Quantum Information & Computation
ISSN 1533-7146 Volume 25 (2025) pp. 57–70
©Cerebration Science Publishing

Article

Quantum Finite Automaton Using Ternary Rotation Quantum Gates and Chrestenson Family Quantum Gates

Yuchen Huang *, Marek Perkowski , Xiaoyu Song and John M. Acken

Department of Electrical and Computer Engineering, Portland State University, Portland, OR 97201, USA; h8mp@pdx.edu (M.P.); songx@pdx.edu (X.S.); acken@pdx.edu (J.M.A.)
* Correspondence author: yuchen@pdx.edu

Received date: 16 November 2024; Accepted date: 6 February 2025; Published online: 20 February 2025

Abstract: Quantum automata can solve certain problems with a smaller state space than classical automata. We developed a quantum finite automaton using ternary rotation quantum gates and the Chrestenson family of ternary quantum gates. The main idea of this paper is to show how to combine rotation ternary quantum circuit-based Quantum Finite Automaton and quantum reversible circuit-based Deterministic Finite Automaton to build a more powerful machine. The combined machine can enable more complex language and pattern recognition. The developed quantum finite automaton and resulting combined machine can be used for robotics applications such as language, gesture, and motion recognition.

Keywords: quantum finite automaton; ternary; quantum gates; rotation quantum gates; Chrestenson family quantum gates

1. Introduction

Quantum information science improves problem-solving using quantum effects. Quantum computer achieves exponential speedup in solving certain factoring problems [1] and complex system simulations [2–4]. Significant efforts were made to develop small-scale quantum computers across various physical systems [5–11]. Quantum algorithms on prototype machines showed benefits in solving specific problems [12–16]. Substantial progress has been made in the last 20 years in quantum algorithm and quantum technology [17]. However, progress in quantum automata was not significant, possibly due to the fact that quantum memory is still not industrially viable.

Quantum concepts can improve the capabilities of Finite Automaton (FA). FA is used to model mathematics, AI, games, and linguistics [18,19]. Quantum Finite Automaton (QFA) was invented independently by Kondacs and Watrious [20] and by Moore and Crutchfield [21]. Theoretical predictions suggest QFA is more efficient than classical FA. Another model called Quantum Finite State Machines (QFSM) was introduced in [22].

In [23], an optical QFA was developed for a proof-of-principle demonstration to solve the promise problem of determining whether the length of an input string can be divided by a prime number P with no remainder or with a remainder R using three-dimensional quantum states. It showed the resulting quantum automaton can solve such a problem using a state space with only three orthonormal states, whereas a classical automaton requires no less than P states. The circuit in [23] uses ternary quantum logic rather than standard binary quantum logic. There are two important contributions of paper [23]. First, it introduced multi-valued quantum logic for a practical application in quantum automata. Second, this machine was built in a real quantum optical technology in contrast to most published papers about quantum automata that are of a purely theoretical nature. In our opinion, [23] starts a new direction in the study and practical application of quantum automata. However, [23] only showed QFA's advantage over classical Deterministic Finite Automaton (DFA) for solving a particular promise problem using a particular ternary unitary operation. Therefore its application is quite limited.

Our paper expands the work in [23] by developing a family of ternary rotation-based QFA and the Chrestenson family of ternary quantum gates [24] based QFA. The main idea of our paper is that we combine rotation-based machines and quantum reversible circuit machines to accept more complex languages. We further extend QFA applications to more practical scenarios such as pattern recognition and robotics applications.

DOI: 10.2478/qic-2025-0003

The main contributions of our paper are: (1) introducing two new gates similar to the Chrestenson gate such that these gates enable the creation of arbitrary ternary permutative gates; (2) showing methods to realize many types of quantum automata using these gates; (3) introducing the concept of hybrid automata which combine traditional quantum automata with quantum realizations of DFA; and (4) demonstrating that such automata can find application in robotics.

Section 2 gives an overview of DFA versus QFA and QFA's benefits over DFA. Section 3 explains the development of QFA using the Chrestenson family of ternary quantum gates. Section 4 shows how to combine a rotation-based ternary quantum circuit machine and a quantum reversible circuit machine to build a more powerful machine. Section 5 talks about extending QFA applications to more practical scenarios. Section 6 outlines new ideas of Quantum Finite State Machine with Data Path (QFSMD) to enable more powerful quantum circuits by introducing truly quantum gates such as Hadamard. Section 7 concludes and summarizes our paper.

2. Overview of DFA and QFA

2.1. A Typical Deterministic Finite Automaton (DFA)

A Deterministic Finite Automaton (DFA) is a finite state machine that accepts or rejects input strings of symbols. In automata theory, a finite state machine is called a Deterministic Finite Automaton if each of its transitions is uniquely determined by its source state and input symbol; and reading an input symbol is required for each state transition [25].

Figure 1 is the state diagram of a typical example of DFA [23]. As shown in Figure 1, a DFA has a starting state and a set of accepting states. Starting state is where computations begin, which is denoted graphically by an arrow coming in from nowhere. Accepting states are denoted graphically by a double circle. In this example automaton, there are two states S_0 and S_1, which are denoted graphically by circles. The automaton takes a finite but arbitrarily long sequence of 0s and 1s as input. For each state, there is a transition arrow leading out to the next state for both 0 and 1. Upon reading an input symbol, a DFA jumps deterministically from one state to another by following the transition arrow. For example, if the automaton is currently in state S_0 and the current input symbol is 1, then it deterministically jumps to state S_1. If the automaton ends at an accepting state after reading the last symbol of the input string, this string of symbols is considered as being accepted by the DFA; or else, it is considered as being rejected by the DFA. DFAs are often used to solve decision problems, which are defined as computational problems where the answer for every instance is either Yes or No. Figure 1 also shows examples of accepted strings such as 000, 111110, and 10, and examples of not-accepted strings such as 11, 111, and 011. This DFA can be realized in a classical Boolean logic circuit with Boolean gates and D Flip-Flops. However, this machine can also be built as a quantum automaton with quantum realization of transition and output functions and measurements followed by classical D Flip-Flops [26]. This is illustrated in Figure 2. [26] gives a complete explanation on how to realize this type of quantum automata with classical memory in minimized quantum circuit.

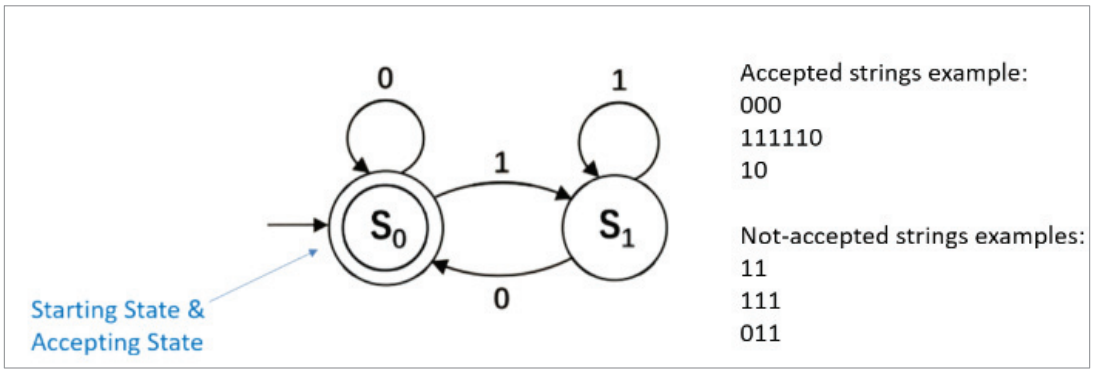

Figure 1. The State Diagram of a Typical DFA.

Figure 2 shows a quantum automaton with quantum realization of the DFA of Figure 1. The state transitions are:

$$S_0 .0 \rightarrow S_0 \quad S_0 .1 \rightarrow S_1 \quad S_1 .1 \rightarrow S_1 \quad S_1 .0 \rightarrow S_0$$

The state transition function is shown below.

$$S_0' = S_0.(\sim\text{input}) \oplus S_1.(\sim\text{input})$$
$$S_1' = S_0.(\text{input}) \oplus S_1.(\text{input})$$

S_0' and S_1' are the next states. The outputs are obtained by direct measurement of internal states, so this is a special case of the Moore machine.

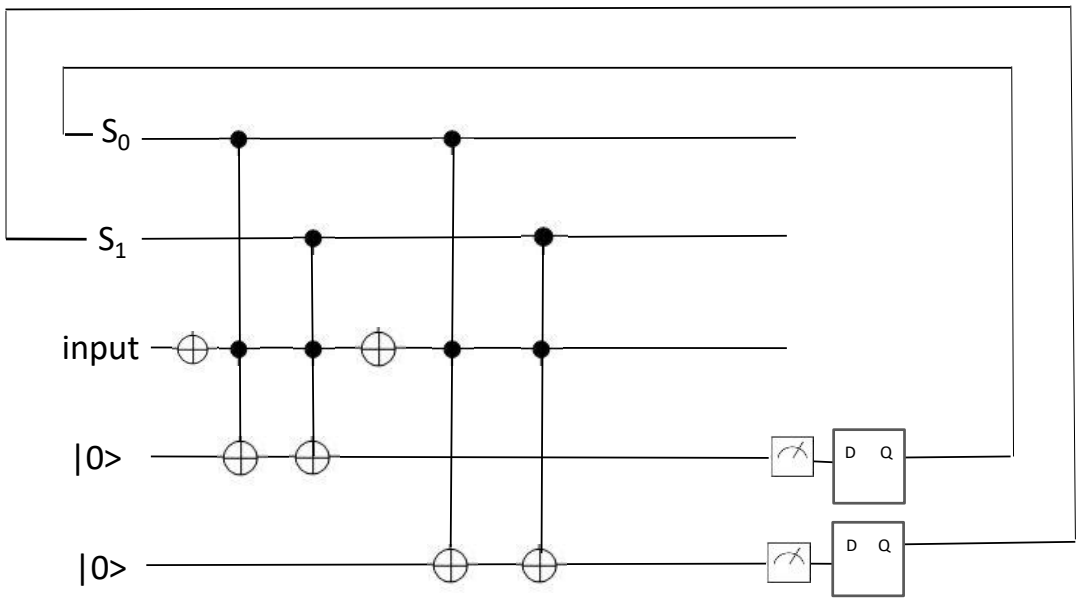

Figure 2. Quantum Automaton with Quantum Realization of DFA of Figure 1.

2.2. A General QFA

Figure 3 shows a general Quantum Finite Automaton (QFA). QFA is the quantum version of DFA and it is realized by changing the basic elements of DFA to quantum components. The upper part of Figure 3 represents the input string that goes to the controller of a quantum state machine represented in the lower part of Figure 3. The controller is a classical computer. This controller of the hybrid system creates pulses to initialize the state $|\psi_0\rangle$ in the quantum circuit presented in the lower part. Next, the controller creates pulses corresponding to symbols of the input string. Finally, the controller creates measurement pulses at the end of the input string. As shown in Figure 3, inside a QFA circuit sits a quantum state $|\psi\rangle$ with m orthonormal basis states $|0\rangle, |1\rangle, \ldots, |m-2\rangle, |m-1\rangle$. One or several of the m basis states are designated as accepting states. Before the computation starts, the QFA rests at a starting state $|\psi_0\rangle$. The computation of a QFA with an input string $\sigma_1\sigma_2\ldots\sigma_{n-1}\sigma_n$ (σ_i is the ith symbol of the string) works as follows: The QFA reads in the input string from left to right, symbol by symbol. Upon reading a symbol σ_i, a corresponding unitary operator $U_{\sigma i}$ is applied to the state. After reading the last symbol of the input string, the final state of the QFA would be $|\psi_n\rangle = U\sigma_n U\sigma_{n-1}\ldots U\sigma_2 U\sigma_1|\psi_0\rangle$. A projective measurement in the basis of $|0\rangle, |1\rangle, \ldots, |m-2\rangle, |m-1\rangle$ is then applied on $|\psi_n\rangle$ and the state would be projected to one of the m basis states. If the state is projected to an accepting state, the string is regarded as being accepted by the QFA; otherwise, the string is rejected [23]. The General QFA from Figure 3 is basically the same as a standard quantum computer. In [23], all the unitary operators $U_{\sigma i}$ are the same. Observe that m can be an arbitrarily large number. The input string of symbols is received by the classical computer. For every input symbol that the classical computer (defined above as a controller) receives, it creates a respective quantum rotation operation in the quantum computer that the classical computer controls. Observe that the quantum automaton in Figure 3 uses quantum states as intermediate states without measurements, while the automaton from Figure 2 uses measurements to create a classical state corresponding to a current quantum state for every transition, which means for every symbol of the input string. The idea of our paper is to combine these two different models of quantum automata into one powerful hybrid model.

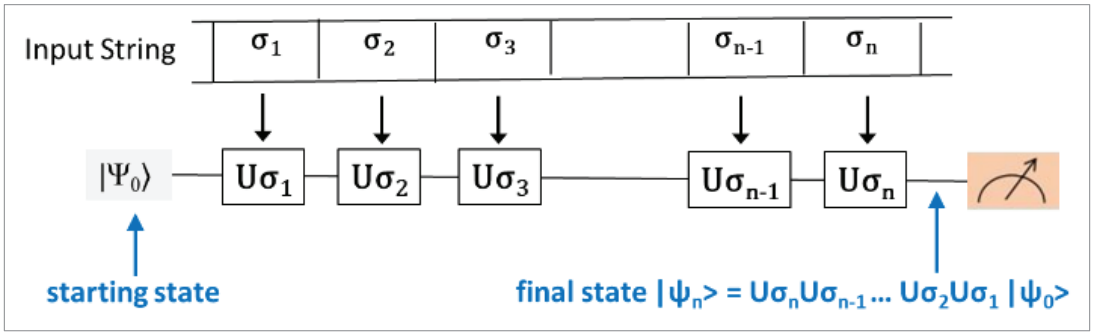

Figure 3. The State Diagram of a General QFA.

2.3. Quantum Benefits of QFA over DFA

It has been shown that Quantum Finite Automaton (QFA) can solve certain promise problems with a significantly smaller state space than a classical Deterministic Finite Automaton (DFA) [23]. An optical quantum automaton was demonstrated to determine if an input string length is divisible by a prime number P or leaves a remainder R in [23]. The QFA in [23] assumes $|0\rangle$ as the only accepting state. After the projective measurement on the final state, the QFA will deterministically obtain classical state 0 and accept the input string when its length is an integer multiple of a prime number P; the QFA will probabilistically obtain either classical state 1 or 2 and reject the input string when its length is not an integer multiple of a prime number P. The QFA from [23] with ternary encoding uses only three orthonormal states, whereas a standard DFA with binary encoding of states requires at least P states [27], showcasing the advantages of QFA and its potential for more complex applications. However, [23] only showed QFA's advantage to DFA for solving a particular promise problem using a particular ternary unitary operation. Its application is quite limited. Next, we will expand the work in [23] by developing the Chrestenson family of ternary quantum gates [24] based QFA. We further extend QFA applications to more practical scenarios such as pattern recognition and robotics applications.

3. Development of Chrestenson Family of Ternary Quantum Gates-Based QFA

In this section, we will explain the development of QFA using the Chrestenson family of ternary quantum gates [24]. We use the Chrestenson family of ternary operation because one of those gates was already realized by several companies and universities in various quantum technologies.

An equivalent of the binary quantum Hadamard gate in ternary quantum logic is one of the Chrestenson family of gates and their generalizations [24,28–30]. The complex third root of unity for Chrestenson gate is:

$$\gamma = e^{i\frac{-2\pi}{3}} = \cos\frac{-2\pi}{3} + i\sin\frac{-2\pi}{3} = -0.5 - i0.866$$

In addition to the well-known Chrestenson gate denoted by *CH*, we are introducing two new counterparts of Chrestenson gates defined by *CH2* and *CH3*. Note these gates are just permutations of the well-known gate *CH*. Observe that we prove the well-known permutative gates (1 2), (0 2), and (0 1) can be created from gate *CH* and our newly introduced gates *CH2* and *CH3*. The Chrestenson gate unitary matrices *CH, CH2, CH3* and their powers CH^2, $CH2^2$, $CH3^2$ are defined below.

$$CH = \frac{1}{\sqrt{3}}\begin{bmatrix} 1 & 1 & 1 \\ 1 & \gamma & \gamma^2 \\ 1 & \gamma^2 & \gamma \end{bmatrix}$$

$$CH2 = \frac{1}{\sqrt{3}}\begin{bmatrix} \gamma & 1 & \gamma^2 \\ 1 & 1 & 1 \\ \gamma^2 & 1 & \gamma \end{bmatrix}$$

$$CH3 = \frac{1}{\sqrt{3}}\begin{bmatrix} \gamma & \gamma^2 & 1 \\ \gamma^2 & \gamma & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$CH^2 = \frac{1}{\sqrt{3}}\begin{bmatrix}1 & 1 & 1\\ 1 & \gamma & \gamma^2\\ 1 & \gamma^2 & \gamma\end{bmatrix}.\frac{1}{\sqrt{3}}\begin{bmatrix}1 & 1 & 1\\ 1 & \gamma & \gamma^2\\ 1 & \gamma^2 & \gamma\end{bmatrix}$$

$$= \frac{1}{3}\begin{bmatrix}3 & 1+\gamma+\gamma^2 & 1+\gamma+\gamma^2\\ 1+\gamma+\gamma^2 & 1+\gamma^2+\gamma^4 & 1+\gamma^3+\gamma^3\\ 1+\gamma+\gamma^2 & 1+\gamma^3+\gamma^3 & 1+\gamma^2+\gamma^4\end{bmatrix} = \frac{1}{3}\begin{bmatrix}3 & 0 & 0\\ 0 & 0 & 3\\ 0 & 3 & 0\end{bmatrix} = \begin{bmatrix}1 & 0 & 0\\ 0 & 0 & 1\\ 0 & 1 & 0\end{bmatrix} = (1\,2)$$

$$CH2^2 = \frac{1}{\sqrt{3}}\begin{bmatrix}\gamma & 1 & \gamma^2\\ 1 & 1 & 1\\ \gamma^2 & 1 & \gamma\end{bmatrix}.\frac{1}{\sqrt{3}}\begin{bmatrix}\gamma & 1 & \gamma^2\\ 1 & 1 & 1\\ \gamma^2 & 1 & \gamma\end{bmatrix} = \frac{1}{3}\begin{bmatrix}0 & 0 & 3\\ 0 & 3 & 0\\ 3 & 0 & 0\end{bmatrix} = \begin{bmatrix}0 & 0 & 1\\ 0 & 1 & 0\\ 1 & 0 & 0\end{bmatrix} = (0\,2)$$

$$CH3^2 = \frac{1}{\sqrt{3}}\begin{bmatrix}\gamma & \gamma^2 & 1\\ \gamma^2 & \gamma & 1\\ 1 & 1 & 1\end{bmatrix}.\frac{1}{\sqrt{3}}\begin{bmatrix}\gamma & \gamma^2 & 1\\ \gamma^2 & \gamma & 1\\ 1 & 1 & 1\end{bmatrix} = \frac{1}{3}\begin{bmatrix}0 & 3 & 0\\ 3 & 0 & 0\\ 0 & 0 & 3\end{bmatrix} = \begin{bmatrix}0 & 1 & 0\\ 1 & 0 & 0\\ 0 & 0 & 1\end{bmatrix} = (0\,1)$$

Our extended Chrestenson gate unitary matrices CHC1 and CHC2 are defined below.

$$\text{CHC1} = CH^2 \times CH_2^2 = \begin{bmatrix}1 & 0 & 0\\ 0 & 0 & 1\\ 0 & 1 & 0\end{bmatrix} \times \begin{bmatrix}0 & 0 & 1\\ 0 & 1 & 0\\ 1 & 0 & 0\end{bmatrix} = \begin{bmatrix}0 & 0 & 1\\ 1 & 0 & 0\\ 0 & 1 & 0\end{bmatrix}$$

$$\text{CHC2} = CH^2 \times CH_3 = \begin{bmatrix}1 & 0 & 0\\ 0 & 0 & 1\\ 0 & 1 & 0\end{bmatrix} \times \begin{bmatrix}0 & 1 & 0\\ 1 & 0 & 0\\ 0 & 0 & 1\end{bmatrix} = \begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 1\\ 1 & 0 & 0\end{bmatrix}$$

We further define the following two Plus1 and Plus2 matrix operations that we call ternary rotations.

$$Plus1 = \begin{bmatrix}0 & 0 & 1\\ 1 & 0 & 0\\ 0 & 1 & 0\end{bmatrix}$$

$$Plus2 = \begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 1\\ 1 & 0 & 0\end{bmatrix}$$

CHC1 is the same as Plus 1. CHC2 is the same as Plus 2. These gates are well known in ternary quantum circuit synthesis. The equations for Plus1, Plus2, (1 2), (0 2), and (0 1) derived earlier prove that all permutative ternary single qutrit quantum gates can be built from any two of gates from set $\{CH, CH2, CH3\}$.

The Chrestenson gate *CH* was realized practically in transmon technology [31]. To the best of our knowledge, the gates *CH2* and *CH3* were not realized practically yet. However, these gates are only permutations of rows and columns of matrix *CH*. Therefore, we believe these gates can be realized similarly to gate *CH*. Observe that gates (1 2), (0 2), (0 1), Plus1, Plus2, and many others can be realized from the three gates introduced above and other permutations. Many authors assume the above-listed permutation gates or their subsets as fundamental basic gates [28]. We however demonstrated above that all ternary permutative quantum single-qutrit gates can be realized only from *CH* and *CH2*, or from *CH* and *CH3*, or from *CH2* and *CH3*. Therefore building any two variants of the Chrestenson gate becomes sufficient to design an arbitrary ternary quantum circuit or quantum automaton with single-qutrit gates. Next, some two-qutrit gates can be built [29,30], which allows the construction of an arbitrary permutative ternary quantum circuit such as an oracle for the Grover algorithm. Observe that in binary quantum logic the smallest universal permutative gate has three qubits, while in ternary quantum logic the smallest universal permutative gate has only two qutrits. This makes synthesis in this logic quite different than in standard binary quantum logic. Interesting automata can be built by mixing binary and ternary bases.

3.1. New Rules about Chrestenson Family Unitary Matrix

This section describes our new design on rules of the Chrestenson family unitary matrix that can be applied to QFA. We define basic quantum ternary states $|0\rangle$, $|1\rangle$, $|2\rangle$ as follows.

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \tag{1}$$

$$|1\rangle = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \tag{2}$$

$$|2\rangle = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{3}$$

In (1)–(3), $|0\rangle$, $|1\rangle$, and $|2\rangle$ is the Dirac notation of basic quantum ternary states; to the right of the equation sign above is the Heisenberg notation of quantum ternary state.

We assume B is the starting state and B can be ternary state $|0\rangle$, $|1\rangle$, $|2\rangle$. We found the following rules of the Chrestenson family ternary matrix operation that can be applied to QFA.

Rule 1: Assume B = $|1\rangle$ and B is the starting state and the only accepting state. CH^2 = (1 2). When applying CH^2 to B 2n times, where n is an integer greater than 0, the final state is the same as the starting state and the accepting state B.

Rule 2: Assume B = $|2\rangle$ and B is the starting state and the only accepting state. CH^2 = (1 2). When applying CH^2 to B 2n times, where n is an integer greater than 0, the final state is the same as the starting state and the accepting state B.

Rule 3: Assume B = $|0\rangle$ and B is the starting state and the only accepting state. $CH2^2$ = (0 2). When applying $CH2^2$ to B 2n times, where n is an integer greater than 0, the final state is the same as the starting state and the accepting state B.

Rule 4: Assume B = $|2\rangle$ and B is the starting state and the only accepting state. $CH2^2 = (0\ 2)$. When applying $CH2^2$ to B 2n times, where n is an integer greater than 0, the final state is the same as the starting state and the accepting state B.

Rule 5: Assume B = $|0\rangle$ and B is the starting state and the only accepting state. $CH3^2$ = (0 1). When applying $CH3^2$ to B 2n times, where n is an integer greater than 0, the final state is the same as the starting state and the accepting state B.

Rule 6: Assume B = $|1\rangle$ and B is the starting state and the only accepting state. $CH3^2$ = (0 1). When applying $CH3^2$ to B 2n times, where n is an integer greater than 0, the final state is the same as the starting state and the accepting state B.

Rule 7: Assume B = $|0\rangle$ and B is the starting state and the only accepting state. CHC1 = Plus 1. When applying CHC1 to B 3n times, where n is an integer greater than 0, the final state is the same as the starting state and the accepting state B.

Rule 8: Assume B = $|0\rangle$ and B is the starting state and the only accepting state. CHC2 = Plus 2. When applying CHC2 to B 3n times, where n is an integer greater than 0, the final state is the same as the starting state and the accepting state B.

Rule 9: Assume B = $|0\rangle$ and B is the starting state & F = $|1\rangle$ and F is the accepting state. When applying Plus1 to B 3n + 1 times, where n is an integer greater than or equal to 0, the final state is the accepting state F.

Rule 10: Assume B = $|0\rangle$ and B is the starting state & F = $|2\rangle$ and F is the accepting state. When applying Plus1 to B 3n + 2 times, where n is an integer greater than or equal to 0, the final state is the accepting state F.

Rule 11: Assume B = $|0\rangle$ and B is the starting state & F = $|1\rangle$ and F is the accepting state. When applying Plus2 to B 3n + 1 times, where n is an integer greater than or equal to 0, the final state is the accepting state F.

Rule 12: Assume B = $|0\rangle$ and B is the starting state & F = $|2\rangle$ and F is the accepting state. When applying Plus2 to B 3n + 2 times, where n is an integer greater than or equal to 0, the final state is the accepting state F.

Rule 13: Assume B = $|1\rangle$ and B is the starting state and the only accepting state. When applying Plus1 and Plus2 to B n times, where n is an integer greater than or equal to 1, as long as the number of times Plus1 is applied is equal to the number of times Plus2 is applied, the final state is the same as the starting state and the only accepting state B.

3.2. A General QFA with Uσi Using Chrestenson Family Matrix

Figure 4 is the state diagram of a general hybrid QFA with Uσ_i using matrices of gates *CH*, *CH2*, *CH3*, CH^2 = *(1 2)*, $CH2^2$ = *(0 2)*, $CH3^2$ = *(0 1)*, CHC1, and CHC2. All the Rules from Section 3.1 can be used to design the general hybrid QFA as shown in Figure 4.

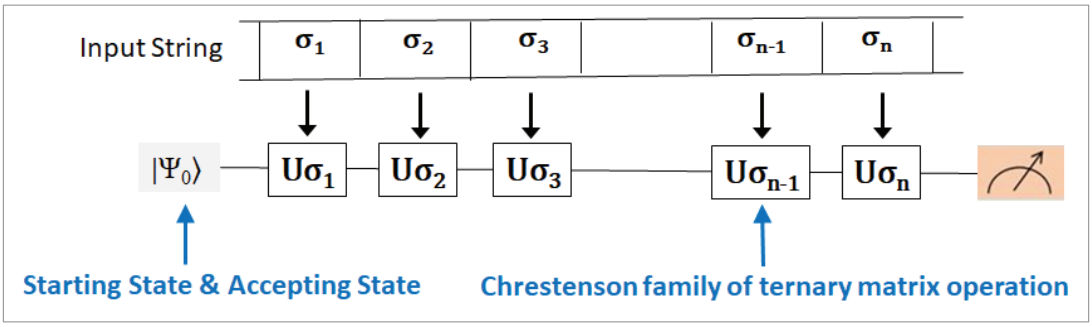

Figure 4. The State Diagram of a General QFA with Uσ_i Using Chrestenson Family Matrix.

The QFA in Figure 4 can be used to determine an input string pattern using a quantum state space with only three orthonormal basis states. QFA encodes quantum states with qutrits using three orthonormal basis states $|0\rangle$, $|1\rangle$, and $|2\rangle$. For example, we can apply Rule 1 from Section 3.1 to the QFA in Figure 4 to detect an input string pattern of a^{2n}, where n is an integer equal to or greater than 1. Assume B = $|1\rangle$ and B is the starting state and the only accepting state. Upon reading in a symbol, the same Chrestenson family of ternary matrix operation Uσ_i = CH^2 is applied to the current quantum state. If the input string pattern is a^{2n}, where n is an integer equal to or greater than 1, the final state is the accepting state.

We can also apply Rule 7 or Rule 8 from Section 3.1 to the QFA in Figure 4 to detect an input string pattern of a^{3n}, using the extended Chrestenson gate unitary matrix CHC1 or CHC2 as Uσ_i, where n is an integer equal to or greater than 1. We can also use the Plus1 and Plus2 unitary matrix as Uσ_i to detect input string patterns with an equal number of a's and b's using Rule 13 from Section 3.1.

We can use the QFA in Figure 4 to solve for many more different promise problems and pattern and language recognition problems by applying the Rules from Section 3.1. We can further use the combinations of CH^2, $CH_2{}^2$, $CH_3{}^2$, CHC1, CHC2, Plus1, and Plus2 ternary operation as Uσ_i. This can facilitate the development of more powerful and versatile quantum circuits, enabling advanced and efficient robotics applications.

4. Combining Rotation Based and Quantum Reversible Circuit Machines

This section shows how to combine a rotation ternary quantum circuit-based QFA and a quantum reversible circuit-based DFA to build a more powerful machine, enabling more complex language and pattern recognition.

As parts of our generalized quantum automata, we can build normal classical Deterministic Finite Automaton (DFA), which is discussed in [26]. An example of such machine M1 is shown in Figure 5. We can also build any machine based on rotations as discussed in Section 3. An example of such machine M2 is shown in Figure 6 using Plus1 and Plus2 ternary operation on input symbols a and b respectively. Assume that M1 is a quantum realization of classical DFA automaton. Assume that M2 is the quantum realization of rotation-based QFA automaton. The main idea of this paper is that we combine machines of type M1 and M2 by some kind of arbitrary Boolean function. The combined machine is more powerful than the normal classical DFA machine M1 or machine M2 and can enable more complex language recognition. The above-mentioned Boolean function can be realized either as a quantum reversible permutative circuit with Toffoli, Feynman, and NOT gates or by using a standard Boolean function or program realized in the controller.

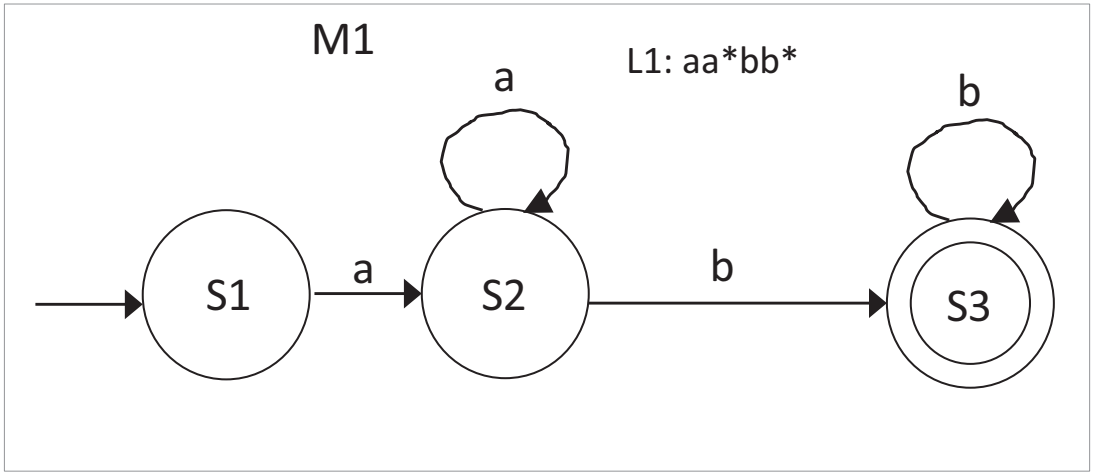

Figure 5. The State Diagram of a Normal Classical DFA Machine M1.

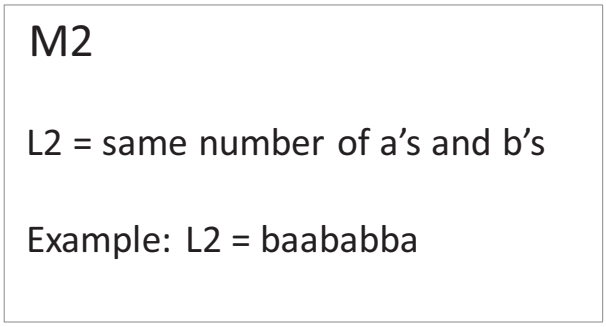

Figure 6. QFA Machine M2 Based on Rotation Ternary Quantum Circuits using Plus1 and Plus2 Ternary Operations.

Figure 7 shows an example of a combined machine M_combo1 with logic AND of Machine M1 and Machine M2. Machine M_combo1 can recognize the language pattern of $a^n b^n$. Machine M1 recognizes all sequences with arbitrary number of a's and arbitrary number of b's in which the symbols a are before symbols b. Machine M2 recognizes all sequences with equal number of a's and b's, regardless of the order of a's and b's. Therefore, Machine M_combo1$= M1 \cap M2$ recognizes language in which a's are always before b's and the number of a's is always equal to the number of b's.

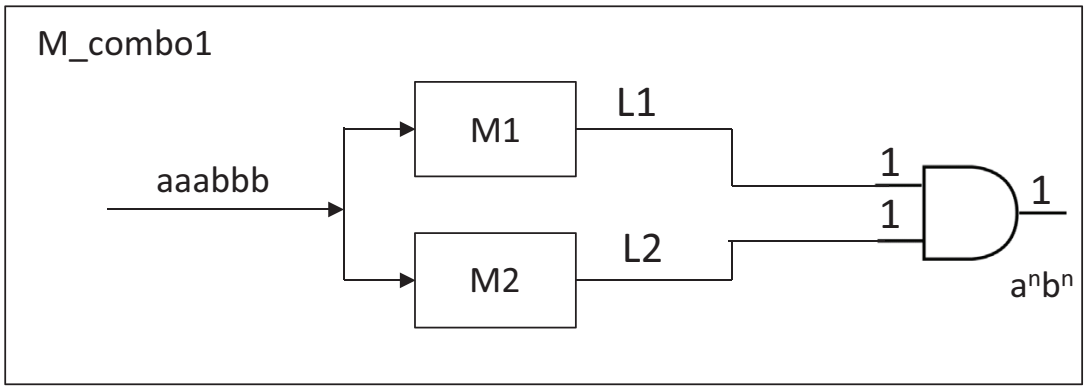

Figure 7. Machine M_combo1 with Logic AND of Machines M1 and M2.

Figure 8 shows machines M3 and M5. Machine M3 can recognize language L3 with a pattern of (aaa)*, where * is an iteration operation of regular languages. Machine M5 can recognize language L5 with a pattern of (aaaaa)*.

Figure 9 shows a combination of machine M_combo2 with logic AND of Machine M3 and Machine M5. Machine M_combo2 can recognize language L15 with pattern $(a^{15})^*$.

Figure 10 shows an example of a combination of machine M_combo3 with logic AND of Machine M17, M23, and M31. Machine M17, M23, and M31 can detect prime numbers 17, 23, and 31, respectively, and detect language $(a^{17})^*$, $(a^{23})^*$, and $(a^{31})^*$, respectively. Machine M_combo3 can detect number 12121 which is the product of numbers 17, 23, and 31 and can detect language L_combo3 which is $(a^{12121})^*$. Machines M17, M23, and M31 only need 3 state space each as explained in [23]. Machine M_combo3 also only needs 3 state space even though it can detect a large number 12121 or complex language L_combo3.

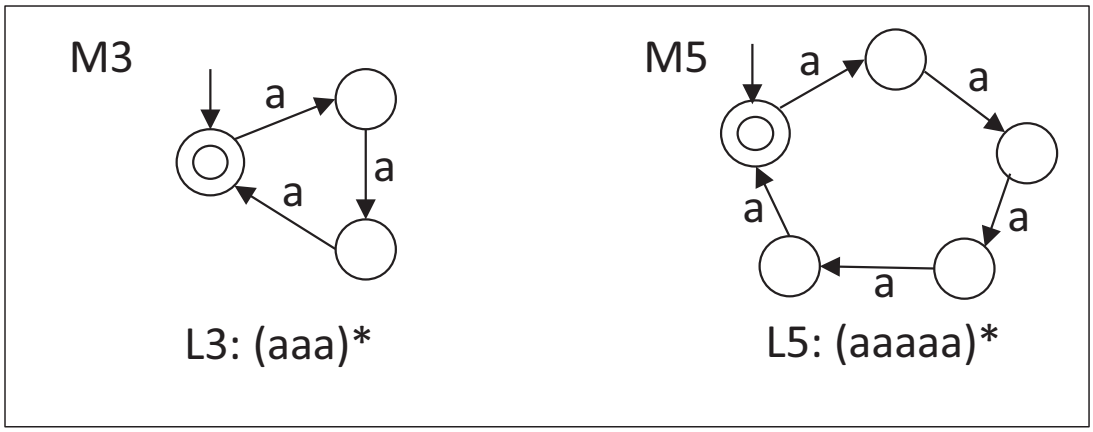

Figure 8. Machine M3 and Machine M5.

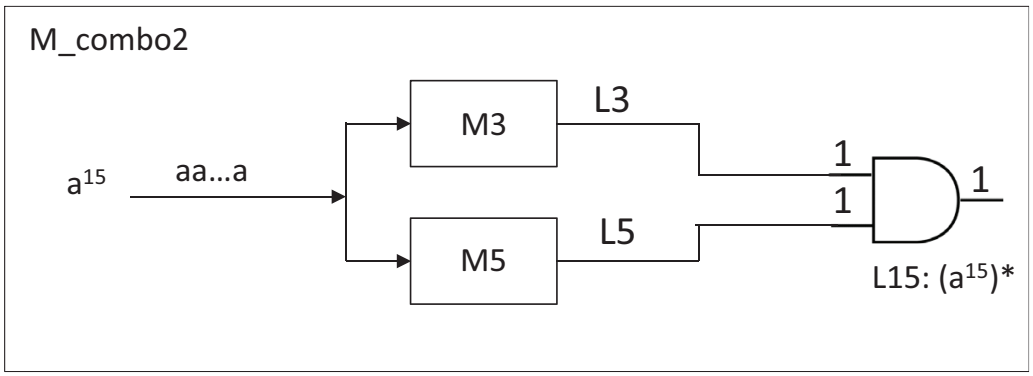

Figure 9. Machine M_combo2 with Logic AND of Machines M3 and M5.

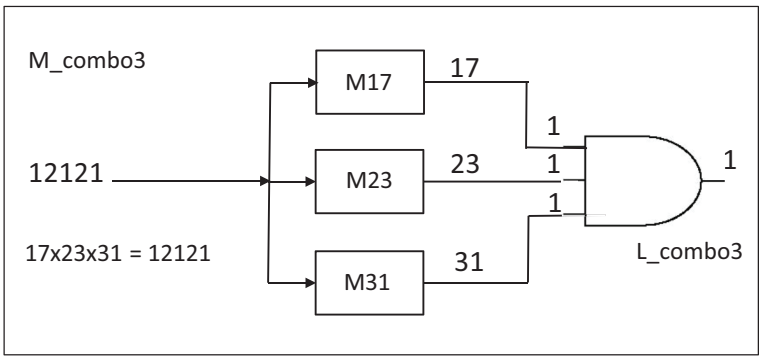

Figure 10. Machine M_combo3 with Logic AND of Machines M17, M23, M31.

5. QFA Application in Robotics

In this section, we will extend applications of our Generalized Hybrid Quantum Finite Automata (GHQFA) to more practical scenarios and describe how these machines can be used in robotics. Figure 11 shows the overall flowchart of QFA applications in robotics.

Figure 11 shows a step-by-step process for recognizing various patterns with GHQFA in robotics application. The flowchart [26] starts with various sensory inputs such as a camera for visual data, microphone for audio data, and light. These sensory inputs are captured by the Sensor block, which collects and converts the raw data into a form that can be further processed. The processed sensor data is then transformed into a sequence of characters (e.g., a sequence of codes, letters, etc.). The sequence of characters is interpreted as language, and then fed into the GHQFA for pattern recognition. Examples of pattern recognition using the GHQFA include language recognizer, gesture recognizer, posture recognizer, music recognizer, motion and emotion recognizer, and dance pattern recognizer. All of these are essential for robotics applications. We will give examples of language recognizer and gesture recognizer next.

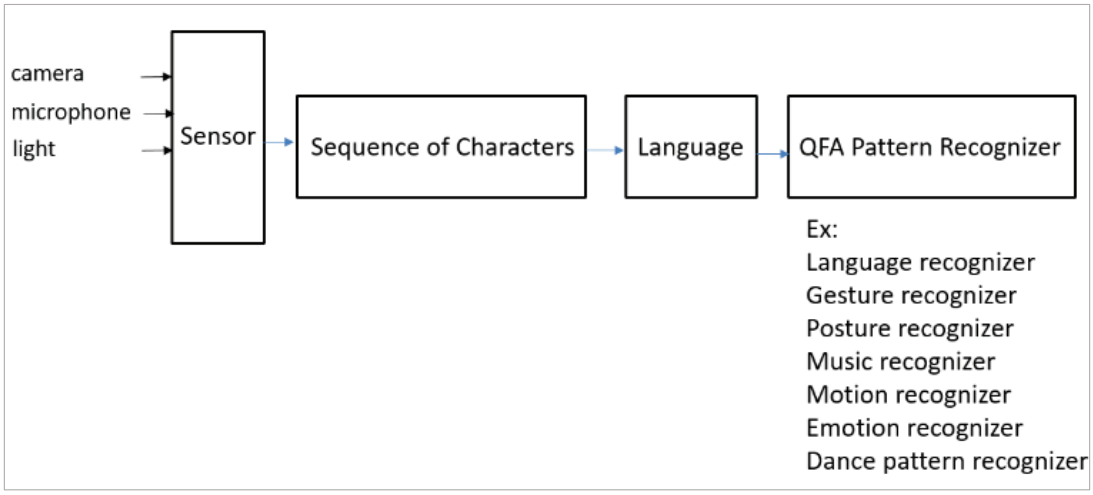

Figure 11. GHQFA Application in Robotics.

5.1. Language Recognizer Examples

Observe that in robotics many problems can be reduced to recognizing languages. For instance, symbols like a or b can correspond to recognizing images of human faces or doors or obstacles when a robot follows the wall of a long corridor. We can order the robot to stop after passing 14 doors on the left, which corresponds to language a^{14}. The following are some examples of this type of language recognizers, where symbols a and b below correspond to recognizing arbitrary patterns, objects, or situations by the perception sub-system of the robot.

$$\textbf{Language } L_1 = a^{2n}$$

where n is an integer greater than zero. CH^2 or $CH_2{}^2$ or $CH_3{}^2$ is applied to $U\sigma_i$ for QFA in Figure 3. We call this QFA as QFA1.

$$\textbf{Language } L_2 = a^{3n}$$

where n is an integer greater than zero, CHC1 or CHC2 or CHC3 is applied to $U\sigma_i$ for QFA in Figure 3. We call this QFA as QFA2.

$$\textbf{Language } L_3 = (a)^{6n}$$

where n is an integer greater than zero. The QFA detecting language L3 is a logic AND of QFA1 and QFA2.

$$\textbf{Language } L_4 = a^{2n}\, b^{3n}$$

where n is an integer greater than zero. When input string is a, CH^2 or $CH_2{}^2$ or $CH_3{}^2$ is applied to $U\sigma_i$ for QFA in Figure 3. When input string is b, CHC1 or CHC2 or CHC3 is applied to $U\sigma_i$ for QFA in Figure 3.

There can be many languages recognized using the Chrestenson family and extended Chrestenson family of ternary operation-based QFA similar to the above examples.

5.2. Gesture Recognizer Examples

$$\textbf{Gesture } G_1 = a^{2n}$$

waving left hand 2n times, where n is an integer greater than zero.

$$\textbf{Gesture } G_2 = a^{3n}$$

waving left hand 3n times, where n is an integer greater than zero.

$$\textbf{Gesture } G_3 = a^{2n}\ b^{3n}$$

waving left hand 2n times then waving right hand 3n times, where n is an integer greater than zero.

Figure 12 shows the work we did for quantum motion and emotion by a humanoid robot controlled by quantum circuits. Emotions are represented by quantum states of the controlling automaton while motions are external actions corresponding to internal emotions [32]. With GHQFA, we can have more powerful quantum circuits and generate more versatile robot gestures and motions. Because machines can be probabilistic, every behavior of the robot can be somehow different, although following certain fixed programmed patterns represented as a flowchart. There can be many gestures recognized and generated in GHQFAs that use the Chrestenson family and extended Chrestenson families of ternary operations, as shown in the above examples. In the future, the GHQFA can be extended to quantum logics with more than three basic quantum states. Practically, it can be extended to four or five basic quantum states. Hybrid deterministic machines can also be created with various mixed bases. Probabilistic and deterministic/ probabilistic machines with mixed bases can also be built.

Figure 12. HR-OS5 Humanoid Robot Actor.

6. Quantum Finite State Machine with Data Path for Language Recognition

The authors of [26] introduced a new method for synthesizing quantum automata from flowcharts using one-hot encoding. The one-hot encoding leads to simpler excitation functions with fewer inputs, leading to Toffoli gates with fewer inputs, therefore minimizing the quantum circuit cost. States are disjoint in one-hot encoding and the method in [26] is beneficial for quantum circuits due to the synthesis based on XOR operations that are a base of CNOT and CCNOT gates. This section talks about combining the idea of GHQFA with Quantum Finite State Machines (QFSM) from published papers [22,26] to the even more advanced concept of Quantum Finite State Machine with Datapath (QFSMD) to enable more powerful quantum circuits for robotics real-time control. The focus is to build a QFSM quantum automaton on top of one or more Quantum Finite Automata (QFA) or Generalized Hybrid Quantum Finite Automata (GHQFA). Figure 13 shows a simple example of QFSM built on top of QFA.

Figure 13 shows a Quantum Finite State Machine with Datapath (QFSMD) for language recognition. The concept of FSMD (Finite State Machine with Data Path) is well known in digital engineering and computer architecture. Such a machine is composed of a controlling automaton and a data path circuit. Controlling automaton sends control signals to the data path, such as A:=A + 1. Data path circuit is composed of registers, memories, arithmetic circuits, and predicate circuits (comparators). It sends status signals to the controller (such as A = 5). For instance, L1 and L0 can be control signals and FLAG can be a feedback signal from the data path.

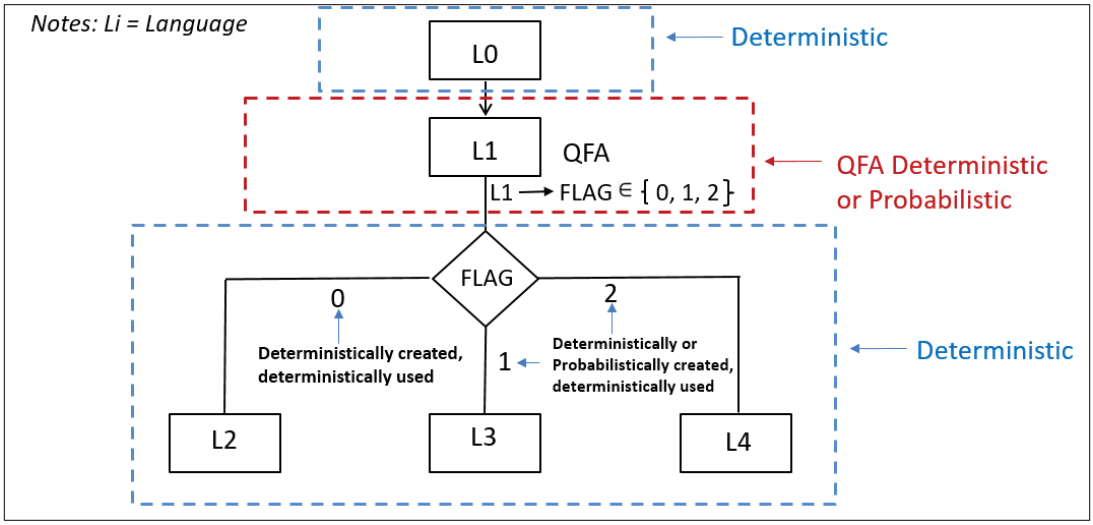

Figure 13. QFSMD Quantum Finite State Machine with Datapath for Language Recognition.

At the beginning in Figure 13, the machine deterministically recognizes the language L0. If the machine recognizes language L0, then the machine goes on to recognize language L1. If language L0 is not recognized, the machine is stopped.

Language L1 is recognized by some QFA. Language L1 has 3 possible recognized values 0, 1, and 2 given as feedback signals to the controller. Here the recognition process by QFA can be probabilistic or deterministic. The results of measurement (obtained probabilistically) are set to a standard FLAG, which is a variable with values 0, 1, or 2.

Now we go back to deterministic automata and we read the value of the flag.
If FLAG = 0, we go deterministically to recognize language L2.
If FLAG = 1, we go deterministically to recognize language L3.
If FLAG = 2, we go deterministically to recognize language L4.

The decision from QFA is deterministic when FLAG = 0. The way to obtain FLAG = 0 is deterministic. The way to obtain FLAG = 1 or FLAG = 2 can be probabilistic or deterministic through QFA, but the decision of what the robot does after FLAG = 1 or FLAG = 2 (which is L3 or L4) is deterministic.

7. Conclusion

In this paper, we developed various hybrid quantum finite automata using the Chrestenson family of ternary quantum gates and other similar gates. The main idea of this paper is to show how to combine rotation ternary quantum circuit-based QFA machine and quantum reversible circuit-based DFA machine (QFSM) to build various more powerful quantum hybrid machines. The combined machines enable more complex language and pattern recognition as well as real-time behavior in robot environments. The resulting GHQFA, QFSMD, and Generalized Hybrid Quantum Finite Automata with Datapath (GHQFSMD) machines can be used to build more powerful and versatile robotics controls. They can be used for robotics applications such as language recognizer, gesture recognizer and motion recognizer, localization, and navigation. We realize that, at this point in time, quantum technologies only allow the construction of small hybrid automata. However, [23] demonstrated that quite interesting languages can be recognized using optical technology. For instance, our method allows a robot to recognize languages as complicated as L_combo3 (from Section 4) when using ternary optical technology and relatively simple quantum hardware.

Although all examples given in this paper are rather simple and straightforward, the general methodology of creating hybrid quantum automata as outlined in Section 6 is very powerful because it can compose machines from hierarchical parallel and sequential structures of blocks. It is well known from digital design theory and practical computer architecture that advanced systems are built from such hierarchies of classical blocks such as counters, adders, etc. We believe that the same methodology can be invented for hybrid quantum automata. In our current research, we work on hybrid quantum controllers for practical robotic applications. Although contemporary quantum computers have a limited number of qudits and so far only qutrits seem to be more practical, we believe that fast progress in quantum technology (especially optical) will allow the realization of larger quantum hybrid automata in the near future. Observe that our quantum hybrid automata can collaborate with other quantum building blocks that

will execute specific control, pattern recognition, robot vision, Machine Learning, artificial intelligence, and security tasks [12,14,15,17,18,20,22,24,26,28,32–34].

Funding
This research received no external funding.

Conflicts of Interest Statement
The authors declare no conflicts of interest.

Data Availability Statement
All data supporting reported results can be found in this paper.

Abbreviations and Parameters

Abbreviation & Parameter	Meaning
QFA	Quantum Finite Automaton
QFSM	Quantum Finite State Machines
DFA	Deterministic Finite Automaton
QFSMD	Quantum Finite State Machine with Data Path
γ	The complex third root of unity for Chrestenson gate
CH	Chrestenson gate
CH2, CH3	New counterpart of Chrestenson gate
(1 2)	CH^2: Power of Chrestenson gate CH
(0 2)	$CH2^2$: Power of new counterpart of Chrestenson gate CH2
(0 1)	$CH3^2$: Power of new counterpart of Chrestenson gate CH3
CHC1, CHC2	Extended Chrestenson gate unitary matrices
Plus1	Plus1 ternary rotation matrix
Plus2	Plus2 ternary rotation matrix
GHQFA	Generalized Hybrid Quantum Finite Automata
FSMD	Finite State Machine with Data Path
GHQFSMD	Generalized Hybrid Quantum Finite Automata with Datapath

References

1. P.W. Shor (1994) "Algorithms for quantum computation: Discrete logarithms and factoring", in *Proceedings of 35th Annual Symposium Foundations of Computer Science*. Santa Fe, NM: IEEE, 124–134.
2. R.P. Feynman (1982) "Simulating physics with computers". *International Journal of Theoretical Physics*, 21, 467–488.
3. D.S. Abrams, and S. Lloyd (1997) "Simulation of many-body Fermi systems on a universal quantum computer". *Physical Review Letters*, 79, 2586.
4. A. Aspuru-Guzik, A.D. Dutoi, P.J. Love and M. Head-Gordon (2005) "Simulated quantum computation of molecular energies". *Science*, 309, 1704–1707.
5. T.D. Ladd et al. (2010) "Quantum computers". *Nature*, 464, 45–53.
6. D.W. Lu et al. (2017) "Enhancing quantum control by bootstrapping a quantum processor of 12 qubits". *npj Quantum Information*, 3, 45.
7. J.M. Gambetta, J.M. Chow and M. Steffen (2017) "Building logical qubits in a superconducting quantum computing system". *npj Quantum Information*, 3, 2.
8. C. Figgatt et al. (2017) "Complete 3-qubit Grover search on a programmable quantum computer". *Nature Communications*, 8, 1918.
9. T.F. Watson et al. (2018) "A programmable two-qubit quantum processor in silicon". *Nature*, 555, 633–637.
10. X. Qiang et al. (2018) "Large-scale silicon quantum photonics implementing arbitrary two-qubit processing". *Nature Photonics*, 12, 534–539.
11. S. Debnath et al. (2016) "Demonstration of a small programmable quantum computer with atomic qubits". *Nature*, 536, 63–66.
12. L.M.K. Vandersypen et al. (2001) "Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance". *Nature*, 414, 883–887.
13. P. Walther et al. (2005) "Experimental one-way quantum computing". *Nature*, 434, 169–176.
14. X.D. Cai, et al. (2013) "Experimental quantum computing to solve systems of linear equations". *Physical Review Letters,* 110, 230501.

15. Z. Li, X. Liu, N. Xu, and J. Du (2015) "Experimental realization of a quantum support vector machine". *Physical Review Letters,* 114, 140504.
16. H. Wang, et al. (2017) "High-efficiency multiphoton boson sampling". *Nature Photonics*, 11, 361–365.
17. M. Ying (2010) "Quantum computation, quantum theory and AI". *Journal of Artificial Intelligence*, 174: 2, 162–176. https://doi.org/10.1016/j.artint.2009.11.009
18. K. Culik II and J. Kari (1993) "Image compression using weighted finite automata". *Computer Graphics*, 17, 305–313.
19. A. Kondacs and J. Watrous (1997) "On the power of quantum finite state automata", in *Proceedings on 38th Annual Symposium Foundations of Computer Science*. IEEE: Miami Beach, FL, 66–75.
20. C. Moore and J.P. Crutchfield (2000) "Quantum automata and quantum grammars". *Theoretical Computer Science*, 237, 275–306.
21. A. Ambainis (2016) "Superlinear advantage for exact quantum algorithms". *SIAM Journal on Computing*, 45, 617–631.
22. M. Lukac and M. Perkowski (2008) "Quantum finite state machines as sequential quantum circuits." *Proceeding of ISMVL* 2009, 160–165.
23. Y. Tian, T. Feng, M. Luo, S. Zheng and Xiaoqi Zhou. (2019) "Experimental demonstration of quantum finite automaton". *Nature Partner Journals, Quantum Information*, published in partnership with the University of New South Wales. *npj Quantum Information* (2019) 5, 56. https://doi.org/10.1038/s41534-019-0163-x
24. J.H. Bae, Ch-B. Bae, G.B. Lee, D.H. Kim, M. Perkowski, M.H.A. Khan (2007) *Minimization of Ternary and Mixed Binary-Ternary Permutative Quantum Circuits.* Report KAIST University, Korea.
25. Nondeterministic finite automaton - Wikipedia
26. Y. Huang, M. Perkowski (2023) "One hot encoding synthesis of quantum automata from flowcharts". *Journal of Quantum Information Science*, 13: 3, 156–176. https://doi.org/10.4236/jqis.2023.133009
27. J. Gruska, D.W. Qiu and S.G. Zheng (2015) "Potential of quantum finite automata with exact acceptance". *International Journal of Foundations of Computer Science*, 26, 381–398.
28. M. Perkowski (2007, May) "Quantum robots. Now or never?" in *Invited Talk at the 5th National Conference on Informatics*. Gdansk, Poland.
29. A. Al-Rabadi, M. Perkowski, L. Casperson and X. Song (2002) "Multiple-valued quantum logic". *Quantum*, 10, 2.
30. M. Khan, M. Perkowski and P. Kerntopf (2003) "Multi-output galois field sum of products synthesis with new quantum cascades", in *Proc. 33rd ISMVL*, Tokyo, 16–19 May 2003, IEEE.
31. T. Roy, Z. Li, E. Kapit, and D.I. Schuster (2023) "Two-Qutrit quantum algorithms on a programmable superconducting processor". *Physical Review Applied*, 19, 064024.
32. R. Deng, Y. Huang, M. Perkowski (2021) "Quantum motions and emotions for a humanoid robot actor", in *Proceedings of the 2021 IEEE 51st International Symposium on Multiple-Valued Logic*, Nur-sultan, May 25–27, 2021, 207–214. https://doi.org/10.1109/ISMVL51352.2021.00043.
33. D. Said (2023) "Quantum computing and machine learning for cybersecurity: Distributed denial of service (DDos) attack detection on smart micro-grid". *Energies*, 16: 8, 3572. https://doi.org/10.3390/en16083572.
34. D. Said, M. Bagaa, A. Qukaira and A. Lakhssassi (2024) "Quantum entropy and reinforcement learning for distributed denial of service attack detection in smart grid". *IEEE Access*, 12, 129858–129869. https://doi.org/10.1109/ACCESS.2024.3441931.