4.2.1 Mapping Automaton Components to Circuit Elements

The MM-1QFA model is defined by the tuple

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

as introduced in Section 3.1.2. Compared to the MO-1QFA case, the key structural difference lies in the measurement strategy: whereas a MO-1QFA performs a single final measurement, a MM-1QFA applies a projective measurement after every symbol.

The first compilation steps closely follow those in Section 4.1, and only diverge when handling measurements:

State Register and Qubit Allocation

The internal state set Q is represented using $n = \lceil \log_2 |Q| \rceil$ qubits, encoding each classical state $q \in Q$ as a computational basis state $|q\rangle$. This is identical to the measure-once case.

Initialisation

The quantum register is initialised in the basis state corresponding to the initial state $q_0 \in Q$. If q_0 is a computational basis vector under the chosen encoding, this requires only a few X gates to flip the default all-zero state to $|q_0\rangle$.

Symbol-Dependent Unitary Evolution

Each input symbol $\sigma \in \Sigma$ is associated with a unitary matrix U_σ acting on the n-qubit register. The operator U_σ is applied before measurement at each step. These matrices are later compiled into native gates as in the MO-1QFA case.

Repeated Measurement

After each unitary U_σ is applied, a projective measurement is performed to decide whether the computation halts. The measurement is defined by three mutually orthogonal subspaces corresponding to:

- the accepting states Q_{acc},
- the rejecting states Q_{rej}, and
- the continuing (non-halting) states $Q_{\text{non}} = Q \setminus (Q_{\text{acc}} \cup Q_{\text{rej}})$.

The corresponding projectors are:

$$P_{\text{acc}} = \sum_{q \in Q_{\text{acc}}} |q\rangle\langle q| , \quad P_{\text{rej}} = \sum_{q \in Q_{\text{rej}}} |q\rangle\langle q| , \quad P_{\text{non}} = I - P_{\text{acc}} - P_{\text{rej}}.$$

The outcome determines whether the computation halts (accepts or rejects) or contines to the next symbol. This makes the circuit fundamentally hybrid: unitary evolution is interrupted by measurements and conditional control logic, which must be tracked classically.

Classical Control Flow

Unlike the MO-1QFA case, which requires only a final measurement, the MM-1QFA circuit must include measurement operations at each time step, as well as classical post-processing to determine whether further computation is needed. In practical implementations, this control flow may be realised via classical conditionals or early termination logic, depending on the hardware platform and circuit model (e.g., mid-circuit measurements with feedback).

Automaton part	Circuit realisation	Explanation
Q	n-qubit basis states	Encode each $q \in Q$ as $|q\rangle$ with $n = \lceil \log_2 |Q| \rceil$.
Σ	unitaries U_σ	Each input symbol σ applies U_σ to the state register.
δ	set $\{U_\sigma\}$	Transition operators, later decomposed into native gates.
q_0	initial state $|q_0\rangle$	Prepared from $|0\rangle^{\otimes n}$ using X gates if necessary.
Q_{acc}, Q_{rej}	repeated three-outcome measurements	After each U_σ, apply a measurement $\{P_{\text{acc}}, P_{\text{rej}}, P_{\text{non}}\}$. Outcome determines halting or continuation.
Control flow	classical feedback	Conditional termination after each step based on measurement results.

Table 4.2: Mapping MM-1QFA components to quantum-circuit constructs.

4.2.2 General Compilation Algorithm

The compilation of a MM-1QFA into a quantum circuit proceeds in two structured steps:

1. **Template Generation:** A symbolic circuit is created to reflect the automaton's structure for input words of fixed length, using placeholders for both unitaries and measurements.
2. **Instantiation and Execution:** For a specific word, the placeholders are replaced with the actual gates implementing the automaton's transitions, and the computation proceeds with intermediate measurements after each symbol.

Let $F = Q_{\text{acc}}$ and $R = Q_{\text{rej}}$ denote, respectively, the sets of accepting and rejecting states. The three-outcome measurement applied after each step is defined by the orthogonal projectors:

$$P_{\text{acc}} = \sum_{q \in F} |q\rangle\langle q|, \qquad P_{\text{rej}} = \sum_{q \in R} |q\rangle\langle q|, \qquad P_{\text{cont}} = I - P_{\text{acc}} - P_{\text{rej}}.$$

Algorithm 3 Template Generation for a MM-1QFA Circuit

Require: Automaton $A = (Q, \Sigma, \delta, q_0, F, R)$, input length L
Ensure: Parametric circuit template with symbolic placeholders

```
n ← ⌈log₂ |Q|⌉
Initialise n qubits in state |q₀⟩
for i = 1 to L do
    Insert symbolic gate [U_{x_i}]
    Insert symbolic measurement {P_acc, P_rej, P_cont}
end for
```

This circuit skeleton remains independent of any particular input string and can be reused for all words of the same length.

To make the template executable, the symbolic gates are instantiated as concrete decompositions according to the input word $x = x_1 x_2 \ldots x_L \in \Sigma^L$. At each step, the automaton either halts or continues depending on the result of the three-outcome measurement.

Algorithm 4 Instantiation and Execution of a Compiled MM-1QFA Circuit

Require: Input $x = x_1 x_2 \ldots x_L$, circuit template, gate library or decomposition scheme for each U_σ
Ensure: Acceptance probability $p_M(x)$

$p \leftarrow 0$ ▷ Cumulative acceptance probability
Initialise quantum state $|\psi_0\rangle = |q_0\rangle$
for $i = 1$ **to** L **do**
  Replace $\boxed{U_{x_i}}$ with gate decomposition for U_{x_i}
  Apply U_{x_i} to current state $|\psi_{i-1}\rangle$
  Compute outcome probabilities:

$$p_{\text{acc}}^{(i)} = \|P_{\text{acc}} |\psi_i\rangle\|^2, \quad p_{\text{rej}}^{(i)} = \|P_{\text{rej}} |\psi_i\rangle\|^2, \quad p_{\text{cont}}^{(i)} = \|P_{\text{cont}} |\psi_i\rangle\|^2$$

  $p \leftarrow p + p_{\text{acc}}^{(i)}$
  Collapse $|\psi_i\rangle$ to $\frac{P_{\text{cont}}|\psi_i\rangle}{\|P_{\text{cont}}|\psi_i\rangle\|}$ if $p_{\text{cont}}^{(i)} > 0$
  if $p_{\text{cont}}^{(i)} = 0$ **then**
    break ▷ Computation halts in accepting or rejecting subspace
  end if
end for
return $p_M(x) = p$

This two-step strategy preserves the semantics of the original MM-1QFA, interleaving unitary evolution and projective measurements. On platforms supporting mid-circuit measurements and classical branching, this logic can be directly implemented. On static circuit backends, equivalent semantics may be approximated using classically post-processed measurement results or circuit unfolding.

4.2.3 Step-by-Step Examples

To illustrate how a MM-1QFA is compiled into a circuit, we now present explicit examples that walk through each phase of the translation. These examples highlight the distinguishing features of the measure-many model, including intermediate measurements, early halting, and hybrid classical-quantum control. Each circuit begins with the symbolic template generated by the compiler, proceeds through instantiation for a specific input word, and concludes with either a full execution or early termination based on measurement outcomes. The goal is to clarify how the abstract operational semantics of the automaton are faithfully realised in the corresponding gate-level circuit.

Early Acceptance on $x = ab$ Consider a MM-1QFA defined over the state space

$$Q = \{q_0, q_1, q_2\}, \quad \Sigma = \{a, b\}, \quad q_0 \text{ initial}, \quad Q_{\text{acc}} = \{q_2\}, \quad Q_{\text{rej}} = \varnothing.$$

Let the transition unitaries be defined as follows:

- U_a maps $|q_0\rangle \mapsto |q_1\rangle$,
- U_b maps $|q_1\rangle \mapsto |q_2\rangle$.

The goal is to process the input string $x = ab$ of length $L = 2$ and observe early acceptance behaviour.

1. **Qubit allocation.** The automaton has three basis states, so

   $$n = \lceil \log_2 3 \rceil = 2 \text{ qubits}.$$

   We assume an encoding where q_0, q_1, and q_2 correspond to $|00\rangle$, $|01\rangle$, and $|10\rangle$, respectively.

2. **Initialisation.** The register is prepared in $|q_0\rangle = |00\rangle$.

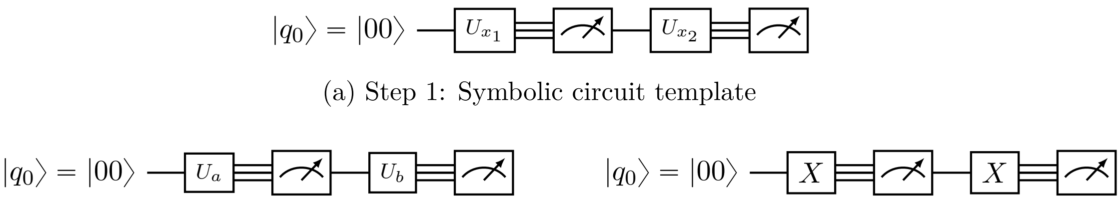
3. **Unitary evolution and template generation.** Since $x = ab$ has length 2, the compiler generates a circuit with two slots for U_{x_1} and U_{x_2}, each followed by a three-outcome measurement. The symbolic circuit template is shown in subfigure 4.5a.

4. **Instantiation.** We substitute U_a and U_b into the template to obtain the concrete circuit for $x = ab$, shown in subfigure 4.5b.

5. **Gate synthesis.** Suppose U_a and U_b are implemented using Pauli-X gates acting on the encoded state transitions (e.g., $|00\rangle \mapsto |01\rangle$ and $|01\rangle \mapsto |10\rangle$). The resulting circuit uses two X gates interleaved with measurements, as shown in subfigure 4.5c.

6. **Measurement semantics.** After applying U_a, the system moves from $|q_0\rangle$ to $|q_1\rangle$. Since $q_1 \notin Q_{\text{acc}} \cup Q_{\text{rej}}$, the first measurement yields CONTINUE. After applying U_b, the system transitions to $|q_2\rangle \in Q_{\text{acc}}$, so the second measurement yields ACCEPT and the computation halts. Thus,

$$p_M(ab) = 1.$$



(a) Step 1: Symbolic circuit template

(b) Step 2: Instantiation for $x = ab$

(c) Step 2: Gate-level synthesis

Figure 4.5: Compilation stages for Example 4.2.3. Input ab causes an early accept: after U_b, the system reaches $|q_2\rangle$, a final state.

Early Rejection on $x = b$ Consider a MM-1QFA defined over the state space

$$Q = \{q_0, q_1\}, \quad \Sigma = \{b\}, \quad q_0 \text{ initial}, \quad Q_{\text{acc}} = \varnothing, \quad Q_{\text{rej}} = \{q_1\}.$$

Let the transition function be defined such that:

- U_b maps $|q_0\rangle \mapsto |q_1\rangle$.

We examine the behaviour of the automaton on the input string $x = b$ of length $L = 1$.

1. **Qubit allocation.** The automaton has two basis states, so the number of required qubits is
$$n = \lceil \log_2 2 \rceil = 1.$$
The classical states are encoded as $|q_0\rangle = |0\rangle$ and $|q_1\rangle = |1\rangle$.

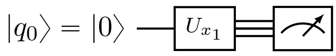
2. **Initialisation.** The register is initialised in the state $|q_0\rangle = |0\rangle$, which requires no gate-level preparation.

3. **Unitary evolution and template generation.** The input $x = b$ consists of a single symbol, so the compiler generates a template with one symbolic unitary gate followed by a measurement. This is shown in subfigure 4.6a.

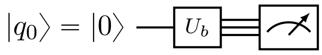
4. **Instantiation.** The placeholder U_{x_1} is replaced by the unitary U_b, as shown in subfigure 4.6b.

5. **Gate synthesis.** Assuming U_b maps $|0\rangle \mapsto |1\rangle$, it can be implemented directly as a Pauli-X gate:
$$U_b = X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

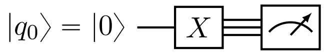
6. **Measurement semantics.** The measurement is performed immediately after U_b. Since $U_b |q_0\rangle = |q_1\rangle$ and $q_1 \in Q_{\text{rej}}$, the outcome is REJECT with probability 1. The computation halts after a single step, and the automaton returns output 0:
$$p_M(b) = \|P_{\text{rej}} |q_1\rangle\|^2 = 1.$$



(a) Step 1: Symbolic circuit template



(b) Step 2: After instantiation for $x = b$



(c) Step 2: Gate-level synthesis

Figure 4.6: Compilation stages for Example 4.2.3. The input b triggers an immediate transition to $|q_1\rangle$, a rejecting state. The automaton halts after one step and returns REJECT.

4.3 Unitary Operators Instantiation

Once the circuit skeleton is constructed for a given MO-1QFA or MM-1QFA, the remaining compilation step consists in instantiating the placeholder unitaries U_σ with actual gate-level operators. This section presents the main strategies available to achieve such instantiation, highlighting the trade-offs between offline preprocessing and dynamic runtime synthesis.

4.3.1 Offline Synthesis

The *offline synthesis* strategy precomputes a gate decomposition for each unitary matrix U_σ associated with the alphabet Σ. This approach is particularly suitable when the automaton is fixed and used to process multiple inputs of the same language class. The steps are:

- For every $\sigma \in \Sigma$, extract the unitary matrix U_σ defined by the automaton's transition function δ.
- Decompose U_σ into a circuit of elementary gates from a fixed universal set (e.g., Clifford+T or $\{\text{CNOT}, R_z, H\}$).
- Store each gate sequence as a reusable fragment in a gate library.

This method guarantees high performance at runtime, as no decomposition is needed during execution. However, it requires more memory to store all precompiled gate sequences, and it lacks adaptability in contexts where U_σ changes dynamically or is defined procedurally.

4.3.2 Template-Based Parameter Loading

An alternative method is *template-based parameter loading*, where the circuit skeleton includes parametrized gates (e.g., Euler-angle rotations) and the actual rotation angles are injected at runtime based on the specific U_σ required. This is achieved by:

- Designing each U_σ as a composition of generic rotation gates (e.g., $R_z(\theta_1)R_y(\theta_2)R_z(\theta_3)$).
- Computing the angles $\theta_1, \theta_2, \theta_3$ classically using a synthesis algorithm (e.g., ZYZ decomposition) from the matrix representation of U_σ.

- Populating the parametrized gates of the circuit with the computed angles just before execution.

This strategy supports adaptive and memory-efficient compilation, especially useful when the automaton model is generated on-the-fly or when circuits are embedded in larger configurable pipelines. The downside is the runtime overhead incurred by angle computation and dynamic loading.

4.3.3 Hybrid and Optimized Approaches

In practice, a hybrid scheme combining both methods is often adopted. Common unitary matrices with known decompositions can be stored offline, while less frequent or dynamically generated ones are handled through runtime parameter loading. Furthermore, if the automaton contains symmetries (e.g., cyclic state transitions), structural optimizations can reduce the number of distinct unitaries needed, enabling further compression of the circuit template.

4.3.4 Summary

Unitary instantiation closes the automaton-to-circuit translation by assigning concrete quantum operations to each input-driven evolution. Offline synthesis prioritizes speed and repeatability; template-based methods emphasize flexibility and memory economy. The choice depends on the application domain—static recognizers may favor offline strategies, while programmable quantum systems benefit from dynamic parameter loading.

5. Conclusion

Abbreviations

ω-QFA	ω-Quantum Finite Automaton
kTQCFA	k-Tape Quantum Finite Automaton with Classical States
l-VFA	l-valued Finite Automaton
l-VPDA	l-valued Pushdown Automaton
1QCFA	One-Way Quantum Finite Automaton with Classical States
1QFA	One-Way Quantum Finite Automaton
1QFA(2)	One-Way Quantum Finite Automaton with Two Observables
1QFkCA	Quantum Finite k-Counter Automaton
1gQFA	Generalised Quantum Finite Automaton
2QCFA	Two-Way Quantum Finite Automaton with Classical States
2QF1CA	Two-Way Quantum Finite One-Counter Automaton
2QFA	Two-Way Quantum Finite Automaton
2T1QFA(2)	Two-Tape One-Way Quantum Finite Automaton with Two Heads
2TQCFA	Two-Tape Quantum Finite Automaton with Classical States
CL-1QFA	One-Way Quantum Finite Automaton with Control Language
E-1QFA	Enhanced One-Way Quantum Finite Automaton
LQFA	Latvian Quantum Finite Automaton
ML-MMQFA	Multi-Letter Measure Many Quantum Finite Automaton
ML-QFA	Multi-Letter Quantum Finite Automaton
ML-RevQFA	Multi-Letter Reversible Quantum Finite Automaton
MM-1gQFA	Measure Many Generalised Quantum Finite Automaton
MM-1QFA	Measure Many One-Way Quantum Finite Automaton

MM-2QFA	Measure Many Two-Way Quantum Finite Automaton
MO-1gQFA	Measure Once Generalised Quantum Finite Automaton
MO-1QFA	Measure Once One-Way Quantum Finite Automaton
MO-2QFA	Measure Once Two-Way Quantum Finite Automaton
MON-1QFA	Measure-Only One-Way Quantum Finite Automaton
NISQ	Noisy Intermediate-Scale Quantum
NQFA	Non-Deterministic Quantum Finite Automaton
PQFA	Postselection Quantum Finite Automaton
QAM	Quantum Arthur-Merlin
QF1CA	Quantum Finite One-Counter Automaton
QFA	Quantum Finite Automaton
QFT	Quantum Fourier Transform
QIP	Quantum Interactive Proof
QTM	Quantum Turing Machine
RevQFA	Reversible One-Way Quantum Finite Automaton
RTQ1CA	Real-Time Quantum One-Counter Automaton

Bibliography

[1] Scott Aaronson. "Quantum computing, postselection, and probabilistic polynomial-time". In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 461.2063 (2005), pp. 3473–3482.

[2] Farid Ablayev and Aida Gainutdinova. "On the lower bounds for one-way quantum automata". In: *International Symposium on Mathematical Foundations of Computer Science*. Springer. 2000, pp. 132–140.

[3] Andris Ambainis and Rusins Freivalds. "1-way quantum finite automata: strengths, weaknesses and generalizations". In: *Proceedings 39th annual symposium on foundations of computer science (Cat. No. 98CB36280)*. IEEE. 1998, pp. 332–341.

[4] Andris Ambainis and John Watrous. "Two-way finite automata with quantum and classical states". In: *Theoretical Computer Science* 287.1 (2002), pp. 299–311.

[5] Andris Ambainis et al. "Probabilities to accept languages by quantum finite automata". In: *International computing and combinatorics conference*. Springer. 1999, pp. 174–183.

[6] Aleksandrs Belovs, Ansis Rosmanis, and Juris Smotrovs. "Multi-letter reversible and quantum finite automata". In: *Developments in Language Theory: 11th International Conference, DLT 2007, Turku, Finland, July 3-6, 2007. Proceedings 11*. Springer. 2007, pp. 60–71.

[7] Alberto Bertoni, Carlo Mereghetti, and Beatrice Palano. "Quantum computing: 1-way quantum automata". In: *Developments in Language Theory: 7th International Conference, DLT 2003 Szeged, Hungary, July 7–11, 2003 Proceedings 7*. Springer. 2003, pp. 1–20.

[8] Alberto Bertoni, Carlo Mereghetti, and Beatrice Palano. "Trace monoids with idempotent generators and measure-only quantum automata". In: *Natural Computing* 9.2 (2010), pp. 383–395.

[9] Aija Bērziņa and Richard Bonner. "Ambainis-Freivalds' Algorithm for Measure-Once Automata". In: *Fundamentals of Computation Theory: 13th International Symposium, FCT 2001 Riga, Latvia, August 22–24, 2001 Proceedings 13*. Springer. 2001, pp. 83–93.

[10] Amandeep Singh Bhatia and Ajay Kumar. "Quantum ω-automata over infinite words and their relationships". In: *International Journal of Theoretical Physics* 58 (2019), pp. 878–889.

[11] Maria Paola Bianchi, Carlo Mereghetti, and Beatrice Palano. "Complexity of promise problems on classical and quantum automata". In: *Computing with New Resources: Essays Dedicated to Jozef Gruska on the Occasion of His 80th Birthday*. Springer, 2014, pp. 161–175.

[12] Maria Paola Bianchi, Carlo Mereghetti, and Beatrice Palano. “On the power of one-way automata with quantum and classical states”. In: *Implementation and Application of Automata: 19th International Conference, CIAA 2014, Giessen, Germany, July 30–August 2, 2014. Proceedings 19*. Springer. 2014, pp. 84–97.

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

[14] Richard Bonner, Rūsiņš Freivalds, and Maksim Kravtsev. “Quantum versus probabilistic one-way finite automata with counter”. In: *International Conference on Current Trends in Theory and Practice of Computer Science*. Springer. 2001, pp. 181–190.

[15] Alex Brodsky and Nicholas Pippenger. “Characterizations of 1-way quantum finite automata”. In: *SIAM Journal on Computing* 31.5 (2002), pp. 1456–1478.

[16] Alessandro Candeloro et al. “An enhanced photonic quantum finite automaton”. In: *Applied Sciences* 11.18 (2021), p. 8768.

[17] AC Cem Say and Abuzer Yakaryilmaz. “Quantum counter automata”. In: *International Journal of Foundations of Computer Science* 23.05 (2012), pp. 1099–1116.

[18] Wenjing Chu et al. “Approximately Learning Quantum Automata”. In: *International Symposium on Theoretical Aspects of Software Engineering*. Springer. 2023, pp. 268–285.

[19] Massimo Pica Ciamarra. “Quantum reversibility and a new model of quantum automaton”. In: *International Symposium on Fundamentals of Computation Theory*. Springer. 2001, pp. 376–379.

[20] Carlo Comin. “(Extended Version) Algebraic Characterization of the Class of Languages recognized by Measure Only Quantum Automata”. In: *arXiv preprint arXiv:1301.3931* (2013).

[21] David Deutsch. “Quantum theory, the Church–Turing principle and the universal quantum computer”. In: *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences* 400.1818 (1985), pp. 97–117.

[22] Rūsiņš Freivalds. “Languages recognizable by quantum finite automata”. In: *International Conference on Implementation and Application of Automata*. Springer. 2005, pp. 1–14.

[23] Debayan Ganguly and Kumar Sankar Ray. “2-tape 1-way quantum finite state automata”. In: *arXiv preprint arXiv:1607.00811* (2016).

[24] Jozef Gruska, Daowen Qiu, and Shenggen Zheng. “Potential of quantum finite automata with exact acceptance”. In: *International Journal of Foundations of Computer Science* 26.03 (2015), pp. 381–398.

[25] Attila Kondacs and John Watrous. “On the power of quantum finite state automata”. In: *Proceedings 38th annual symposium on foundations of computer science*. IEEE. 1997, pp. 66–75.

[26] Maksim Kravtsev. “Quantum finite one-counter automata”. In: *International Conference on Current Trends in Theory and Practice of Computer Science*. Springer. 1999, pp. 431–440.

[27] Lvzhou Li and Yuan Feng. “On hybrid models of quantum finite automata”. In: *Journal of Computer and System Sciences* 81.7 (2015), pp. 1144–1158.

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

[29] T Lin. “On equivalence and emptiness problems of multi-letter (measure many) quantum finite automata”. In: *arXiv preprint arXiv:1203.0113* (2012).

[30] Tianrong Lin. “Another approach to the equivalence of measure-many one-way quantum finite automata and its application”. In: *Journal of Computer and System Sciences* 78.3 (2012), pp. 807–821.

[31] Eduardo Willwock Lussi et al. “Implementing a Quantum Finite Automaton in IBMQ using Custom Control Pulses”. In: *arXiv preprint arXiv:2412.06977* (2024).

[32] Paulo Mateus, Daowen Qiu, and Lvzhou Li. “On the complexity of minimizing probabilistic and quantum automata”. In: *Information and Computation* 218 (2012), pp. 36–53.

[33] Mark Mercer. “Lower bounds for generalized quantum finite automata”. In: *Language and Automata Theory and Applications: Second International Conference, LATA 2008, Tarragona, Spain, March 13-19, 2008. Revised Papers 2*. Springer. 2008, pp. 373–384.

[34] Carlo Mereghetti and Beatrice Palano. “Quantum finite automata with control language”. In: *RAIRO-Theoretical Informatics and Applications* 40.2 (2006), pp. 315–332.

[35] Carlo Mereghetti, Beatrice Palano, and Priscilla Raucci. “Unary quantum finite state automata with control language”. In: *Applied Sciences* 14.4 (2024), p. 1490.

[36] Cristopher Moore and James P Crutchfield. “Quantum automata and quantum grammars”. In: *Theoretical Computer Science* 237.1-2 (2000), pp. 275–306.

[37] Ashwin Nayak. “Optimal lower bounds for quantum automata and random access codes”. In: *40th Annual Symposium on Foundations of Computer Science (Cat. No. 99CB37039)*. IEEE. 1999, pp. 369–376.

[38] 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.

[39] Harumichi Nishimura and Tomoyuki Yamakami. “Interactive proofs with quantum finite automata”. In: *Theoretical Computer Science* 568 (2015), pp. 1–18.

[40] Soumya Debabrata Pani and Chandan Kumar Behera. “Empowering measures for quantum finite automata (QFA)”. In: *2011 3rd International Conference on Computer Research and Development*. Vol. 2. IEEE. 2011, pp. 406–411.

[41] Carla Piazza and Riccardo Romanello. “Mirrors and memory in quantum automata”. In: *International Conference on Quantitative Evaluation of Systems*. Springer. 2022, pp. 359–380.

[42] Daowen Qiu. “Automata theory based on quantum logic: reversibilities and pushdown automata”. In: *Theoretical Computer Science* 386.1-2 (2007), pp. 38–56.

[43] Daowen Qiu. “State complexity of operations on two-way quantum finite automata”. In: *arXiv preprint arXiv:0807.0476* (2008).

[44] Daowen Qiu and Mingsheng Ying. “Characterizations of quantum automata”. In: *Theoretical computer science* 312.2-3 (2004), pp. 479–489.

[45] Daowen Qiu and Sheng Yu. “Hierarchy and equivalence of multi-letter quantum finite automata”. In: *Theoretical Computer Science* 410.30-32 (2009), pp. 3006–3017.

[46] Daowen Qiu et al. “Decidability of the equivalence of multi-letter quantum finite automata”. In: *arXiv preprint arXiv:0812.1061* (2008).

[47] Daowen Qiu et al. “Multi-letter quantum finite automata: decidability of the equivalence and minimization of states”. In: *Acta informatica* 48.5 (2011), p. 271.

[48] DW Qiu, Paulo Mateus, and Amilcar Sernadas. “One-way quantum finite automata together with classical states: Equivalence and Minimization”. In: *arXiv preprint arXiv:0909.1428* (2009).

[49] Jibran Rashid and Abuzer Yakaryılmaz. “Implications of quantum automata for contextuality”. In: *Implementation and Application of Automata: 19th International Conference, CIAA 2014, Giessen, Germany, July 30–August 2, 2014. Proceedings 19*. Springer. 2014, pp. 318–331.

[50] Zachary Remscrim. “Lower bounds on the running time of two-way quantum finite automata and sublogarithmic-space quantum turing machines”. In: *arXiv preprint arXiv:2003.09877* (2020).

[51] Zachary Remscrim. “The power of a single qubit: Two-way quantum finite automata and the word problem”. In: *arXiv preprint arXiv:2003.09879* (2020).

[52] Oksana Scegulnaja-Dubrovska, Lelde Lāce, and Rūsiņš Freivalds. “Postselection finite quantum automata”. In: *International Conference on Unconventional Computation*. Springer. 2010, pp. 115–126.

[53] Zhengjun Xi, Xin Wang, and Yongming Li. “Some algebraic properties of measure-once two-way quantum finite automata”. In: *Quantum Information Processing* 7 (2008), pp. 211–225.

[54] Ligang Xiao and Daowen Qiu. “State complexity of one-way quantum finite automata together with classical states”. In: *arXiv preprint arXiv:2112.03746* (2021).

[55] Abuzer Yakaryilmaz and AC Say. “Languages recognized by nondeterministic quantum finite automata”. In: *arXiv preprint arXiv:0902.2081* (2009).

[56] Tomoyuki Yamakami. “One-way reversible and quantum finite automata with advice”. In: *Information and Computation* 239 (2014), pp. 122–148. ISSN: 0890-5401.

[57] A Chi-Chih Yao. “Quantum circuit complexity”. In: *Proceedings of 1993 IEEE 34th Annual Foundations of Computer Science*. IEEE. 1993, pp. 352–361.

[58] Shenggen Zheng, Lvzhou Li, and Daowen Qiu. “Two-tape finite automata with quantum and classical states”. In: *International Journal of Theoretical Physics* 50.4 (2011), pp. 1262–1281.

[59] 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.

[60] Shenggen Zheng et al. “One-Way Finite Automata with Quantum and Classical States.” In: *Languages alive* 7300 (2012), pp. 273–290.

[61] Shenggen Zheng et al. “State succinctness of two-way finite automata with quantum and classical states”. In: *Theoretical Computer Science* 499 (2013), pp. 98–112.

Acknowledgments