Quantum Finite Automata and the Recognition of the Regular Languages: A Survey

Alex Hof

December 6, 2017

1 Introduction

Among the simplest classical models of computation is the deterministic finite automaton, or DFA. A DFA is a machine with a finite number of internal states and no other memory which, given an input word in some fixed alphabet, reads the word one character at a time and changes its internal state based on the symbol read. Once the word has been read, the machine will either accept it or reject it based upon the state it ends up in; the set of all words which a machine M accepts is called the language recognized by M. It is well-known that the set of languages which can be recognized by some DFA is precisely the set of languages which can be generated by some regular expression; these are the so-called regular languages.

In recent decades, however, there has been increasing interest in computational models which, rather than operating in a manner describable by classical physics, take advantage of the strange behavior of quantum systems; such attention has been driven in part by results such as the celebrated algorithm of Shor, which uses the power of quantum analogues to Turing machines to decompose large numbers into their prime factors in polynomial time [11]. This algorithm, along with others such as Grover's algorithm for database lookups in time proportional to the square root of the number of entries [5], suggests that quantum algorithms have rich possibilities to offer us in terms of computational efficiency. Therefore, it seems desirable to explore the properties of quantum computation, and in particular to design and analyze counterparts of fundamental classical models such as the DFA. In the case of the DFA, this counterpart is called a quantum finite automaton, or QFA.

There exist several distinct definitions of what a QFA is and how exactly it operates, the two most popular being that of Moore and Crutchfield [7] and that of Kondacs and Watrous [6]. In broad strokes, these formulations all essentially operate like DFAs, but with the distinction that, instead of simply occupying one internal state, they exist in a complex superposition of states which evolves according to operators corresponding to the symbols of the input word. Due to the effects of observation on a quantum system, however, one faces a choice not found in the classical case: namely, how often the machine should be observed.

The automata of Moore and Crutchfield allow exactly one observation, after the entire input word has been read, while those of Kondacs and Watrous are observed after every character of input; therefore, the former have come to be known as measure-once quantum finite automata, or MO-QFAs, and the latter measure-many quantum finite automata, or MM-QFAs. Comparison of these models by Ambainis and Freivalds [1] and Brodsky and Pippenger [3] demonstrated that MM-QFAs are capable of recognizing a larger class of languages than MO-QFAs, where recognition is couched in terms of giving the correct classification of any string with some bounded probability of error.

However, both models suffer from a troubling flaw: they are strictly weaker than their classical counterparts, in that they recognize only strict subsets of the regular languages. This problem arises because the requirements for a valid quantum state impose strict limitations on the allowable transition operators, corresponding in the classical case to the restriction that each symbol permute the states of the machine [3]. This paper will examine various efforts to ameliorate this problem; of particular interest will be the multi-letter automata of Belovs et al. [2], their subsequent analysis by Qiu and Yu [8], and the recent multihead automata of Ganguly et al. [4].

Finally, we propose a model, which we have named the repeating quantum finite automaton, which is capable of recognizing the regular languages but requires exponential time to do.

2 Preliminaries

2.1 DFAs and the regular languages

To understand the motivations for the definitions we will explore, it is will be helpful to have a rigorous picture of their classical counterpart. Formally, a deterministic finite automaton is a quintuple $M = (Q, \Sigma, \delta, q_0, Q_{\text{acc}})$, where the entries are as follows:

- Q is a finite set containing all of M's **internal states**.
- Σ is a finite set containing the **input alphabet**; that is, it is the set of all characters allowable in words which are to be processed by M.
- $\delta : Q \times \Sigma \to Q$ is M's **transition function**; given the machine's current state and the next letter of the input word, it outputs M's state after processing that letter.
- $q_0 \in Q$ is M's **start state**; the machine begins every computation in this state.
- $Q_{\text{acc}} \subseteq Q$ is the set of M's **accepting states**; once the letters of the input word are exhausted, M will accept the word if its current state is in Q_{acc} and reject it otherwise.

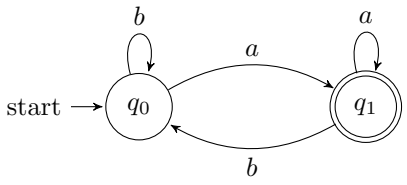

Figure 1: A DFA accepting the language $(a \cup b)^* a$

M runs by reading the letters of its input word, which is a finite, possibly empty sequence of characters of Σ, one letter at a time and evolving its state, which is initially q_0, according to δ. After all characters have been processed, the machine will accept the word if the state it has ended up in is an element of Q_{acc} and reject it otherwise. The computation has no output other than the decision to accept or reject the input word; unlike a Turing machine, which might write the result of a computation to its tape, or indeed an actual personal computer, which can relay a wide variety of information to the user through its display, a DFA can do nothing but say "Yes" or "No".

The language $L(M)$ recognized by M is defined to be the set of all strings of characters of Σ which M accepts. A DFA on the alphabet $\{a, b\}$ accepting the language of words ending with a is depicted in Figure 1; the double circle indicates that q_1 is an accepting state and the arrows and characters above them represent the transition function. The quintuple representation of this machine is $(\{q_0, q_1\}, \{a, b\}, \delta, q_0, \{q_1\})$, where δ is defined as

$$\delta(q, \sigma) = \begin{cases} q_1 & \text{if } \sigma = a \\ q_0 & \text{if } \sigma = b \end{cases}.$$

If this machine is given the word aab as input, it will begin in state q_0, transition to q_1 after reading the first a, remain in q_1 upon reading the second a, and finally transition back to q_0 on reading b. Since there are no more input letters, the machine will halt at this point; q_0 is not an accepting state, so the string will be rejected.

At this point it is natural to wonder whether there exists a DFA accepting each possible language. That is, if Σ is a finite alphabet and $\mathcal{L}$ is a subset of Σ^*, the set of all finite words formed from letters of Σ, can we create a DFA M such that $L(M) = \mathcal{L}$? As it turns out, this is not possible in general. For example, the language $\{a^n b^n \mid n \geq 0\}$ of words from $\{a, b\}^*$ which consist of some number of as followed by the same number of bs cannot be recognized by any DFA, since the only memory available is the internal state of the machine and there are infinitely many possible numbers of as; because a DFA may have only finitely many states, there is no way for the machine to accurately remember the number of as encountered when processing the string in all possible cases.

Therefore, we should refine our question slightly and instead ask precisely which languages *can* be recognized by DFAs. Such languages are called **regular**

languages. We can define this class of languages, as we have done, in terms of the model of computation which recognizes them; however, it is also useful to instead think about the constructions which *generate* them. Specifically, the regular languages are those which can be generated by the regular grammars or, equivalently, by the regular expressions, constructions which we will now explore.

A **grammar**, as defined in [9], is a construct which encodes rules for generating allowable strings. Formally, a grammar is a 4-tuple $G = (V, \Sigma, R, S)$, where the entries are as follows:

- V is a finite set called the **rule alphabet**.
- $\Sigma \subset V$ is the **terminal alphabet** of G; that is, the words G generates are in Σ^*. The set $V \setminus \Sigma$ of symbols in the rule alphabet but not the terminal alphabet is called the **nonterminal alphabet** of G.
- R is a finite set of **rules**, or **productions**, of the form $x \to y$ for $x \in V^* \setminus \Sigma^*$ and $y \in V^*$.
- $S \in V \setminus \Sigma$ is the nonterminal **start symbol**.

G is used to produce a string of Σ^* as follows. We begin with a one-character string in V^* consisting our start symbol S. We then repeatedly apply our production rules to the string by searching for a substring that matches the left-hand side of any rule and replacing it with the string on the right-hand side of the rule. We stop when we have eliminated all nonterminals from our string, and the resulting element of Σ^* is said to have been **generated** by G. We denote by $L(G)$ the language of all strings which can be generated by G.

A grammar $G = (V, \Sigma, R, S)$ is said to be **regular** if, for every rule $x \to y$ in R, two conditions hold. The first is that x consists of a single nonterminal symbol; that is, our grammar operates solely by replacing nonterminals with other strings without regard for the characters around them. The second is that y is either the empty string, which we denote ε, a string consisting of a single terminal symbol, or a string consisting of a single terminal symbol followed by a single nonterminal symbol. This limits the grammar to producing terminal symbols one at a time in the order in which they appear in the generated string.

As an example, we can formalize the language accepted by the DFA of Figure 1 with the regular grammar $(\{a, b, S\}, \{a, b\}, R, S)$, where R contains the rules $S \to aS$, $S \to bS$, and $S \to a$. To generate the string $baba$, we use the second production, followed by the first, followed by the second, and conclude with the third:

$$S \to bS \to baS \to babS \to baba$$

It is not difficult to verify that this grammar does in fact generate exactly the language accepted by the machine.

Finally, we can also define the regular languages as those which are expressible as **regular expressions**. Since the definitions of an allowable regular expression and of the language which it describes are both widely known and

somewhat involved, we do not reproduce them here, although we will occasionally make use of them to express languages compactly, as in the caption of Figure 1; for the interested reader, an overview can be found in the sixth chapter of [9].

We also collect some well-known properties of regular languages, as set forth in [9]. It is not difficult to see that every finite language is regular. Moreover, the regular languages exhibit a number of important closure properties. Let $L, K \subseteq \Sigma^*$ be regular languages over the same alphabet. Then the following are also regular languages:

- $L \cup K$, the **union** of the two languages.
- $L \cap K$, the **intersection** of the two languages.
- $L^C = \Sigma^* \setminus L$, the **complement** of L.
- $L \setminus K$, the **difference** of L and K.
- $LK = \{\ell k \mid \ell \in L, k \in K\}$, the **concatenation** of the two languages.
- $L^* = \{w_1 w_2 \dots w_n \mid w_1, w_2, \dots, w_n \in L, n \geq 0\}$, the **Kleene star** of L. Note that this includes the empty string.
- $L^R = \{\sigma_1 \sigma_2 \dots \sigma_n \mid \sigma_1, \sigma_2, \dots, \sigma_n \in \Sigma^*, \sigma_n \sigma_{n-1} \dots \sigma_1 \in L, n \geq 0\}$, the **reverse** of L.
- $\phi(L)$, where ϕ is a **homomorphism** on Σ^*. That is, ϕ is a function from Σ^* to T^* for some finite alphabet T such that, for all $x, y \in \Sigma^*$, $\phi(xy) = \phi(x)\phi(y)$. Note that this necessitates $\phi(\varepsilon) = \varepsilon$, and indeed ϕ may be uniquely defined by its behavior on Σ.

Some of these properties may be inferred from the others, but we include them all for the sake of completeness. The regular languages also satisfy the so-called **pumping lemma**, which states that, roughly speaking, if w is a sufficiently long string of a particular regular language L, there exists some nonempty substring within w that can be repeated an arbitrary number of times or eliminated without producing a string outside of L.

2.2 Hilbert spaces

The mathematical underpinnings of the quantum-mechanical concepts needed to define QFAs are quite extensive, and to build them up from first principles is outside the scope of this survey. Therefore, we assume a basic knowledge of linear algebra and treat a number of other topics somewhat more perfunctorily than they deserve.

Most of quantum mechanics takes place in the so-called Hilbert spaces, the definition of which requires the notion of a **Hermitian inner product**. Consider a vector space $\mathcal{V}$ over the field $\mathbb{C}$ of complex numbers. Then a binary operator $\langle \cdot, \cdot \rangle : \mathcal{V} \times \mathcal{V} \to \mathbb{C}$ is called a Hermitian inner product on $\mathcal{V}$ if it has the following properties:

- $\langle u, v\rangle = \overline{\langle v, u\rangle}$ for all $u, v \in \mathcal{V}$.
- $\langle \cdot, \cdot \rangle$ is linear in its second argument. By the first property, this implies conjugate-linearity in the first argument.
- For every $v \in \mathcal{V}$, $\langle v, v\rangle$ is real and nonnegative. Moreover, $\langle v, v\rangle = 0$ if and only if v is the zero vector.

In this case $\mathcal{V}$ is called an **inner product space**. It is worth noting that many authors, especially in mathematical circles, use a slightly different definition in which the inner product is linear in the *first* argument and conjugate-linear in the *second*; however, the given definition is more common in physics and therefore more appropriate in this context.

An inner product induces a notion of length, or **norm**, on $\mathcal{V}$. This norm, denoted by $\|\cdot\| : \mathcal{V} \to \mathbb{R}$, is given by $\|v\| = \sqrt{\langle v, v\rangle}$, and it in turn induces a notion of distance, or **metric**, on $\mathcal{V}$. Specifically, the distance between any two vectors is defined to be the norm of the difference between them; by inspecting the definition of the inner product, one can verify that this is well-defined.

Let $\mathcal{H}$ be a vector space over $\mathbb{C}$ endowed with a Hermitian inner product. Then we say that $\mathcal{H}$ is a **Hilbert space** if it is **complete** with respect to the metric induced by the inner product. Rather than rigorously defining completeness here, we will content ourselves with saying that it corresponds in some sense to $\mathcal{H}$ containing all the points which "should" be in it; the interested reader is encouraged to pursue studies in mathematical analysis.

In a Hilbert space, and indeed in any inner product space, there are several additional notions that will be of interest to us. Suppose $\mathcal{V}$ is an inner product space over $\mathbb{C}$ and $u, v \in \mathcal{V}$. Then we say that u and v are **orthogonal** if $\langle u, v\rangle = 0$. More generally, a set $X \subseteq \mathcal{V}$ is said to be orthogonal if each pair of distinct elements in X is orthogonal. If $X \subseteq V$ is orthogonal and, in addition, each element of X has length 1 under the norm induced by the inner product (that is, $\langle x, x\rangle = 1$ for each $x \in X$), we say that X is **orthonormal**.

Finally, it will be useful to review an alternate notation for the inner product preferred by many physicists. If $\mathcal{V}$ is an inner product space (usually a Hilbert space in practice) and $v \in \mathcal{V}$, it is common to instead denote v by the **ket** $|\psi\rangle$, where ψ is some arbitrary symbol defined by the author. If $|\psi\rangle$ corresponds to the vector v, the **bra** $\langle\psi|$ denotes the linear operator from $\mathcal{V}$ to $\mathbb{C}$ which takes any vector u to $\langle v, u\rangle$. Thus the **bracket** $\langle\psi|\psi\rangle$, which denotes $\langle\psi|$ applied to $|\psi\rangle$, is simply equal to the inner product of v with itself, or $\|v\|^2$. In the case where $\mathcal{V}$ is finite-dimensional, $|\psi\rangle$ can be thought of as the column vector corresponding to v with respect to some fixed orthonormal basis. In this case, we can view $\langle\psi|$ simply as the conjugate transpose of $|\psi\rangle$ and $\langle\psi|\psi\rangle$ as the usual matrix product of $\langle\psi|$ and $|\psi\rangle$. In either framework, the bras and kets behave normally as objects of their respective types, with the only notational differences being that, as is often the case in linear algebra, we forsake the use of parentheses for function arguments and that we always write $\langle a|b\rangle$ instead of $\langle a||b\rangle$.

2.3 Quantum mechanics

We will now review the actual quantum-mechanical concepts necessary to define QFAs; this discussion is largely based on the explication found in [6], with some alterations made to yield greater generality and a few additional details included.

To define a particular **quantum system**, it is necessary only to specify a Hilbert space $\mathcal{H}$, which determines the allowable states of the system. In physical reality, a quantum system might, for example, correspond to an electron, photon, or other such object; we will define them more abstractly as superpositions of the states in our automata. A **quantum state** is vector in $\mathcal{H}$ which has norm 1; that is, our states live on a spherical shell of radius 1 around the origin (where we generalize the notion of sphericality to complex vector spaces of arbitrary dimension).

Given a system occupying some quantum state $s \in \mathcal{H}$, we are often interested in obtaining information about that state. An **observable** is a decomposition

$$\mathcal{H} = \bigoplus_{\alpha \in I} \mathcal{O}_\alpha$$

of $\mathcal{H}$ into subspaces, indexed by some set I, which are pairwise orthogonal; that is, for any $\alpha \neq \beta \in I$, $x \in \mathcal{O}_\alpha$, and $y \in \mathcal{O}_\beta$, we have $\langle x, y \rangle = 0$. An **observation** of the system with respect to the given observable exposes some element of I to the observer, where the probability of any particular $\alpha \in I$ is given by the squared norm of s's projection onto $\mathcal{O}_\alpha$; if we define $s_\alpha = \mathrm{Proj}_{\mathcal{O}_\alpha}(s)$, this is to say that the probability of α is $\langle s_\alpha, s_\alpha \rangle$. The condition $\|s\| = 1$ guarantees that this will yield a valid probability distribution.

In addition to revealing information about the system, however, an observation actually alters it as well. If the observation returns α and s_α is defined as before, the system's state following the observation will be given by $\frac{s_\alpha}{\|s_\alpha\|}$. That is, the act of observing projects the system's state onto the subspace corresponding to the value observed, and the result is renormalized to yield a valid new state.

In quantum mechanics, the spaces constituting an observable are often thought of as the eigenspaces of some operator on $\mathcal{H}$ which satisfies certain properties, and in this case the value observed is the eigenvalue of the associated space. For example, a measurement of some particle's position along a given spatial axis would use an observable defined by an operator with uncountably many real eigenvalues, one for each possible location along the axis. However, this conceptual machinery will not be necessary for our purposes.

At this point, it may be helpful to consider an example. Suppose we have a classical system which may occupy one of two states, a or b, and we wish to define its quantum analogue. Then we can let $\mathcal{H} = \mathbb{C}[\{a, b\}]$ be the space of $\mathbb{C}$-linear combinations of a and b, which are taken to be linearly independent, and uniquely define an inner product on $\mathcal{H}$ by stipulating that $\{a, b\}$ is an orthonormal set. In this case, $\mathcal{H}$ is a 2-dimensional Hilbert space.

Therefore, our quantum system, instead of simply being in one of the two states, occupies a superposition $s = z_1 a + z_2 b$, where $z_1, z_2 \in \mathbb{C}$ and $\overline{z_1} z_1 + \overline{z_2} z_2 =$

$|z_1|^2 + |z_2|^2 = 1$. We can define an observable corresponding to our original two states by

$$\mathcal{H} = \mathbb{C}a \oplus \mathbb{C}b;$$

observing the system with respect to this decomposition will cause it to assume state a with probability $|z_1|^2$ and b with probability $|z_2|^2$ and relay the outcome to the observer.

In fact, if we replace a by 0 and b by 1, we can see that the classical system in question is essentially a bit, and so the analogue we have just defined is a quantum bit, or qubit [10], one of the foundational constructs in the field of quantum computing.

3 MO-QFAs

We are now prepared to examine quantum analogues to the deterministic finite automata. We begin with the measure-once quantum finite automata of Moore and Crutchfield [7], originally introduced in 1997.

3.1 Definition

Although Moore and Crutchfield introduced MO-QFAs, we will follow the equivalent definition of Brodsky and Pippenger [3], who created more directly comparable formalisms for MO-QFAs and MM-QFAs and appear to have introduced the present terminology, although later sources use it without explicit citation [2, 8, 4]. A **measure-once quantum finite automaton** is, formally, a quintuple $M = (Q, \Sigma, \delta, q_0, Q_{\text{acc}})$, where the entries are as follows:

- Q is a finite set of classical states. The machine will assume states in the quantum system corresponding to the $|Q|$-dimensional Hilbert space $\mathcal{H}(Q) = \mathbb{C}[Q]$, the set of all $\mathbb{C}$-linear combinations of elements of Q with an inner product defined such that Q is an orthonormal basis for the space.

- Σ is a finite input alphabet, which we augment with a distinct end-of-tape symbol \$ by defining the tape alphabet $\Gamma = \Sigma \sqcup \{\$\}$.

- $\delta : Q \times \Gamma \times Q \to \mathbb{C}$ is a transition function. If the machine is in a superposition $s \in \mathcal{H}(Q)$ and reads character $\sigma \in \Gamma$, the new state of the machine will be produced by applying to s the linear transformation U_σ defined by setting $U_\sigma(q) = \sum_{q' \in Q} \delta(q, \sigma, q')q'$ for $q \in Q$ and extending linearly. Since $U_\sigma(s)$ must also be a valid quantum state, U_σ must be norm-preserving, or **unitary**, for each σ; this is equivalent to the requirement that the columns of U_σ's matrix representation with respect to the basis Q be orthonormal in $\mathcal{H}(Q)$.

- $q_0 \in Q$ is the initial state of the machine; that is, M starts in a superposition given by the basis vector corresponding to q_0.

- $Q_{\text{acc}} \subseteq Q$ is the set of classical accepting states, and we define $Q_{\text{rej}} = Q \setminus Q_{\text{acc}}$ to be the set of classical rejecting states. We can then see that the subspaces $\mathcal{H}(Q_{\text{acc}})$ and $\mathcal{H}(Q_{\text{rej}})$ of $\mathcal{H}(Q)$ spanned by Q_{acc} and Q_{rej} respectively are orthogonal to one another.

When we run M, we conceptualize it as having a read head moving along a finite tape which contains the input word followed by the end-of-tape symbol \$. M begins in the quantum superposition q_0 with its head pointing to the first character of the input word and, for each symbol σ it reads, evolves its superposition according to the operator U_σ defined by δ, moving its head one symbol along the tape in the process. After the end-of-tape character has been read, we halt the machine and measure its state with respect to the observable $\mathcal{H}(Q) = \mathcal{H}(Q_{\text{acc}}) \oplus \mathcal{H}(Q_{\text{rej}})$. Depending on the result of the measurement, we either accept or reject the word.

Moore and Crutchfield also introduce a generalization wherein the initial superposition need not correspond to any classical state, nor even have norm 1, and the operators U_σ need not be unitary; their terminology would suggest the name "generalized quantum finite automata" for such machines. However, the formalism is somewhat unclear since the authors never explicitly address the problem of ensuring a valid probability distribution; throughout their work, they seem to treat such machines as though they guarantee that their states have projections onto the accepting space with magnitude in $[0, 1]$, but never give any restriction which would ensure this. For this reason, and because the loosened restrictions make the machines no longer properly quantum-mechanical, we do not treat them here, although Moore and Crutchfield demonstrated that they are capable of recognizing the regular languages.

3.2 Quantum languages

One of the most immediately striking differences between the MO-QFAs and the DFAs is that while the latter are deterministic, as their name would suggest, the former are not. From our definition above, we can see that, unless a word happens to send the initial state to a superposition contained entirely in either $\mathcal{H}(Q_{\text{acc}})$ or $\mathcal{H}(Q_{\text{rej}})$, it may be either accepted or rejected, each with some positive probability. Therefore, it is no longer possible to speak naively of the language recognized by a certain machine; more advanced definitions are required.

Moore and Crutchfield solve this problem by generalizing the notion of a language. They define a **quantum language** as a function $f : \Sigma^* \to [0, 1]$ which, given an input word, assigns it some value corresponding to the probability that it is in the language. Within this framework, a classical language can be identified with its characteristic function, which assigns every element in the language a probability of 1 and every element not in the language a probability of 0 [7]. The authors then define the quantum language accepted by a MO-QFA M to be the function f_M which takes a word to the probability that M accepts it, and designate the class of quantum languages accepted by some MO-QFA as

the "quantum regular languages". Since we are dealing with a larger variety of quantum finite automata, we will instead refer to these as the **measure-once quantum regular languages**, or MO-QRLS, to avoid ambiguity.

Moore and Crutchfield proved a number of basic properties for the class of MO-QRLs. In brief, for any fixed alphabet Σ, this class includes all constant functions from Σ^* to $[0, 1]$ and is closed under the operations of taking weighted averages, multiplication, and complementation (where the complement of a language f is given by the function $1 - f$). Moreover, if f is a measure-once quantum regular language on Σ^*, T is a finite alphabet, and $\phi : T^* \to \Sigma^*$ a homomorphism, then the **inverse image** of f under ϕ, defined as $f \circ \phi$, is a measure-once quantum regular language on T^*.

The authors also demonstrate an analogue to the pumping lemma for regular languages, which, surprisingly, allows *any* subword of a word to be repeated some number of times, although the allowed numbers of repetitions are much more restricted than in the classical case. Specifically, if $f : \Sigma^* \to [0, 1]$ is a MO-QRL and $w \in \Sigma^*$, the properties of unitary matrices allow us to demonstrate for any $\delta > 0$ the existence of some positive integer k such that $|f(uw^k v) - f(uv)| \leq \delta$ for any $u, v \in \Sigma^*$. By applying this lemma to MO-QFAs which take values in $\{0, 1\}$, Moore and Crutchfield obtain constraints which imply that there exist regular languages of which the characteristic functions are not MO-QRLS. However, it is possible to show that, if a classical language L's characteristic function is a MO-QRL, L is regular [7].

3.3 Probabilistic recognition

Therefore, the set of classical languages which can be recognized with certainty by MO-QFAs is a strict subset of the regular languages. This state of affairs is unsatisfactory for a number of reasons. Since we have gone to the trouble of incorporating quantum behavior into our automata, we would like to have some sort of advantage to make it worth our while, but instead we seem to have made our machines weaker. Moreover, since most of the MO-QFAs we can define do not behave deterministically, this way of looking at the problem renders a majority of our machines useless if we are interested solely in the recognition of the classical languages.

Brodsky and Pippenger [3] attempted to resolve these issues by focusing on the classical languages and using a relaxed definition of the language accepted by an automaton common in the study of probabilistic automata. This definition takes a probabilistic machine M with corresponding quantum language f_M to accept a classical language L with cut-point $\lambda \in (0, 1)$ exactly when $f_M(L) \subseteq (\lambda, 1]$ and $f_M(L^C) \subseteq [0, \lambda]$. If there exists a $\varepsilon > 0$ such that $f_M(L) \subseteq (\lambda + \varepsilon, 1]$ and $f_M(L^C) \subseteq [0, \lambda - \varepsilon)$, it is said that M **accepts** L **with bounded error** and ε is called the **margin** [3].

Brodsky and Pippenger use these notions to define a number of language classes relevant to MO-QFAs; as they note, the cut-point of a given automaton can be modified as desired by a suitable alteration to the machine, so these classes are not parameterized by λ [3]. The authors define **RMO** to be the

class of all languages accepted by MO-QFAs with bounded error and **UMO** the class of languages accepted by MO-QFAs with unbounded error, so that **RMO** $\subseteq$ **UMO**.

As it turns out, **RMO** is a strict subclass of the regular languages, and in fact Brodsky and Pippenger were able to show that any language in **RMO** can be accepted by a MO-QFA with transition function defined such that the machine never exhibits quantum behavior by occupying a superposition involving multiple states, which is to say a MO-QFA which is also a DFA [3]. As the authors note, this implies that, for any fixed $\delta > 0$, **RMO** is exactly the class of languages accepted by MO-QFAs with margin at least δ, yielding the equivalence of these classes across all δ.

The intersection of the class of DFAs with the class of MO-QFAs is actually set of classical machines known as the **group automata**, and so **RMO** is the set of **group languages** accepted by these automata [3]. Since the quantum-mechanical formalism is superfluous in this case, we will not examine these languages in detail.

The class **UMO** is, in some sense, more promising. The authors were able to demonstrate, by constructing a MO-QFA accepting the non-regular language of words of $\{a, b\}^*$ which do not possess the same number of as as bs with unbounded error, that this class is not a subset of the regular languages; incidentally, this implies that **RMO** $\subset$ **UMO** [3]. However, this result is not as exciting as it might appear at first glance, since the lack of a bound on the error for these machines means that there will be words both within and without the language which are accepted with arbitrarily similar probabilities near the cut-point; therefore, the automata will not be useful in practice for establishing whether such words are in the language. It is apparent that a more powerful formalism is needed.

4 MM-QFAs

A measure-once quantum finite automaton, true to its name, performs exactly one observation in the process of reading any given input word, and this is the most natural quantum analogue to a deterministic finite automaton in that, in both cases, the machine's state changes predictably when encountering any particular input symbol and acceptance of a string is determined only by the state after processing it fully.

However, when Kondacs and Watrous [6] approached the problem of defining quantum analogues to the DFAs in 1997, slightly ahead of Moore and Crutchfield, they did so in a way that led them to consider a different construction altogether. Their work focused mainly on the so-called 2-QFAs, which correspond in the classical case to generalized versions of DFAs which operate on a circular tape and which, at every step, choose to move their read head forward, backward, or not at all based on the input symbol and their internal state [6]. The 2-QFAs are defined by replacing both the internal state and the position of the read head of such a machine with quantum superpositions, and the resulting

automata are, collectively, strictly more powerful that the DFAs in that they are capable of recognizing all regular languages, as well as some non-regular languages such as $\{a^n b^n \mid n \geq 1\}$, in linear time with bounded error [6].

Now, since the 2-way DFAs on which the 2-QFAs are based are equivalent in power to the usual DFAs [6], it might seem as though the 2-QFAs are perfectly reasonable analogues to the DFAs, and that we have succeeded in our quest to find a model for the QFAs which offers some advantage over classical methods. However, as Ambainis and Freivalds pointed out in 1998, this construction requires the dimension of the Hilbert space containing the machine's state vectors to become arbitrarily large as the input word does, since the number of possible positions for the read head depends on the word size, and so it is not likely to be practically implementable in the foreseeable future, if at all [1]. Therefore, scholarly attention has largely been focused on the machines Kondacs and Watrous called 1-QFAs, which are defined as 2-QFAs which must move forward along the input word at every step and which traverse the tape, no longer taken to be circular, exactly once [6]; these have come to be known as the measure-many quantum finite automata [3].

4.1 Definition

As before, while giving due credit to the machines' original inventors, Kondacs and Watrous [6], we choose to reproduce Brodsky and Pippenger's more straightforward definition [3], with slight stylistic modifications. Formally speaking, a **measure-many quantum finite automaton** is defined as a sextuple $M = (Q, \Sigma, \delta, q_0, Q_{\text{acc}}, Q_{\text{rej}})$, where the entries are as follows:

- Q, as in the MO-QFA definition, is a finite set of classical states. As before, we define $\mathcal{H}(Q)$ to be the associated Hilbert space.
- Σ is a finite input alphabet; we again add an end-of-tape symbol \$ to obtain the tape alphabet $\Gamma = \Sigma \sqcup \{\$\}$.
- $\delta : Q \times \Gamma \times Q \to \mathbb{C}$ is a transition function satisfying the same requirements as in the MO-QFA case, and we define the transition matrices U_σ as before as well.
- $q_0 \in Q$ is, as in the MO-QFA case, the classical state the machine begins in.
- $Q_{\text{acc}}, Q_{\text{rej}} \subseteq Q$ are disjoint sets containing the accepting and rejecting classical states, respectively, of M. We also define the set of non-halting states by $Q_{\text{non}} = Q \setminus (Q_{\text{acc}} \sqcup Q_{\text{rej}})$, and adopt the convention that $q_0 \in Q_{\text{non}}$.

As in the case of a MO-QFA, we take our tape to contain the input word followed by the marker \$ and begin with M's read head on the first symbol of the input word and M in superposition q_0. Also as in the measure-once case, our machine responds to reading a symbol σ by evolving its internal superposition

according to the operator U_σ and, if possible, moving its read head forward. The distinction is that, after every such transformation, the MM-QFA performs a measurement with respect to the observable $\mathcal{H}(Q) = \mathcal{H}(Q_{\text{acc}}) \oplus \mathcal{H}(Q_{\text{non}}) \oplus \mathcal{H}(Q_{\text{rej}})$. If the observed value is acc, M halts and accepts; if rej, the machine halts without accepting. Otherwise, the machine will continue to operate unless the end of the tape has been reached, in which case it will halt without accepting [6].

Having defined MM-QFAs, we are of course interested in knowing whether it was worth the bother: that is, whether they are actually more powerful than the MO-QFAs. This question was addressed in 1998 by Ambainis and Freivalds, who concluded that MO-QFAs are strictly less powerful, in a terse note explaining their decision to focus on MM-QFAs [1]. As they point out, it is not difficult to simulate any MO-QFA exactly with an appropriately-constructed MM-QFA, and they cite the non-empty finite languages as examples recognizable by MM-QFAs but not MO-QFAs [1]. Therefore, we are justified in giving them further consideration.

4.2 Associated language classes

As in the measure-once case, Brodsky and Pippenger defined **RMM** to be the class of languages accepted by MM-QFAs with bounded error and **UMM** the class with unbounded error. Moreover, they gave a family of subclasses $\mathbf{RMM}_\delta$ parameterized by $\delta > 0$; $\mathbf{RMM}_\delta$ is the class of languages accepted by MM-QFAs with margin at least δ, so that $\mathbf{RMM} = \bigcup_{\delta>0} \mathbf{RMM}_\delta$ [3]. Although we adopt their notation, we do so somewhat anachronistically; unlike Moore and Crutchfield, who were interested in generalizing the idea of a language to that of a quantum language and approached their machines through that lens [7], Kondacs and Watrous worked in terms of bounded-error acceptance from the beginning [6], so in the measure-many case there is a body of work related to these language classes which predates Brodsky and Pippenger.

The first of these results, those due to Kondacs and Watrous themselves, are not at all encouraging. Specifically, these authors proved that **RMM** is a subset of the class of regular languages, and moreover they produced an example, the language on $\{a, b\}$ of words ending in a, of a language which is regular but not recognizable with bounded error by a MM-QFA and thus demonstrates that the containment is strict [6].

Therefore, we have again fallen short of the capabilities of the DFAs. However, for MM-QFAs the situation is not quite as dire as it was for MO-QFAs. Whereas the set of MO-QFAs which are also classical machines are just as powerful as the class of all MO-QFAs in terms of bounded-error recognition [3], and the same is not true for MM-QFAs. Ambainis and Freivalds proved that, although any MM-QFA which gives the correct answer with probability at least 7/9 can be simulated by a classical machine called a **reversible finite automaton**, there exist languages which can be accepted by MM-QFAs with probability of correctness over 0.65 and which cannot be recognized by reversible finite automata [1]. Since the reversible finite automata correspond to the MM-QFAs

which accept or reject deterministically and thus are classical [3], this demonstrates that MM-QFAs have advantages over some classical analogues, if not the DFAs.

Brodsky and Pippenger provided some additional results which shed light on the properties of **RMM**. Specifically, they showed that this class is closed under complement, inverse homomorphism, and another operation called **word quotient**, demonstrated that it is *not* closed under homomorphism, and produced a necessary condition on a regular language's minimal DFA for it to be in **RMM** [3]. However, as they note, they were not able to reach a conclusion regarding the closure properties of **RMM** and **UMM** with respect to Boolean operations such as union and intersection.

5 Enhanced models

Although this invites further study of the theoretical properties of MM-QFAs, we have already noted that they were known at their inception to be less powerful than the DFAs, and in particular not to recognize the language of words ending in a certain letter [6]. Therefore, we turn our attention to models which extend the automata's capabilities in some significant but reasonable way.

5.1 Multi-letter automata

In 2007, Belovs et al. proposed the idea of **multi-letter automata**, which are similar to automata of the usual sort but, for some fixed k, have access to the last k letters they have read at any given point (initially, the "letters" other than the initial symbol of the word which are taken into account are all a special empty-letter symbol Λ) [2]. Using this idea, they introduced the **k-letter deterministic finite automata, quantum finite automata, and group finite automata**, which they denoted DFA_k, QFA_k, and GFA_k respectively. However, since their initial model of a quantum finite automaton was measure-once [2], we instead adopt the notation MO-QFA_k and refer to the **k-letter measure-once quantum finite automata**.

The authors show that the language given by the regular expression $(a \cup b)^* a$, to which we have previously referred on a number of occasions, can be recognized by a GFA_2 and so is within the reach of a MO-QFA_k for any $k \geq 2$. However, the languages recognizable by MO-QFA_ks for any k are still a proper subset of the regular languages, and in fact a regular language is recognizable by some MO-QFA_k if and only if it does not contain the following construction: a pair of distinct states q and q' and a pair of nonempty words t and z such that processing z takes both q and q' to q' and processing t takes each state to itself [2]. This construction is illustrated in Figure 2.

Belovs et al. demonstrated that the union of the classes of languages recognized by MO-QFA_ks over all $k \geq 1$ is closed under intersection, union, and complement, but not Kleene star or concatenation [2]. In 2009, Qiu and Yu expanded on this analysis by observing that, for any $k \geq 1$, the MO-QFA_ks

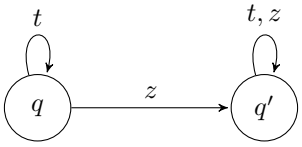

Figure 2: The forbidden construction of [2]

are strictly less powerful than the MO-QFA$_{k+1}$s [8]. These authors also demonstrated that the language a^*b^*, which can be recognized by a MM-QFA, is not recognizable by a MO-QFA$_k$ for any k; therefore, neither MM-QFAs nor MO-QFA$_k$s are strictly better, but, as Qiu and Yu also demonstrated, there exist regular languages not recognizable by either [8].

Although Belovs et al. raise the possibility of defining the MM-QFA$_k$s [2], neither they nor Qiu and Yu explore these machines' properties.

5.2 Multihead automata

One of the most recent developments in the field is the introduction by Ganguly et al. of **multihead automata** [4]. These machines are based on the MM-QFAs and work in essentially the same way, but with important distinctions. To begin with, they require a start-of-tape symbol ¢ as well as the end-of-tape symbol \$; although Kondacs and Watrous originally included such a symbol as well [6], Brodsky and Pippenger showed that it was unnecessary [3], but no such analysis has been performed in the multihead case. A more important distinction is that the machine, per the name, has multiple input heads, each of which can move to the right or remain stationary at every step [4]. To determine how the heads should move, we measure the observable corresponding to the state of the machine at every step, then move the heads according to the state the machine collapses into [4]. Unlike in the traditional MM-QFA case, we have no explicit rejecting states; instead, we halt whenever the next transition is not defined, and accept based on whether we have ended up in an accepting state [4].

Ganguly et al. were able to show that even 2-head quantum finite automata are able to accept a variety of regular languages, as well as some non-regular languages, even including an example which is not context-free [4]. However, they have as of yet been unable to prove that this model accepts all regular languages [4].

However, by observing with such exactness at each time step, this model ceases to be properly quantum; instead it is simply probabilistic, since the machine spends no meaningful time in any non-classical state. Moreover, it involves more complication of our usual notion of an automaton than is ideal; we would rather not have to add additional heads simply to define the quantum version of a DFA.

6 Closing remarks

We have given a basic overview of the major models of quantum finite automata, including the measure-once and measure-many variants, explicated their known capabilities, and examined two proposed extensions which yield additional power. Many of these models are in need of further study, with major unanswered questions about their computational power. However, as we have seen, all of them except one which has a minimally important quantum component are unable to recognize the regular languages with bounded error, which makes them somewhat unexciting as objects of study.

Therefore, although it is somewhat beyond the scope of this survey, we are unable to resist throwing our hat into the ring by defining as a candidate for further study the notion of a **repeating quantum finite automaton**, or RQFA, building upon the MM-QFAs of Kondacs and Watrous. The basic insight is that, in addition to halting and accepting or rejecting, we allow the machine to decide at any point that the computation has failed and restart it from the beginning. This will allow us to sink excess information into "junk states" at the high cost of the computation time added by the possibility of repetition.

Formally, we define a RQFA as a septuple $M = (Q, \Sigma, \delta, q_0, Q_{\text{acc}}, Q_{\text{rej}}, Q_{\text{jun}})$, where the entries are as follows:

- Q is the finite set of classical states; we define $\mathcal{H}(Q)$ as before.
- Σ is the finite input alphabet; as before, we let $\Gamma = \Sigma \sqcup \{\$\}$ be the tape alphabet.
- $\delta : Q \times \Gamma \times Q \to \mathbb{C}$ is the transition function. As usual, we require that the associated matrices U_σ be unitary.
- $q_0 \in Q$ is the classical start state.
- $Q_{\text{acc}}, Q_{\text{rej}}, Q_{\text{jun}} \subseteq Q$ are the accepting, rejecting, and **junk states** of M respectively. These are required to be disjoint, and we define the non-halting states by $Q_{\text{non}} = Q \setminus (Q_{\text{acc}} \sqcup Q_{\text{rej}} \sqcup Q_{\text{jun}})$. The associated subspaces of $\mathcal{H}(Q)$ are defined as usual. We require that $q_0 \in Q_{\text{non}}$ and that, for any input word w, there is a non-zero probability that the machine halts without repeating the computation when processing w.

Then M operates by beginning in superposition q_0 with the input word on the tape followed by \$ and the read head pointing to the input word's first letter. For each symbol $\sigma \in \Gamma$ that it reads, M evolves its internal superposition according to U_σ and then measures the observable $\mathcal{H}(Q) = \mathcal{H}(Q_{\text{acc}}) \oplus \mathcal{H}(Q_{\text{rej}}) \oplus \mathcal{H}(Q_{\text{jun}}) \oplus \mathcal{H}(Q_{\text{non}})$. If the result is acc, it halts and accepts. If rej, it halts without accepting. If jun, M returns to its initial configuration and starts over entirely. Finally, if non is observed, the machine either moves its read head along the tape and continues or, if that is not possible, halts without accepting.

Our last condition on M makes the probability that it will loop forever negligible, since as we increase the number of repetitions to infinity the probability of

nontermination will tend exponentially to 0. However, although this condition will be clear in the example we define, it is not intuitively obvious how easily it might be checked for an arbitrary proposed RQFA, which is one of this model's major limitations.

However, it *is* possible to recognize the regular languages without any error using these machines. Suppose we have a DFA $M = (S, \Sigma, \gamma, q_0, A)$ and define a RQFA $M' = (Q, \Sigma, \gamma', q_0, \{a\}, \emptyset, Q_{\text{jun}})$ to simulate it as follows. Let $q_0, \ldots, q_{n-1}$ be an enumeration of the elements of S and, for each pair $i \neq j \in \{0, 1, \ldots, n-1\}$, define a state $r_{\{i,j\}}$ which is indexed by both elements irrespective of order. Let $Q_{\text{jun}} = \{r_{\{i,j\}} \mid i \neq j \in \{0, 1, \ldots, n-1\}\}$ and $Q = S \sqcup \{a\} \sqcup Q_{\text{jun}}$. Moreover, define δ' such that, for each $\sigma \in \Sigma$, the transition matrix U_σ is as follows.

Define $c : \{0, 1, \ldots, n-1\}^2 \to \mathbb{C}$ by

$$c(i,j) = \begin{cases} -1 & \text{if } i < j \\ 0 & \text{if } i = j \\ 1 & \text{if } i > j \end{cases}$$

and, for each $q \in Q$, let $K_\sigma(q) = \{j \in \{0, 1, \ldots, n-1\} \mid \delta(q_j, \sigma) = q\}$. Then we will require for each $0 \leq i < n$ that

$$U_\sigma(q_i) = \sqrt{\frac{1}{|K_\sigma(\delta(q_i, \sigma))|}} \left(\delta(q_i, \sigma) + \sum_{j \in K_\sigma(\delta(q_i,\sigma)) \setminus \{i\}} c(i,j) r_{\{i,j\}} \right).$$

Although this formula is somewhat involved, its basic justification is that it finds a way to respect the original transition function δ while using the junk states to keep the columns of U_σ corresponding to S orthonormal; each $r_{\{i,j\}}$ provides an interaction term that we can use to enforce orthogonality between the outputs at any states whose indices are in the same class $K_\sigma(q)$. Since the vectors to which elements of S are mapped are orthonormal, we can obtain an orthonormal basis for $\mathcal{H}(Q)$ which contains all of them; we conclude our definition of U_σ by assigning the other basis vectors arbitrarily to inputs from $Q \setminus S$.

Therefore, the U_σ are unitary, and, if we extend our definition of δ to a function from Γ to $S \sqcup \{a\}$ by

$$\delta(q, \$) = \begin{cases} a & \text{if } q \in A \\ q & \text{if } q \in S \setminus A \end{cases},$$

the same methodology gives us a unitary transition matrix $U_\$$.

We can see that the machine M' will, with nonzero probability, model exactly the transitions followed by M with the same input word and then, when it encounters \$, either accept or reject as M would with some nonzero probability and redo the computation otherwise. Therefore, each repetition of the computation will have nonzero probability of interpreting the input word correctly, and we can see that it will always simply start from the beginning again if it does not. Therefore, as the number of repetitions increases to infinity, the

probability that M' has not terminated with the correct answer tends to zero exponentially, so the machine will eventually terminate and accept or reject the word correctly.

However, some basic reasoning about the infinite sums involved, here elided, tells us that the expected number of repetitions required is in the worst case exponential in the length of the input word, so it is unclear how useful these automata would be in practice. However, they do provide a simple quantum analogue to the DFAs capable of recognizing the regular languages, so we believe them to be deserving of further inquiry.

References

[1] A. Ambainis, R. Freivalds, One-way quantum finite automata: Strengths, weaknesses and generalizations, in: Proceedings of the 39th Annual Symposium on Foundations of Computer Science, IEEE Computer Society Press, Palo Alto, CA, USA, 1998, pp. 332-341. Also quant-ph/9802062, 1998.

[2] Belovs A., Rosmanis A., Smotrovs J. (2007) Multi-letter Reversible and Quantum Finite Automata. In: Harju T., Karhumki J., Lepist A. (eds) Developments in Language Theory. DLT 2007. Lecture Notes in Computer Science, vol 4588. Springer, Berlin, Heidelberg.

[3] A. Brodsky, N. Pippenger, Characterizations of 1-way quantum finite automata, SIAM Journal on Computing 31(5) (2002) 1456-1478. Also Technical Report TR-99-03, University of British Columbia, 1999.

[4] D. Ganguly, K. Chatterjee, K.S. Ray, 1-Way Multihead Quantum Finite State Automata, Applied Mathematics 7 (2016) 1005-1022.

[5] L.K. Grover, A fast quantum mechanical algorithm for database search, in: Proceedings of the 28th Annual ACM Symposium on Theory of Computing, Philadelphia, PA, USA, 1996, pp. 212-219.

[6] A. Kondacs, J. Watrous, On the Power of Quantum Finite State Automata, Proceedings of the 38th IEEE Conference on Foundations of Computer Science (1997) 66-75.

[7] C. Moore, J. Crutchfield, Quantum automata and quantum grammars, Theoretical Computer Science 237 (1-2) (2000) 275-306. Also Santa Fe Institute Working Paper 97-07-062, 1997.

[8] D. Qiu, S. Yu, Hierarchy and equivalence of multi-letter quantum finite automata, Theoretical Computer Science 410 (2009) 3006-3017.

[9] E. Rich, Automata, Computability, and Complexity: Theory and Applications, Pearson Prentice Hall (2008).

[10] B. Schumacher, Quantum coding, Physical Review A 51(4) (1995) 2738-2747.

[11] P.W. Shor, Algorithms for quantum computation: discrete logarithms and factoring, Proc. 35th Symp. on Foundations of Computer Science, 1994, pp. 124-134.