Universiteit Leiden

Symbolic Simulation of Quantum Finite Automata

Sophie Kuo (s3915069)

Supervisors:
Marcello Bonsague & Nusa Zidaric

Bachelor Thesis– Informatica

Leiden Institute of Advanced Computer Science (LIACS)
liacs.leidenuniv.nl

July 1, 2026

Abstract

Quantum finite automata (QFA) are simple models of quantum computation. In particular, the behavior of a measure-once quantum finite automaton (MO-QFA), is given by a sequence of multiplications of quantum operators represented as unitary matrices, determined by the input string. It is ended by a single measurement resulting in the probability of an outcome. A naive simulation of MO-QFAs relies on numerical linear algebra over complex vectors. While polynomial in the dimension of the automaton and the input length, in the past such simulations typically used floating-point arithmetic and therefore suffered from approximation errors. Alternative previous approaches based on encoding of the state evolution as boolean satisfiability problem (SAT) do not consider the linear structure of quantum evolution, leading to an exponential number of Boolean variables. Our symbolic approach over the field extension preserves linearity exact for Clifford operations, and provably bounded-error otherwise. Although the state representation remains exponential in the number of qubits, our method eliminates approximation errors and maintains a closed-form algebraic representation.

Contents

1 Introduction

In this chapter we give an introduction to quantum computing, quantum finite automata, and the simulation thereof, ending with a short thesis overview.

1.1 Current Situation

A *quantum computer* [15] is a computer of which quantum concepts such as superposition and entanglement are essential parts. Instead of the classical bit, it uses quantum bits or *qubits* as its unit of information. These qubits can not only take on the values 0 and 1, but also find themselves in a *superposition* of these basis states. Using this superposition property, and wave interference, the quantum computer can perform quantum algorithms.

In the last few decades, scientific interest in quantum computing has increased considerably, following the discovery that quantum computers may have significantly more computing power than their classical counterparts [18]. However, no quantum computer in existence is ready for large-scale/general-purpose real-world applications yet, as the engineering of physical qubits has proven to be difficult. In the meantime, the computational power and expressiveness of quantum machines can still be studied using theoretical models.

One of these models of quantum computing is the *quantum finite automaton* (QFA) [13], the quantum equivalent of the classical finite automata. The QFA allows for a study of quantum computing in a way that is minimal, yet expressive. The finite dimensional state-space of a QFA is easy to translate to an implementation as a system with finitely many particles, so study of QFA's could eventually also benefit any physical implementation of a quantum computer.

Simulation of a quantum computer is a difficult task due to its complexity. A QFA is a simpler model, but finding an exact manner of simulation is still a complex task, especially as the number of states and/or the size of the alphabet grows [11]. This is due to the fact that past attempts typically used floating-point arithmetic and therefore suffered from approximation errors [11]. Alternative previous approaches based on encoding of the state evolution as boolean satisfiability problem (SAT) problems [3] suffered from exponential blow-up and a possible loss of linear structure.

In this thesis we aim to develop a *symbolic* simulation for QFA's, aiming towards an efficient and possibly exact simulation of its behaviour. The question we seek to answer is:

Can measure-once quantum finite automata be simulated efficiently and exactly using symbolic matrix representations over finite algebraic field extensions?

1.2 Thesis Overview

This thesis consists of six chapters, with subsections and references. This chapter contains the introduction of the current situation; Section 2 provides necessary background on , quantum mechanics and computing, classical automata and quantum automata; Section 3 discusses related work; Section 4 describes the steps taken in our approach: our usage of finite field extension, and our definition of the seq-MO-QFA, that are needed in the symbolic simulation algorithm presented in Section 5, where we also show some examples; Section 6 concludes and discusses possible further research.

This bachelor thesis was conducted as part of the bachelor class of the computer science programme at LIACS, under the supervision of Marcello Bonsangue and Nusa Zidaric.

2 Background

In this chapter we will establish the theoretical and mathematical foundation needed in the rest of the thesis. We first review the relevant part of quantum computing, then we introduce the universal quantum gate set. We conclude with recalling classical finite automata (FA), and a class of quantum finite automata (QFA).

2.1 Quantum Computing Foundation

In this section we review the relevant part of quantum computing, first we introduce the qubit and single- and two-qubit systems, then quantum gates.

2.1.1 Single- and Two-Qubit Systems

In quantum computing, information is encoded in quantum bits, or *qubits*, the quantum equivalent of the classical bit. A qubit is a two-level system, mathematically represented as a unit vector in a two-dimensional complex vector space. Synonymous to a classical bit, a qubit can have values '0' and '1', these are the basis states $|0\rangle$ and $|1\rangle$. Here Dirac notation is used, the standard notation for quantum states. However, as opposed to a classical bit, a qubit can also take on other values than $|0\rangle$ and $|1\rangle$. It can also be in *superposition*, which is any linear combination of the $|0\rangle$ and $|1\rangle$ basis states:

$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, where $\alpha, \beta \in \mathbb{C}$. [15]

To obtain this quantum state (the values of α and β), we cannot measure a qubit directly. If a qubit is measured, we will either observe $|0\rangle$ with probability $|\alpha|^2$, or $|1\rangle$ with probability $|\beta|^2$. Since they are probabilities, naturally $|\alpha|^2 + |\beta|^2 = 1$.

To calculate the probability of the state $|\psi\rangle$ being projected onto a basis state $|q\rangle$:

$P = |\langle\psi|q\rangle|^2$

Representation of a single-qubit state as a vector is done in the following way:

$$|\psi\rangle = \alpha \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \beta \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$$

When working with two qubits, the combination of the two single-qubit basis states $|0\rangle$ and $|1\rangle$ leads to four new computational basis states for the two-qubit system: $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$. The system can also reside in any superposition of these four states:

$|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle$, where $\alpha_x \in \mathbb{C}$. [15]

Similar to single-qubit systems, we obtain measurement result x (= 00, 01, 10, 11) with probability $|\alpha_x|^2$.

Representation of a two-qubit system as a vector is done in the following way:

$$|\psi\rangle = \alpha_{00}\begin{bmatrix}1\\0\\0\\0\end{bmatrix} + \alpha_{01}\begin{bmatrix}0\\1\\0\\0\end{bmatrix} + \alpha_{10}\begin{bmatrix}0\\0\\1\\0\end{bmatrix} + \alpha_{11}\begin{bmatrix}0\\0\\0\\1\end{bmatrix} = \begin{bmatrix}\alpha_{00}\\\alpha_{01}\\\alpha_{10}\\\alpha_{11}\end{bmatrix}$$

More generally, a quantum system consisting of n qubits has 2^n basis states.

2.1.2 Quantum Gates

Quantum computation can be formulated as a sequence of quantum *gates* acting on a register of qubits. These gates are reversible transformations of the quantum state and are represented by unitary matrices:

Definition 2.1 (Unitary matrix)**.** *A matrix U is unitary if its matrix inverse U^{-1} equals its conjugate transpose U^*, that is, if:*

$$U^*U = UU^* = I,$$

where I is the identity matrix.

Applying a quantum gate X to a state, represented by a state vector $|\psi\rangle$, is then simply a matter of multiplying the matrix with the vector $X|\psi\rangle$. For example, the quantum equivalent of the classical NOT gate is represented in the following way:

$$X \equiv \begin{bmatrix}0 & 1\\1 & 0\end{bmatrix},$$

and applying this gate to a generic, single-qubit state, results in:

$$X|\psi\rangle = \begin{bmatrix}0 & 1\\1 & 0\end{bmatrix}\begin{bmatrix}\alpha\\\beta\end{bmatrix} = \begin{bmatrix}\beta\\\alpha\end{bmatrix}$$

where the roles of $|0\rangle$ and $|1\rangle$ have been interchanged [15].

This X gate is a single-qubit gate, and is thus represented by a two-by-two matrix. A gate which operates on a two-qubit state is represented by a four-by-four matrix. If we want to represent two single-qubit gates X operating on a two-qubit system, we need to take the *tensor product* between these gates $X \otimes X$, to create a four-by-four matrix:

Definition 2.2 (Kronecker product)**.** *The Kronecker product is the tensor product of two matrices. If A is an $m \times n$ matrix and B a $p \times q$ matrix, the Kronecker product $A \otimes B$ is the $pm \times qn$ block matrix:*

$$A \otimes B = \begin{bmatrix} a_{11}B & \dots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \dots & a_{mn}B \end{bmatrix} \text{ [5],}$$

More generally, if a gate X acts on k qubits, we need to take the tensor product $X^{\otimes k}$.

If this gate acts on one of the two qubits, we need to take the tensor product with the *identity* matrix for the other. More generally, if a gate acts on k qubits in an n-qubit system, we need to take its tensor product with the identity matrix for the other $n-k$ qubits.

2.2 Universal Quantum Gates

In Section 4, we restrict our simulations to quantum automata whose unitary transitions are composed of gates from a small set. A fundamental result in quantum computing states that the set $\{H, T, CNOT\}$ is *universal* for quantum computation [15], where

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad T = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}, \quad CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

It is important to note that the entries of T are 1 and $e^{i\pi/4} = \frac{\sqrt{2}}{2} + i\frac{\sqrt{2}}{2} = \frac{1}{\sqrt{2}} + i\frac{1}{\sqrt{2}}$. Considering this gate set will allow us to do symbolic quantum computations within a simple algebraic structure.

Theorem 2.1 ((Solovay-Kitaev theorem)[8])**.** *The set $\{H, T, CNOT\}$ is universal for quantum computation in the following sense:*

1. *It generates a dense subgroup of the unitary group of unitary matrices on any number of qubits n.*
2. *For any n-qubit unitary U and any desired accuracy $\varepsilon > 0$, there exists a sequence of gates from $\{H, T, CNOT\}$ whose product V satisfies $\|U - V\| < \varepsilon$, where $\|\cdot\|$ is the operator norm.*
3. *The length of this sequence is $O(\log^c(1/\varepsilon))$ for some constant $c \approx 2$. The explicit sequence can be computed algorithmically and the actual value of the constant c is algorithm-dependent.*

To make the above theorem concrete, next we present two examples of unitaries on two-qubits and show how they can be represented — exactly or approximately — using the universal gate set $\{H, T, CNOT\}$.

2.2.1 Example A: Exact Synthesis of the SWAP Gate

Consider the **SWAP** gate, which exchanges two qubits q_0 and q_1:

$$\text{SWAP} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

SWAP is a Clifford gate (it permutes basis states) and can be decomposed exactly into the products of three CNOTs:

$$\text{SWAP} = CNOT_{0\to1} \cdot CNOT_{1\to0} \cdot CNOT_{0\to1},$$

where $CNOT_{c\to t}$ denotes CNOT with control q_c and target q_t. Clearly, the SWAP gate is exactly representable using only $\{H, T, CNOT\}$ (in fact, CNOT alone suffices, no H or T needed). The decomposition has length 3. More generally, the decomposition has length at most $O(n^2/\log n)$ [1], demonstrating that exact synthesis of Clifford unitaries is rather efficient.

2.2.2 Example B: Approximate Synthesis of a Non-Clifford Gate

The $\sqrt{\textbf{SWAP}}$ gate, is a non-Clifford two-qubit gate essential for generating entanglement that cannot be represented exactly using $\{H, T, CNOT\}$ alone:

$$\sqrt{\text{SWAP}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1+i}{\sqrt{2}} & \frac{1-i}{\sqrt{2}} & 0 \\ 0 & \frac{1-i}{\sqrt{2}} & \frac{1+i}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Its entries involve $e^{\pm i\pi/4} = \frac{1}{\sqrt{2}} \pm i\frac{1}{\sqrt{2}}$, which lie in $\mathbb{Q}(\sqrt{2}, i)$. However, $\sqrt{\text{SWAP}}$ is **not** in the Clifford group because the Clifford group only contains matrices whose entries are in $\{0, \pm1, \pm i\}/\sqrt{2}$ (up to global phase). Using the Solovay-Kitaev theorem, we can approximate $\sqrt{\text{SWAP}}$ to any desired accuracy ε with a sequence of H, T, and $CNOT$ gates. For instance, the following sequence of length 15 achieves an approximation error $\varepsilon \approx 0.01$ [19]:

$$CNOT \cdot H_0 \cdot T_0 \cdot H_0 \cdot CNOT \cdot H_1 \cdot T_1 \cdot H_1 \cdot T_0 \cdot (H \otimes H) \cdot CNOT \cdot H_0 \cdot T_0 \cdot H_0,$$

where $CNOT$ acts on qubit 0 as source and 1 as target, T_0 denotes $T \otimes I$ acting on qubit 0 (with identity on qubit 1), and T_1 denotes $I \otimes T$ acting on qubit 1 (and similarly for H).

In general, the Solovay-Kitaev theorem guarantees existence but does not give the shortest possible decomposition[8], and other algorithms based on brute-force search or look-up tables can give better results. For this thesis, we assume that a decomposition algorithm is fixed so that the simulation itself is independent of how the decomposition was obtained.

2.3 Classical Finite Automata

A classical *finite automaton* (FA) is a mathematical model of computation. A subclass of these, the *deterministic finite automaton*, is defined as the following:

Definition 2.1 (DFA). *A deterministic finite automaton is a 5-tuple $(Q, \Sigma, q_0, A, \delta)$, where*

Q is a finite set of states;

Σ is a finite input alphabet;

$q_0 \in Q$ is the initial state;

$A \subseteq Q$ *is the set of accepting states;*

$\delta : Q \times \Sigma \to Q$ *is the transition function.*

For any element q of Q and any symbol $\sigma \in \Sigma$, we interpret $\delta(q, \sigma)$ as the state to which the FA moves, if it is in state q and receives the input σ. [12]

Starting in its defined initial state, an FA processes a string $s \in \Sigma^*$ symbol by symbol, following the transitions dictated by its transition function. If it is in an accepting state at the end of the input string, the FA outputs that the given string was accepted. Otherwise, rejected.

The deterministic part of the DFA indicates that for any one input string, there is one possible sequence of operations the DFA can take, leading to an output that is the same each time the algorithm is run for this particular input.

A *nondetermistic finite automaton* (NFA) does not have this property. It is defined as the following:

Definition 2.2 (NFA)**.** *A nondeterministic finite automaton is a 5-tuple $(Q, \Sigma, q_0, A, \delta)$, where*

Q *is a finite set of states;*

Σ *is a finite input alphabet;*

$q_0 \in Q$ *is the initial state;*

$A \subseteq Q$ *is the set of accepting states;*

$\delta : Q \times (\Sigma \cup \{\epsilon\}) \to 2^Q$ *is the transition function.*

For every element q of Q and every element σ of $\Sigma \cup \{\epsilon\}$, we interpret $\delta(q, \sigma)$ as the set of states to which the FA can move, if it is in state q and receives the input σ, or, if $\sigma = \epsilon$, the set of states other than q to which the NFA can move from state q without receiving any input symbol. [12]

The transition function of an NFA maps onto a set of states, instead of a singular one. This means multiple transitions for the same symbol can lead from a state. Furthermore the addition of *empty transitions*, or *ϵ-transitions*, enables the NFA to move states without processing any input at all.

2.4 Quantum Finite Automata

A *quantum finite automaton* (QFA) is the quantum equivalent of the classical finite automaton. One of the simplest variants is the *measure-once quantum finite automaton*, which is defined as following:

Definition 2.3 (MO-QFA)**.** *A Measure-Once Quantum Finite Automaton is a 5-tuple $(Q, \Sigma, \{U_a\}_{a \in \Sigma}, q_0, Q_{acc}, Q_{rej})$, where:*

Q *is a finite set of basis states;*

Σ *is a finite input alphabet;*

each U_a is a unitary matrix on $\mathcal{H}_Q$;

$q_0 \in Q$ is the initial basis state;

$Q_{acc}, Q_{rej} \subseteq Q$ with $Q_{acc} \cap Q_{rej} = \emptyset$.

For each transition, the unitary matrix corresponding to the processed input is applied to the state. For an input $s = a_1 a_2 \cdots a_k \in \Sigma^*$, the automaton computes $|\psi_{\text{final}}\rangle = U_{a_k} \cdots U_{a_2} U_{a_1} |q_0\rangle$, starting from the vector $|q_0\rangle$ of dimension $|Q|$, with a 1 in the position corresponding to the state q_0 and 0 everywhere else. The resulting quantum state is then measured in the computational basis. It accepts if the outcome lies in Q_{acc}, and rejects if it lies in Q_{rej}. The acceptance probability is

$$P_{\text{acc}}(s) = \sum_{q \in Q_{\text{acc}}} |\langle q|\psi_{\text{final}}\rangle|^2.$$

3 Related Work

This chapter provides an overview of previous work related to quantum finite automata and simulation thereof.

Earlier research on quantum machines had more of a focus on quantum Turing machines [6], quantum logic gates [4], and quantum complexity [2]. Only later, the focus shifted to include a model intrinsically linked with these previous subjects, but generally considered simpler: the quantum finite state automaton [9, 10, 13]. This is the quantum equivalent of the classical finite state automaton [12], a long-established classical computational model.

This earliest research on QFA mostly concerned itself with definitions and computational power. Different categories were defined, such as the 1-way vs. 2-way quantum automata [10], or measure-once [13] vs. measure-many automata [10]. There has also been research on theoretical complexity bounds of quantum automata [7, 14] since.

A few attempts have been made at simultation of QFA and similar quantum algorithms. Lippa et al. [11] attempted to create a python library which provides a simple API for functionality required to simulate quantum finite automata. However, this research had more of a focus on time complexity in relation to automata factors, rather than exactness. The authors acknowledge that their simulated results are "quite close" to theoretically predicted, but thus not exactly.

4 Approach

In this chapter, we lay out the complete experimental framework. To summarize, we want to do an exact simulation of QFA's. To this end, we want to limit ourselves to the universal gate set, and simulate its operations on a generic state and some arbitrary, simple QFA's in a symbolic way. In the following sections, we explain two important steps: finite field extension and extension of the MO-QFA with seq-transitions.

4.1 Finite Field Extension for Exact Symbolic Computation

Classical simulations of quantum systems typically rely on floating-point arithmetic to represent complex amplitudes. This introduces rounding errors that accumulate over many gate operations, making it impossible to verify exact acceptance probabilities or to distinguish exponentially small probabilities from zero. In this thesis, we aim for exact symbolic simulation, which requires a representation of complex numbers that supports exact arithmetic.

We have seen that the universal gate set $\{H, T, CNOT\}$ (and the derived gate $S = T^2$) involves only two irrational constants: $\frac{1}{\sqrt{2}}$ (from H and T) and i (from S and T). To represent all amplitudes that appear during simulation using this unversal gate set, we work in the field extension [20]

$$\mathbb{Q}(\sqrt{2}, i) = \mathbb{Q}(\sqrt{2})(i),$$

obtained by adjoining $\sqrt{2}$ and then i to the rational numbers $\mathbb{Q}$. The field $\mathbb{Q}(\sqrt{2}, i)$ is a finite extension of $\mathbb{Q}$ of degree 4: a basis over $\mathbb{Q}$ is $\{1, \sqrt{2}, i, \sqrt{2}i\}$. This means every element is represented by four rational numbers, enabling exact arithmetic via rational operations, precisely what is needed for our specific universal gate set. Using $\frac{1}{\sqrt{2}} = \frac{\sqrt{2}}{2}$, every element $z \in \mathbb{Q}(\sqrt{2}, i)$ can be written uniquely in the *canonical form*

$$z = a + \frac{1}{\sqrt{2}}b + ci + \frac{1}{\sqrt{2}}di, \qquad a, b, c, d \in \mathbb{Q}, i = \sqrt{-1}$$

by absorbing the factor $1/2$ into the rational coefficients b and d. For example, the Hadamard gate entry $\frac{1}{\sqrt{2}}$ can be represented as $0 + \frac{1}{\sqrt{2}} + 0 \cdot i + 0 \cdot \frac{1}{\sqrt{2}}i$, while the T gate entry $e^{i\pi/4} = \frac{1}{\sqrt{2}} + i\frac{1}{\sqrt{2}}$ is represented as $0 + \frac{1}{\sqrt{2}} + 0 \cdot i + \frac{1}{\sqrt{2}}i$. Most important, all entries of H, S, T, and $CNOT$ lie in $\mathbb{Q}(\sqrt{2}, i)$.

The set $\mathbb{Q}(\sqrt{2}, i)$ is closed under addition, multiplication, and (for non-zero elements) inversion. This guarantees that any sequence of unitary operations composed from these gates has amplitudes that remain within the field. Moreover, the field is closed under complex conjugation:

$$\overline{a + b\sqrt{2} + ci + d\sqrt{2}i} = a + b\sqrt{2} - ci - d\sqrt{2}i.$$

This is essential for computing probabilities $|\psi|^2 = \psi \cdot \overline{\psi}$, which also lie in $\mathbb{Q}(\sqrt{2})$ (the real subfield), since the imaginary parts cancel.

More generally, the set of all algebraic numbers that appear as entries of products of H, S, T, and $CNOT$ gates is contained in the ring $\mathbb{Z}[1/\sqrt{2}, i]$, and each such entry satisfies $|z| \leq 1$ with equality for many (e.g., i, $e^{i\pi/4}$, and products thereof). This is a consequence of unitarity: every row of a unitary matrix has Euclidean norm 1, so each entry has magnitude smaller than or equal to 1.

Our finite field extension is important for exact computation because each amplitude is represented as a tuple of four rational numbers, enabling decidable comparison between expressions and exact probability calculations. In contrast, floating-point arithmetic represents numbers as $\pm 2^e \times (1.d_1 d_2 \ldots d_{52})_2$, which can only approximate numbers like $1/\sqrt{2}$ or $e^{i\pi/4}$. Even simple operations like $(1/\sqrt{2})^2 = 1/2$ give rounding error when computed in floating-point.

Arithmetic operations in $\mathbb{Q}(\sqrt{2}, i)$ reduce to arithmetic on the four rational coefficients. For example:

$$(a_1 + \frac{b_1}{\sqrt{2}} + c_1 i + \frac{d_1}{\sqrt{2}} i) + (a_2 + \frac{b_2}{\sqrt{2}} + c_2 i + \frac{d_2}{\sqrt{2}} i)$$
$$= (a_1 + a_2) + \frac{(b_1 + b_2)}{\sqrt{2}} + (c_1 + c_2)i + \frac{(d_1 + d_2)}{\sqrt{2}} i.$$

Multiplication is more involved but still requires only a constant number of rational multiplications and additions (using the equality $\sqrt{2}^2 = 2$ and $i^2 = -1$). Hence, a symbolic gate application to a state executes $O(d)$ field operations, where $d = 2^n$ is the state vector dimension, and each field operation costs $O(1)$ rational operations. This is the same asymptotic complexity as floating-point simulation, but with exact results.

The field $\mathbb{Q}(\sqrt{2}, i)$ has degree 4 over $\mathbb{Q}$. This is sufficient for all gates in our universal set $\{H, T, CNOT\}$. However, if one considers gates with entries involving higher radicals (e.g., $\sqrt[4]{2}$ or $\cos(\pi/8)$), a larger field extension would be required for an exact representation. But we have seen that the Solovay-Kitaev theorem provides arbitrary approximations within $\{H, T, CNOT\}$ while remaining within $\mathbb{Q}(\sqrt{2}, i)$. Thus, without loss of generality, we can approximate any unitary arbitrarily well while staying in this fixed finite field extension $\mathbb{Q}(\sqrt{2}, i)$.

4.2 seq-Measure-Once Quantum Finite Automata

In this thesis, we want to limit ourselves to the three gates in the universal quantum gate set: Hadamard, phase, and CNOT. This is so that we can limit ourselves to our field $\mathbb{Q}(\sqrt{2}, i)$ for exact simulation, and so this manner of simulation is easily translatable to implementation. Since this is a universal gate set, any quantum gate can be theoretically decomposed into a combination of these three gates to any desired accuracy $\varepsilon > 0$. To this end, we will define a model of MO-QFA which enables using this decomposition to simulate any QFA for further research: the seq-MO-QFA.

In this section we will first define this seq-MO-QFA, and then prove the equivalence relations with the standard MO-QFA.

4.2.1 Definition

We now define a variant of the MO-QFA where reading an input symbol triggers a quantum gate, optionally followed by seq-transitions.

Definition 4.1 (seq-MO-QFA)**.** *A uniform seq-Measure-Once Quantum Finite Automaton is a tuple* $M_seq = (Q, \Sigma, \{\text{seq}_a\}_{a\in\Sigma}, q_0, Q_{acc}, Q_{rej})$*, where:*

- Q *is a finite set of basis states (computational basis of* n *qubits);*
- Σ *is a finite input alphabet;*
- *For each* $a \in \Sigma$, $\text{seq}_a =< U_a^{(1)}, U_a^{(2)}, \ldots, U_a^{(m_a)} >$ *is a finite non-empty tuple of unitary matrices on* $\mathcal{H}_Q$*;*
- $q_0 \in Q$ *is the initial state;*
- $Q_{acc}, Q_{rej} \subseteq Q$ *are disjoint.*

The execution on input $w = a_1 a_2 \cdots a_k \in \Sigma^*$ proceeds as follows:

1. Initialize $|\psi_1^0\rangle = |q_0\rangle$.
2. For $i = 1$ to k:
   - **Consume** a_i: apply $G_{a_i}^{(1)}$ to $|\psi_i^0\rangle$ (this transition is *labeled by* the input symbol) and obtain the state $|\psi_i^1\rangle$;
   - **seq-transitions** For $j = 2$ to m_{a_i}:
     - Apply $G_{a_i}^{(j)}$ to $|\psi_i^{j-1}\rangle$ (these are the post seq-transitions associated with a_i, consuming no input) resulting into the state $|\psi_i^j\rangle$.
3. Measure the resulting final state $|\psi_k^{m_{a_k}}\rangle$ in the computational basis;
4. Accept if outcome $\in Q_{\text{acc}}$, reject if $\in Q_{\text{rej}}$.

Note that the first unitary in seq_a is the *input-consuming* transition; all subsequent unitaries are seq-steps. This models a quantum computer where reading a classical symbol triggers the first quantum operation, and the remaining operations complete the processing of that symbol internally, without further consuming of classical input. This means that the total unitary transformation associated with the symbol a is:

$$U_a = U_a^{(m_a)} \cdots U_a^{(2)} U_a^{(1)}.$$

4.2.2 Equivalence

We now prove that uniform seq-MO-QFAs are exactly as powerful as standard MO-QFAs, with trivial transformations in both directions.

From Standard MO-QFA to seq-MO-QFA Clearly, every standard MO-QFA is an seq-MO-QFA where the sequence of unitaries associated to each input symbol $a \in \Sigma$ is just the one element sequence consisting of the unitary U_a of the standard MO-QFS. More interesting is the case when we can decompose unitaries in finite products of basic unitaries, such as basic gates of a circuit.

Theorem 4.1. *Let $M = (Q, \Sigma, \{U_a\}_{a\in\Sigma}, q_0, Q_{acc}, Q_{rej})$ be a standard MO-QFA. Suppose for each $a \in \Sigma$, U_a can be expressed as a finite product of unitaries, for example from a finite gate set $\mathcal{G}$:*

$$U_a = U_a^{(m_a)} U_a^{(m_{a-1})} \cdots U_a^{(1)} .$$

Then there exists a uniform seq-MO-QFA M_ε such that $L(M_\varepsilon) = L(M)$.

Proof. Define M_ε with the same Q, Σ, q_0, Q_{acc}, Q_{rej}, and gate set $\mathcal{G}$. For each $a \in \Sigma$, set

$$\text{seq}_a =< U_a^{(1)}, U_a^{(2)}, \ldots, U_a^{(m_a)} > .$$

Consider any input $w = a_1 a_2 \cdots a_k$. In M_ε, the evolution is:

$$|\psi_{\text{final}}\rangle = \underbrace{U_{a_k}^{(m_{a_k})} \cdots U_{a_k}^{(2)} U_{a_k}^{(1)}}_{U_{a_k}} \cdots \underbrace{U_{a_1}^{(m_{a_1})} \cdots U_{a_1}^{(2)} U_{a_1}^{(1)}}_{U_{a_1}} |q_0\rangle = U_{a_k} \cdots U_{a_1} |q_0\rangle.$$

This is exactly the final state of M. Hence, the measurement outcome distribution, and therefore the acceptance probability, is identical for all inputs. Thus $L(M_\varepsilon) = L(M)$. □

From seq-MO-QFA to Standard MO-QFA

Theorem 4.2. *Let M_ε be a uniform seq-MO-QFA. Then there exists a standard MO-QFA M such that $L(M) = L(M_\varepsilon)$.*

Proof. Given $M_\varepsilon = (Q, \Sigma, \{\text{seq}_a\}_{a\in\Sigma}, q_0, Q_{\text{acc}}, Q_{\text{rej}})$ with $\text{seq}_a =< U_a^{(1)}, \ldots, U_a^{(m_a)} >$, define a standard MO-QFA $M = (Q, \Sigma, \{U_a\}_{a\in\Sigma}, q_0, Q_{\text{acc}}, Q_{\text{rej}})$ where

$$U_a = U_a^{(m_a)} \cdots U_a^{(2)} U_a^{(1)}.$$

For input $w = a_1 \cdots a_k$, the final state in M is:

$$|\psi_{\text{final}}\rangle = U_{a_k} \cdots U_{a_1} |q_0\rangle.$$

In M_ε, the final state is:

$$|\psi_{\text{final}}\rangle = \left(\prod_{j=2}^{m_{a_k}} U_{a_k}^{(j)}\right) U_{a_k}^{(1)} \cdots \left(\prod_{j=2}^{m_{a_1}} U_{a_1}^{(j)}\right) U_{a_1}^{(1)} |q_0\rangle.$$

These two expressions are identical because matrix multiplication is associative and the factors appear in exactly the same order from right to left (the first gate applied is the rightmost in the product). Thus, the two automata produce identical final states for every input, yielding the same acceptance probability. Hence $L(M_\varepsilon) = L(M)$. □

Our definition of seq-MO-QFA imposes that for each input symbol a, the sequence seq_a is **fixed and deterministic** (no branching), and that seq-steps form a **DAG** (directed acyclic graph) within the processing of each symbol. If we instead remove these restrictions and allow arbitrary ε-transitions (including nondeterministic choice, superposition of different ε-paths, and ε-cycles), several complications arise.

If U_ε has infinite order (e.g., a rotation by an irrational multiple of π), then arbitrarily long ε-paths produce distinct unitary operators, leading to a infinite set of possible states before reading the next input.

If the automaton may select one of several ε-transitions nondeterministically then the introduced branching requires summing over paths in the acceptance amplitude introducing possible violation of basic quantum computing laws such as the Euclidean norm of a state must alwys be 1. Also, the automaton could apply a superposition of different ε-paths, leading to interference effects.

In classical and probabilistic automata, and in some weighted automata, ε-elimination is straightforward [16] because:

- The ε-closure is computed by solving linear equations (for probabilities) or by semiring operations (for weighted automata).
- ε-cycles can be collapsed using fixed-point computations (e.g., by known convergence results of infinite sums or by calculating the Kleene star in the semiring).

Quantum automata lack a direct analogue because unitaries do not form a semiring with additive closure (the sum of two unitaries is not generally unitary), preserving unitarity restricts how we can combine ε-paths, and interference requires careful handling of amplitudes, preventing simple "closure" operations.

5 Symbolic Simulation Method

In this section we describe the core simulation method: representing quantum states symbolically and precomputing how each gate from our universal set transforms a generic symbolic state. The challenge is that the automaton may operate on n qubits (or, more generally, on a d-dimensional Hilbert space with $d = 2^n$), while our basic gates H, S, T, and $CNOT$ are defined for small numbers of qubits. We must therefore explain how to apply these gates to states of arbitrary dimension.

5.1 State Representation

Let the automaton have n qubits, so the Hilbert space is $\mathcal{H} = (\mathbb{C}^2)^{\otimes n}$ with dimension $d = 2^n$. We represent a quantum state as a column vector $|\psi\rangle$ of length d, where each entry is an element of the field extension $\mathbb{Q}(\sqrt{2}, i)$ defined in Section 2. Each entry has the canonical form:

$$\psi = a + \frac{1}{\sqrt{2}}b + ci + \frac{1}{\sqrt{2}}di, \qquad a, b, c, d \in \mathbb{Q}, i = \sqrt{-1}.$$

5.2 Preparing an MO-QFA for a Simulation

For an MO-QFA whose transition matrices are arbitrary unitaries, we first decompose each unitary U_a into a product of H, T, and $CNOT$ gates using the Solovay-Kitaev algorithm, and construct an seq-MO-QFA where seq_a is exactly that product.

If our decomposition achieves accuracy ε_a, the computed acceptance probability $P_{acc}^{\text{approx}}(w)$ satisfies

$$|P_{acc}^{\text{exact}}(w) - P_{acc}^{\text{approx}}(w)| \leq \sum_a \varepsilon_a \cdot n_a(w) \tag{1}$$

where $n_a(w)$ is the number of times a appears in w [15]. Since the error accumulates at most linearly in the input length, we can pre-select ε_a inversely proportional to the maximum input length considered, or simply simulate with a fixed small ε and accept a bounded error. For automata with unbounded input length our approximation approach gives a simulation that is *exact for the approximating automaton* and *provably close* to the original.

We have seen that clifford unitary can be expressed exactly as a finite product of H, T, and $CNOT$ gates. For such unitaries, our symbolic simulation yields *exact* results with no approximation error.

Corollary 5.1. *Let $M = (Q, \Sigma, \{U_a\}_{a\in\Sigma}, q_0, Q_{acc}, Q_{rej})$ be an MO-QFA where each U_a is an n-qubit Clifford unitary. Then there exists an seq-MO-QFA M' using only the gates $\{H, T, CNOT\}$ such that:*

- *M' has the same basis states Q,*
- *For each $a \in \Sigma$, the sequence seq_a has length at most $O(n^2/\log n)$ [1],*
- *$L(M') = L(M)$ exactly.*

For MO-QFAs that contain non-Clifford gates (e.g., arbitrary rotation gates), we cannot guarantee exact simulation using only $\{H, T, CNOT\}$ However, universality guarantees approximation: if for every $a \in \Sigma$ the transition unitary U_a is approximated by a sequence seq_a up to ε then the number the number of gates used per symbol of M_ε is $O(\log^c(1/\varepsilon))$, independent of the input length by the Solovay-Kitaev theorem.

The overhead in preparing an MO-QFA for simulation is multiplicative in the length of the sequences approximating the unitary of each action. Since the length grow only logarithmically in $1/\varepsilon$, high precision (e.g., $\varepsilon = 10^{-10}$) requires only a modest increase in the number of symbolic operations.

5.2.1 Extending Gates to the Full Dimension

A gate G defined on k qubits (with $k \leq n$) must be extended to act on the n-qubit system. The standard method is to take the tensor product with the identity operator on the remaining $n - k$ qubits. However, the specific extension depends on which qubits the gate acts on.

Definition 5.1 (Gate extension)**.** *Let G be a $2^k \times 2^k$ unitary matrix acting on qubits $q_1, q_2, \ldots, q_k$ (with $1 \leq q_1 < q_2 < \cdots < q_k \leq n$). The extension of G to the full n-qubit space is:*

$$G_{ext} = I_{2^{q_1-1}} \otimes G \otimes I_{2^{n-q_k}} \quad \textit{(simplified for adjacent qubits)}$$

More generally, if the qubits are not contiguous, one permutes the qubits so that the target qubits become adjacent, applies $G \otimes I$, then permutes back. In practice, we precompute the permutation and apply it symbolically.

For the purpose of deriving transformation rules, we assume without loss of generality that gates act on the *first* k qubits. Any other configuration can be reduced to this case by permuting the basis states (i.e., relabeling indices), which is a simple symbolic rearrangement of the vector entries.

Example 5.1 (Single-qubit gate on an n-qubit system)**.** *If $n \geq 2$ and a single-qubit gate G (e.g., H, S, or T) acts on qubit j, its extension is:*

$$G_{ext} = I_{2^{j-1}} \otimes G \otimes I_{2^{n-j}}.$$

In the computational basis ordered lexicographically ($|00\ldots0\rangle, |00\ldots1\rangle, \ldots, |11\ldots1\rangle$), applying G_{ext} to a state vector has the effect of applying G to each pair of basis states that differ only in the j-th qubit, leaving other qubits unchanged.

Example 5.2 (CNOT on an n-qubit system)**.** *The CNOT gate acts on two qubits: control c and target t. Its extension is:*

$$CNOT_{ext} = I_{2^{c-1}} \otimes \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \otimes I_{2^{n-\max(c,t)}}$$

with appropriate permutation if c and t are not adjacent. The effect is: for basis states where the control qubit is $|1\rangle$, flip the target qubit; otherwise leave unchanged.

5.3 Precomputed Transformation Rules

Rather than performing matrix multiplication from scratch for each automaton step, we precompute how each gate transforms a generic symbolic state. These rules are derived once and then applied repeatedly.

For a single-qubit gate acting on qubit j of an n-qubit system, the transformation affects only the amplitudes of basis states that differ in qubit j. A gate $G = \begin{bmatrix} g_{00} & g_{01} \\ g_{10} & g_{11} \end{bmatrix}$ acts on each component independently:

$$\begin{bmatrix} \psi_0' \\ \psi_1' \end{bmatrix} = \begin{bmatrix} g_{00} & g_{01} \\ g_{10} & g_{11} \end{bmatrix} \begin{bmatrix} \psi_0 \\ \psi_1 \end{bmatrix}$$

where ψ_0 and ψ_1 are the amplitudes of the qubit. Recall our specific gates:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \tag{2}$$

$$S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}, \tag{3}$$

$$T = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{\sqrt{2}} + i\frac{1}{\sqrt{2}} \end{bmatrix}. \tag{4}$$

Single-qubit transformation rules. Let $\alpha = \psi_0$ and $\beta = \psi_1$ be the two amplitudes in a pair, each expressed in the canonical form $\alpha = a + \frac{1}{\sqrt{2}}b + ci + \frac{1}{\sqrt{2}}di$ (and similarly for β with primed coefficients). Then:

$$\boxed{H:} \quad \begin{bmatrix} \alpha' \\ \beta' \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} \alpha + \beta \\ \alpha - \beta \end{bmatrix} \tag{5}$$

$$= \begin{bmatrix} \frac{1}{2}(b+b') + \frac{1}{\sqrt{2}}(a+a') + \frac{1}{2}(d+d')i + \frac{1}{\sqrt{2}}(c+c')i \\ \frac{1}{2}(b-b') + \frac{1}{\sqrt{2}}(a-a') + \frac{1}{2}(d-d')i + \frac{1}{\sqrt{2}}(c-c')i \end{bmatrix} \tag{6}$$

$$\boxed{S:} \quad \begin{bmatrix} \alpha' \\ \beta' \end{bmatrix} = \begin{bmatrix} \alpha \\ i\beta \end{bmatrix} = \begin{bmatrix} a + \frac{1}{\sqrt{2}}b + ci + \frac{1}{\sqrt{2}}di \\ -c' - \frac{1}{\sqrt{2}}d' + a'i + \frac{1}{\sqrt{2}}b'i \end{bmatrix} \tag{7}$$

$$\boxed{T:} \quad \begin{bmatrix} \alpha' \\ \beta' \end{bmatrix} = \begin{bmatrix} \alpha \\ e^{i\pi/4}\beta \end{bmatrix} = \begin{bmatrix} a + \frac{1}{\sqrt{2}}b + ci + \frac{1}{\sqrt{2}}di \\ \frac{1}{\sqrt{2}}a' - \frac{1}{2}b' - \frac{1}{\sqrt{2}}c' + \frac{1}{2}d' + \left(\frac{1}{\sqrt{2}}a' + \frac{1}{2}b' + \frac{1}{\sqrt{2}}c' + \frac{1}{2}d'\right)i \end{bmatrix} \tag{8}$$

The rule for T is derived from $e^{i\pi/4} = \frac{1}{\sqrt{2}} + i\frac{1}{\sqrt{2}}$. Note that $S = T^2$, so applying T twice yields the S rule.

Two-qubit gate: CNOT. For CNOT acting on control qubit c and target qubit t, we partition basis states into quadruplets where only qubits c and t vary. For each fixed assignment of the

remaining $n-2$ qubits, we have four basis states: $|00\rangle, |01\rangle, |10\rangle, |11\rangle$ (referring to the values of (q_c, q_t)). CNOT transforms them as:

$$CNOT : \begin{bmatrix} \psi_{00} \\ \psi_{01} \\ \psi_{10} \\ \psi_{11} \end{bmatrix} \mapsto \begin{bmatrix} \psi_{00} \\ \psi_{01} \\ \psi_{11} \\ \psi_{10} \end{bmatrix}.$$

That is, it swaps the amplitudes of $|10\rangle$ and $|11\rangle$. Symbolically, for each quadruplet with amplitudes $\alpha_{00}, \alpha_{01}, \alpha_{10}, \alpha_{11}$ (each in canonical form), the transformation is simply:

$$\alpha'_{00} = \alpha_{00}, \quad \alpha'_{01} = \alpha_{01}, \quad \alpha'_{10} = \alpha_{11}, \quad \alpha'_{11} = \alpha_{10}.$$

This is an exact symbolic permutation, requiring no arithmetic on the coefficients.

Multiple-qubit gates via tensor products. When a gate acts on a set of qubits that is not contiguous, or when we apply the same gate to multiple qubits (e.g., $H^{\otimes n}$), we can either:

1. Apply the single-qubit transformation rule independently to each qubit, in sequence, or
2. Precompute the combined transformation matrix as a tensor product and derive a single rule.

For $H^{\otimes n}$ (Hadamard on all n qubits), the transformation is the 2^n-dimensional Walsh-Hadamard transform, which can be computed efficiently using the recursive structure:

$$(H^{\otimes n})_{i,j} = \frac{1}{\sqrt{2^n}}(-1)^{i \cdot j}$$

where $i \cdot j$ is the bitwise dot product. Symbolically, this maps each amplitude to a signed sum of all 2^n amplitudes. While this is exponential, we only precompute it once per automaton size.

5.4 Simulation Algorithm

Given an seq-MO-QFA M (Definition 4.1) with n qubits and an input string $w = a_1 a_2 \dots a_k$, the simulation proceeds as follows:

Algorithm 1 Symbolic simulation of an seq-MO-QFA

Input: Automaton M, input string $w = a_1 \dots a_k$
Output: Exact symbolic expression for $P_{acc}(w)$
Initialize symbolic state vector $|\psi\rangle$ of length 2^n with $|\psi_{q_0}\rangle = 1$ (i.e., $a_{q_0} = 1$, all other coefficients zero)
for $i = 1$ to k **do**
  for $j = 1$ to m_{a_i} **do**
    Let G be the j-th gate in seq_{a_i}
    Extend G to act on n qubits (determine which qubits it acts on)
    Apply the precomputed transformation rule for G to $|\psi\rangle$
  end for
end for
Compute $P_{acc}(w) = \sum_{q \in Q_{acc}} |\langle q|\psi\rangle|^2$, where $|\langle q|\psi\rangle|^2 = (\text{Re}(\psi_q))^2 + (\text{Im}(\psi_q))^2$
Return $P_{acc}(w)$ as a symbolic expression in the initial coefficients (which are then substituted with 0/1 for the actual initial state)

The key observation is that the entire computation is performed symbolically. The final probability expression is a rational linear combination of terms of the form $a_p a_q$, $b_p b_q$, $c_p c_q$, $d_p d_q$, and cross-terms, all with coefficients in $\mathbb{Q}(\sqrt{2})$. When we substitute the concrete initial state (e.g., $a_{q_0} = 1$, all others 0), the expression collapses to a single algebraic number in $\mathbb{Q}(\sqrt{2}, i)$.

5.4.1 Example 1: Single-Qubit Hadamard on a 1-Qubit Automaton

Consider a 1-qubit automaton where we apply H to the initial state $|0\rangle$. Using the precomputed H rule with $\alpha = 1$ (i.e., $a = 1$, $b = c = d = 0$) and $\beta = 0$ (all coefficients zero):

$$H|0\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}.$$

This matches the known result.

5.4.2 Example 2: CNOT on a 3-Qubit System With Control Qubit 2 and Target Qubit 0

This example demonstrates gate extension to non-contiguous qubits. Consider a 3-qubit system with qubits indexed 0,1,2. We apply CNOT with control on qubit 2 and target on qubit 0. The basis states are 8 strings $|q_2 q_1 q_0\rangle$ (order chosen for convenience: most significant = qubit 2). The CNOT acts as: if $q_2 = 1$, flip q_0. The transformation rules for each quadruplet with fixed q_1 and varying (q_2, q_0) are:

Input basis	Output basis	Amplitude mapping
$\|0q_10\rangle$	$\|0q_10\rangle$	$\psi'_{000} = \psi_{000}$
$\|0q_11\rangle$	$\|0q_11\rangle$	$\psi'_{001} = \psi_{001}$
$\|1q_10\rangle$	$\|1q_11\rangle$	$\psi'_{010} = \psi_{011}$ (if we index as $\|q_2q_1q_0\rangle$)
$\|1q_11\rangle$	$\|1q_10\rangle$	$\psi'_{011} = \psi_{010}$

Thus, for each $q_1 \in \{0,1\}$, we swap the amplitudes of $|1q_10\rangle$ and $|1q_11\rangle$. Symbolically, if we start with an arbitrary state $|\psi\rangle = \sum_{x,y,z\in\{0,1\}} \psi_{xyz}|xyz\rangle$, after CNOT we have:

$$\psi'_{xyz} = \begin{cases} \psi_{xyz} & \text{if } x = 0 \\ \psi_{xy(1-z)} & \text{if } x = 1 \end{cases}$$

where x is qubit 2 (control), y is qubit 1 (unaffected), z is qubit 0 (target). This rule is applied symbolically by permuting indices in the coefficient array.

5.4.3 Example 3: Two-Qubit MO-QFA

Consider a two-qubit quantum automoton $M_3 = (Q_3, \Sigma_3, \{U_a\}_{a\in\Sigma}, q_0, Q_{\text{acc}}, Q_{\text{rej}})$, where:

$Q = \{q_0, q_1, q_2, q_3\}$;

$\Sigma = \{a, b, c\}$;

$$U_a = H\otimes H = \tfrac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}, U_b = S\otimes S = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & i & 0 & 0 \\ 0 & 0 & i & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}, U_c = CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$Q_{acc} = \{q_2, q_3\}$.

Let us take as input string $s = cbab$. This results in the final state $|\psi_{final}\rangle = U_cU_aU_bU_c|q_0\rangle = (S\otimes S)(H\otimes H)(S\otimes S)CNOT|q_0\rangle$.

Applying a double Hadamard gate to a two-qubit system results in:

$$(H\otimes H)|\psi\rangle = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}\begin{bmatrix} a_1 + \frac{1}{\sqrt{2}}b_x + c_1i + \frac{1}{\sqrt{2}}d_1i \\ a_2 + \frac{1}{\sqrt{2}}b_2 + c_2i + \frac{1}{\sqrt{2}}d_2i \\ a_3 + \frac{1}{\sqrt{2}}b_3 + c_3i + \frac{1}{\sqrt{2}}d_3i \\ a_4 + \frac{1}{\sqrt{2}}b_4 + c_4i + \frac{1}{\sqrt{2}}d_4i \end{bmatrix} =$$

$$\begin{bmatrix} \frac{1}{2}(a_1+a_2+a_3+a_4) + \frac{1}{2\sqrt{2}}(b_1+b_2+b_3+b_4) + \frac{1}{2}(c_1+c_2+c_3+c_4)i + \frac{1}{2\sqrt{2}}(d_1+d_2+d_3+d_4)i \\ \frac{1}{2}(a_1-a_2+a_3-a_4) + \frac{1}{2\sqrt{2}}(b_1-b_2+b_3-b_4) + \frac{1}{2}(c_1-c_2+c_3-c_4)i + \frac{1}{2\sqrt{2}}(d_1-d_2+d_3-d_4)i \\ \frac{1}{2}(a_1+a_2-a_3-a_4) + \frac{1}{2\sqrt{2}}(b_1+b_2-b_3-b_4) + \frac{1}{2}(c_1+c_2-c_3-c_4)i + \frac{1}{2\sqrt{2}}(d_1+d_2-d_3-d_4)i \\ \frac{1}{2}(a_1-a_2-a_3+a_4) + \frac{1}{2\sqrt{2}}(b_1-b_2-b_3+b_4) + \frac{1}{2}(c_1-c_2-c_3+c_4)i + \frac{1}{2\sqrt{2}}(d_1-d_2-d_3+d_4)i \end{bmatrix} \tag{9}$$

Applying a double phase gate to a two-qubit system results in:

$$(S\otimes S)|\psi\rangle = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & i & 0 & 0 \\ 0 & 0 & i & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}\begin{bmatrix} a_1 + \frac{1}{\sqrt{2}}b_1 + c_1i + \frac{1}{\sqrt{2}}d_1i \\ a_2 + \frac{1}{\sqrt{2}}b_2 + c_2i + \frac{1}{\sqrt{2}}d_2i \\ a_3 + \frac{1}{\sqrt{2}}b_3 + c_3i + \frac{1}{\sqrt{2}}d_3i \\ a_4 + \frac{1}{\sqrt{2}}b_4 + c_4i + \frac{1}{\sqrt{2}}d_4i \end{bmatrix} = \begin{bmatrix} a_1 + \frac{1}{\sqrt{2}}b_1 + c_1i + \frac{1}{\sqrt{2}}d_1i \\ -c_2 - \frac{1}{\sqrt{2}}d_2 + a_2i + \frac{1}{\sqrt{2}}b_2i \\ -c_3 - \frac{1}{\sqrt{2}}d_3 + a_3i + \frac{1}{\sqrt{2}}b_3i \\ -a_4 - \frac{1}{\sqrt{2}}b_4 - c_4i - \frac{1}{\sqrt{2}}d_4i \end{bmatrix} \tag{10}$$

To obtain the state $|\psi_1\rangle$ after the first transition, we simply use the precomputed CNOT rule equation where amplitudes are swapped:

$$CNOT_{0\to1}: \begin{bmatrix}\psi_{00}\\ \psi_{01}\\ \psi_{10}\\ \psi_{11}\end{bmatrix} \mapsto \begin{bmatrix}\psi_{00}\\ \psi_{01}\\ \psi_{11}\\ \psi_{10}\end{bmatrix}.$$

Then we apply equation 10 to this new symbolic state $|\psi_1\rangle$ to obtain the next state $|\psi_2\rangle$ for this input string:

$$|\psi_2\rangle = (S\otimes S)|\psi_1\rangle = \begin{bmatrix} a_1 + \frac{1}{\sqrt{2}}b_1 + c_1 i + \frac{1}{\sqrt{2}}d_1 i \\ -c_2 - \frac{1}{\sqrt{2}}d_2 + a_2 i + \frac{1}{\sqrt{2}}b_2 i \\ -c_4 - \frac{1}{\sqrt{2}}d_4 + a_4 i + \frac{1}{\sqrt{2}}b_4 i \\ -a_3 - \frac{1}{\sqrt{2}}b_3 - c_3 i - \frac{1}{\sqrt{2}}d_3 i \end{bmatrix}$$

Then we apply equation 9 to this new symbolic state $|\psi_2\rangle$ to obtain the next state $|\psi_3\rangle$ for this input string:

$$|\psi_3\rangle = (H\otimes H)|\psi_2\rangle =$$

$$\begin{bmatrix} \frac{1}{2}(a_1 - c_2 - c_4 - a_3) + \frac{1}{2\sqrt{2}}(b_1 - d_2 - d_4 - b_3) + \frac{1}{2}(c_1 + a_2 + a_4 - c_3)i + \frac{1}{2\sqrt{2}}(d_1 + b_2 + b_4 - d_3)i \\ \frac{1}{2}(a_1 + c_2 - c_4 + a_3) + \frac{1}{2\sqrt{2}}(b_1 + d_2 - d_4 + b_3) + \frac{1}{2}(c_1 - a_2 + a_4 + c_3)i + \frac{1}{2\sqrt{2}}(d_1 - b_2 + b_4 + d_3)i \\ \frac{1}{2}(a_1 - c_2 + c_4 + a_3) + \frac{1}{2\sqrt{2}}(b_1 - d_2 + d_4 + b_3) + \frac{1}{2}(c_1 + a_2 - a_4 + c_3)i + \frac{1}{2\sqrt{2}}(d_1 + b_2 - b_4 + d_3)i \\ \frac{1}{2}(a_1 + c_2 + c_4 - a_3) + \frac{1}{2\sqrt{2}}(b_1 + d_2 + d_4 - b_3) + \frac{1}{2}(c_1 - a_2 - a_4 - c_3)i + \frac{1}{2\sqrt{2}}(d_1 - b_2 - b_4 - d_3)i \end{bmatrix}$$

Then we apply equation 10 to this new symbolic state $|\psi_3\rangle$ to obtain the final state $|\psi_{final}\rangle$ for this input string:

$$|\psi_{final}\rangle = (S\otimes S)|\psi_3\rangle =$$

$$\begin{bmatrix} \frac{1}{2}(a_1 - c_2 - c_4 - a_3) + \frac{1}{2\sqrt{2}}(b_1 - d_2 - d_4 - b_3) + \frac{1}{2}(c_1 + a_2 + a_4 - c_3)i + \frac{1}{2\sqrt{2}}(d_1 + b_2 + b_4 - d_3)i \\ -\frac{1}{2}(c_1 - a_2 + a_4 + c_3) - \frac{1}{2\sqrt{2}}(d_1 - b_2 + b_4 + d_3) + \frac{1}{2}(a_1 + c_2 - c_4 + a_3)i + \frac{1}{2\sqrt{2}}(b_1 + d_2 - d_4 + b_3)i \\ -\frac{1}{2}(c_1 + a_2 - a_4 + c_3) - \frac{1}{2\sqrt{2}}(d_1 + b_2 - b_4 + d_3) + \frac{1}{2}(a_1 - c_2 + c_4 + a_3)i + \frac{1}{2\sqrt{2}}(b_1 - d_2 + d_4 + b_3)i \\ -\frac{1}{2}(a_1 + c_2 + c_4 - a_3) - \frac{1}{2\sqrt{2}}(b_1 + d_2 + d_4 - b_3) - \frac{1}{2}(c_1 - a_2 - a_4 - c_3)i - \frac{1}{2\sqrt{2}}(d_1 - b_2 - b_4 - d_3)i \end{bmatrix}$$

To obtain the probability of acceptance, we calculate:

$$P_{\text{acc}}(w) = \textstyle\sum_{q\in Q_{\text{acc}}} |\langle q|\psi_{\text{final}}\rangle|^2 = |\langle q_2|\psi_{\text{final}}\rangle|^2 + |\langle q_3|\psi_{\text{final}}\rangle|^2 =$$

$$\begin{aligned} &|-\tfrac{1}{2}(c_1 + a_2 - a_4 + c_3) - \tfrac{1}{2\sqrt{2}}(d_1 + b_2 - b_4 + d_3) + \tfrac{1}{2}(a_1 - c_2 + c_4 + a_3)i + \tfrac{1}{2\sqrt{2}}(b_1 - d_2 + d_4 + b_3)i|^2 + \\ &\quad |-\tfrac{1}{2}(a_1 + c_2 + c_4 - a_3) - \tfrac{1}{2\sqrt{2}}(b_1 + d_2 + d_4 - b_3) - \tfrac{1}{2}(c_1 - a_2 - a_4 - c_3)i - \tfrac{1}{2\sqrt{2}}(d_1 - b_2 - b_4 - d_3)i|^2 \end{aligned}$$

If we apply this symbolic result to the initial state $|q_0\rangle$, where $a_1 = 1$ and all other coefficients equal 0, we obtain $P(s) = \frac{1}{2}$.

5.4.4 Example 4: Two-Qubit seq-MO-QFA

Consider a two-qubit quantum automoton $M_3 = (Q_3, \Sigma_3, \{U_a\}_{a\in\Sigma}, q_0, Q_{\text{acc}}, Q_{\text{rej}})$, where:

$Q = \{q_0, q_1, q_2, q_3\}$;

$\Sigma = \{a, b\}$;

$$U_a = SWAP = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, U_b = H \otimes I = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix}$$

$Q_{acc} = \{q_2, q_3\}$.

Using the Solovay-Kitaev algorithm, we decompose the SWAP gate, which is a Clifford gate, into the product of three CNOTs as done in section 2.

Then we construct a seq-MO-QFA where seq_a is this product:

$$\text{seq}_a = CNOT_{0\to1} \cdot CNOT_{1\to0} \cdot CNOT_{0\to1},$$

seq_b remains the $(H \otimes I)$, which is a Hadamard gate acting on qubit 1, extended to the two-qubit system.

Applying the tensor product $(H \otimes I)$ to a two-qubit system results in:

$$\begin{aligned}(H \otimes I)|\psi\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} a_1 + \frac{1}{\sqrt{2}}b_x + c_1 i + \frac{1}{\sqrt{2}}d_1 i \\ a_2 + \frac{1}{\sqrt{2}}b_2 + c_2 i + \frac{1}{\sqrt{2}}d_2 i \\ a_3 + \frac{1}{\sqrt{2}}b_3 + c_3 i + \frac{1}{\sqrt{2}}d_3 i \\ a_4 + \frac{1}{\sqrt{2}}b_4 + c_4 i + \frac{1}{\sqrt{2}}d_4 i \end{bmatrix} = \\ \begin{bmatrix} \frac{1}{2}(b_1 + b_3) + \frac{1}{\sqrt{2}}(a_1 + a_3) + \frac{1}{2}(d_1 + d_3)i + \frac{1}{\sqrt{2}}(c_1 + c_3)i \\ \frac{1}{2}(b_2 + b_4) + \frac{1}{\sqrt{2}}(a_2 + a_4) + \frac{1}{2}(d_2 + d_4)i + \frac{1}{\sqrt{2}}(c_2 + c_4)i \\ \frac{1}{2}(b_1 - b_3) + \frac{1}{\sqrt{2}}(a_1 - a_3) + \frac{1}{2}(d_1 - d_3)i + \frac{1}{\sqrt{2}}(c_1 - c_3)i \\ \frac{1}{2}(b_2 - b_4) + \frac{1}{\sqrt{2}}(a_2 - a_4) + \frac{1}{2}(d_2 - d_4)i + \frac{1}{\sqrt{2}}(c_2 - c_4)i \end{bmatrix}\end{aligned} \tag{11}$$

Let us take as input string $s = ab$. This results in the final state $|\psi_{final}\rangle = \text{seq}_b \cdot \text{seq}_a \cdot |q_0\rangle$.

To first apply seq_a, we first consume the input character a, and apply the first matrix in the sequence to $|\psi_0\rangle$ to obtain state $|\psi_1^1\rangle$, where the amplitudes of the second qubit are switched:

$$CNOT_{0\to1} : \begin{bmatrix} \psi_{00} \\ \psi_{01} \\ \psi_{10} \\ \psi_{11} \end{bmatrix} \mapsto \begin{bmatrix} \psi_{00} \\ \psi_{01} \\ \psi_{11} \\ \psi_{10} \end{bmatrix}.$$

Then we apply the second matrix in the sequence, consuming no input, resulting in the state $|\psi_1^2\rangle$:

$$CNOT_{1\to 0} : \begin{bmatrix}\psi_{00}\\ \psi_{01}\\ \psi_{11}\\ \psi_{10}\end{bmatrix} \mapsto \begin{bmatrix}\psi_{00}\\ \psi_{10}\\ \psi_{11}\\ \psi_{01}\end{bmatrix}.$$

And finally, the third matrix in the sequence, completing the simulation of the original SWAP gate, to obtain $|\psi_1^3\rangle$:

$$CNOT_{0\to 1} : \begin{bmatrix}\psi_{00}\\ \psi_{10}\\ \psi_{11}\\ \psi_{01}\end{bmatrix} \mapsto \begin{bmatrix}\psi_{00}\\ \psi_{10}\\ \psi_{01}\\ \psi_{11}\end{bmatrix}.$$

Then applying seq_b, which only consists of $(H \otimes I)$, by applying the precomputed equation, we obtain the final state $|\psi_{final}\rangle$:

$$|\psi_{final}\rangle = (H \otimes I)|\psi_1^3\rangle = \begin{bmatrix} \frac{1}{2}(b_1+b_2)+\frac{1}{\sqrt{2}}(a_1+a_2)+\frac{1}{2}(d_1+d_2)i+\frac{1}{\sqrt{2}}(c_1+c_2)i \\ \frac{1}{2}(b_3+b_4)+\frac{1}{\sqrt{2}}(a_3+a_4)+\frac{1}{2}(d_3+d_4)i+\frac{1}{\sqrt{2}}(c_3+c_4)i \\ \frac{1}{2}(b_1-b_2)+\frac{1}{\sqrt{2}}(a_1-a_2)+\frac{1}{2}(d_1-d_2)i+\frac{1}{\sqrt{2}}(c_1-c_2)i \\ \frac{1}{2}(b_3-b_4)+\frac{1}{\sqrt{2}}(a_3-a_4)+\frac{1}{2}(d_3-d_4)i+\frac{1}{\sqrt{2}}(c_3-c_4)i \end{bmatrix}.$$

To obtain the probability of acceptance, we calculate:

$$P_{\text{acc}}(w) = \sum_{q\in Q_{\text{acc}}} |\langle q|\psi_{\text{final}}\rangle|^2 = |\langle q_2|\psi_{\text{final}}\rangle|^2 + |\langle q_3|\psi_{\text{final}}\rangle|^2 =$$

$$|\tfrac{1}{2}(b_1-b_2)+\tfrac{1}{\sqrt{2}}(a_1-a_2)+\tfrac{1}{2}(d_1-d_2)i+\tfrac{1}{\sqrt{2}}(c_1-c_2)i|^2 + |\tfrac{1}{2}(b_3-b_4)+\tfrac{1}{\sqrt{2}}(a_3-a_4)+\tfrac{1}{2}(d_3-d_4)i + \tfrac{1}{\sqrt{2}}(c_3-c_4)i|^2$$

If we apply this symbolic result to the initial state $|q_0\rangle$, where $a_1 = 1$ and all other coefficients equal 0, we obtain $P(s) = \frac{1}{2}$.

As any decomposition in the original MO-QFA was of a Clifford unitary, this result is *exact*.

5.5 Complexity of the Method

Let $dim = 2^n$ be the dimension of the Hilbert space. Each symbolic state is represented by $4dim$ rational coefficients (the a, b, c, d for each of dim entries). Applying a single-qubit gate to a specific qubit requires iterating over dim independent tuples, performing a constant number of arithmetic operations on rational numbers. Hence, time per gate is $O(dim) = O(2^n)$. This is the same asymptotic complexity as classical simulation using floating-point vectors.

The advantage over floating-point is *exactness* (no rounding error). The memory usage is $O(dim)$ rational coefficients, each represented as a quadruple of arbitrary-precision rationals.

For automata with m_a gates per symbol and input length k, the total time is $O(k \cdot (\sum_a m_a) \cdot 2^n)$. Since m_a is constant per automaton, the simulation is linear in input length and exponential in the number of qubits.

6 Conclusions and Further Research

In this thesis we have devised a way of performing symbolic simulation of a measure-once quantum finite-state automaton. This is an exact manner of simulation, avoiding any approximation errors introduced by complex numbers, by limiting ourselves to a field extended from rational numbers. Furthermore, by limiting ourselves to the quantum universal gate set (Hadamard, phase, CNOT), the symbolic state after each transition is very simple to derive using precomputed equations. To this end we have devised our own variant of the MO-QFA: the seq-MO-QFA.

The only error possibly contained in our method is an approximation error introduced by non-Clifford gates, which cannot be exactly decomposed into the universal gate set, merely approximated.

This thesis could function as a foundation for further research. If one were to devise a structural way of decomposing any quantum gate into a product of the universal gates, any quantum automaton could be simulated in a simple way in our exact symbolic way using our defined seq-MO-QFA. Furthermore, as our simulations were done manually, we were limited to simpler automata due to time constraints. Implementation would allow for simulation of much more complex automata and a more comprehensive analysis of the method. Naturally, one could look at if this method could be extended to other variants of quantum automata, including measure many automata.

Beyond simulation itself, the exact symbolic representation developed in this thesis opens the door to automated *analysis* of quantum finite automata. Because $P_{acc}(w)$ is computed as a closed-form expression over $\mathbb{Q}(\sqrt{2}, i)$ rather than a floating-point approximation, equality of acceptance probabilities becomes a decidable comparison of rational coefficients rather than an approximate numerical test. This suggests that *equivalence checking* between two MO-QFAs could be automated: since $P_{acc}(w)$ is linear in the initial state, equivalence on all of Σ^* should reduce, as in the classical theory of weighted automata [16, 17], to checking equality on a finite basis of the reachable state space, rather than on every individual input string.

Similarly, the error bound of Equation 1 already provides a building block for deciding ε-*inclusion* between an automaton and an approximated version of itself; a natural generalization would compute, or bound, $\max_w |P_1(w) - P_2(w)|$ for two arbitrary automata rather than only between an automaton and its own approximation.

Finally, restricting to the finite field $\mathbb{Q}(\sqrt{2}, i)$ is also the field used in exact Clifford+T *gate synthesis* [1], suggesting that the same representation could support synthesizing a seq-MO-QFA exactly realizing a target unitary or acceptance behaviour, rather than only simulating one. We leave these three directions (equivalence, bounded-error inclusion, and synthesis) as an open problem for future work.

References

[1] Scott Aaronson and Daniel Gottesman. Improved simulation of stabilizer circuits. *Phys. Rev. A*, 70:052328, Nov 2004.

[2] Leonard M. Adleman, Jonathan DeMarrais, and Ming-Deh A. Huang. Quantum computability. *SIAM J. Comput.*, 26:1524–1540, 1997.

[3] Diego Alonso, Pedro Sánchez, and Francisco Sánchez-Rubio. Engineering the development of quantum programs: Application to the boolean satisfiability problem. *Advances in Engineering Software*, 173:103216, 2022.

[4] Adriano Barenco, Charles H. Bennett, Richard Cleve, David P. DiVincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John A. Smolin, and Harald Weinfurter. Elementary gates for quantum computation. *Phys. Rev. A*, 52:3457–3467, Nov 1995.

[5] Stephen Barnett. *Matrix Methods for Engineers and Scientists*. McGraw-Hill, 1979.

[6] Paul Benioff. Quantum mechanical hamiltonian models of turing machines. *Journal of Statistical Physics*, 29:515–546, 11 1982.

[7] Zeyu Chen and Junde Wu. On the simulation cost of quantum finite automata, 2026.

[8] Christopher M. Dawson and Michael A. Nielsen. *The Solovay-Kitaev algorithm*. Quantum Information & Computation, 2006.

[9] Stanley Gudder. Quantum automata: An overview. *International Journal of Theoretical Physics 38*, 1999.

[10] Atilla Kondacs and John Watrous. On the power of quantum finite state automata. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, page 66, USA, 1997. IEEE Computer Society.

[11] Gustaw Lippa, Krzysztof Makieła, and Marcin Kuta. Simulations of quantum finite automata. *Computational Science – ICCS 2020*, 12142:441 – 450, 2020.

[12] John C. Martin. *Introduction to Languages and the Theory of Computation*. McGraw-Hill, 2011.

[13] Cristopher Moore and James P. Crutchfield. Quantum automata and quantum grammars. *Theoretical Computer Science*, 237(1):275–306, 2000.

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

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

[16] A. Paz. *Introduction to Probabilistic Automata*. Computer science and applied mathematics. Academic Press, 1971.

[17] Jacques Sakarovitch. *Elements of Automata Theory*. Cambridge University Press, 2009.

[18] Andrew Steane. Quantum computing. *Reports on Progress in Physics*, 1998.

[19] Farrokh Vatan and Colin Williams. Optimal quantum circuits for general two-qubit gates. *Physical Review A*, 69(3), March 2004.

[20] Joachim Von Zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. Cambridge University Press, 1999.