ELSEVIER

Theoretical Computer Science 237 (2000) 275–306

Theoretical Computer Science

www.elsevier.com/locate/tcs

Quantum automata and quantum grammars

Cristopher Moore, James P. Crutchfield

Santa Fe Institute, 1399 Hyde Park Road, Santa Fe, NM 87501, USA

Received September 1997; revised June 1998
Communicated by M. Nivat

Abstract

To study quantum computation, it might be helpful to generalize structures from language and automata theory to the quantum case. To that end, we propose quantum versions of finite-state and push-down automata, and regular and context-free grammars. We find analogs of several classical theorems, including pumping lemmas, closure properties, rational and algebraic generating functions, and Greibach normal form. We also show that there are quantum context-free languages that are not context-free, so **QCFL** $\neq$ **CFL**.

Keywords: Quantum computation; Grammars; Automata; Complexity classes

1. Introduction

Nontraditional models of computation – such as real-valued, analog, spatial, molecular, stochastic, and quantum computation – have received a great deal of interest in both physics and computer science in recent years (e.g. [1, 4, 10, 21, 8, 31, 9]). This stems partly from a desire to understand computation in dynamical systems, such as ordinary differential equations, iterated maps, cellular automata, and recurrent neural networks, and partly from a desire to circumvent the fundamental limits on current computing technologies by inventing new computational model classes.

Quantum computation, in particular, has become a highly active research area. This is driven by the recent discovery of quantum algorithms for factoring that operate in polynomial time [29], the suggestion that quantum computers can be built using familiar physical systems [7, 14, 19], and the hope that errors and decoherence of the quantum state can be suppressed so that such computers can operate for long times [30, 33].

If we are to understand computation in a quantum context, it might be useful to translate as many concepts as possible from classical computation theory into the

E-mail addresses: moore@santafe.edu (C. Moore), jpc@santafe.edu (J.P. Crutchfield).

0304-3975/00/$ - see front matter
PII: S0304-3975(98)00191-1

quantum case. From a practical viewpoint, we might as well start with the lowest levels in the computational hierarchy and work upward. In this paper we begin in just this way by defining quantum versions of the simplest language classes – the regular and context-free languages [16].

To do this, we define quantum finite-state and push-down automata (QFAs and QPDAs) as special cases of a more general object, a *real-time quantum automaton*. In this setting a formal language becomes a function that assigns quantum probabilities to words. We also define *quantum grammars*, in which we sum over all derivations to find the amplitude of a word. We show that the corresponding languages, generated by quantum grammars and recognized by quantum automata, have pleasing properties in analogy to their classical counterparts. These properties include pumping lemmas, closure properties, rational and (almost) algebraic generating functions, and Greibach normal form.

For the most part, our proofs simply consist of tracking standard results in the theory of classical languages and automata, stochastic automata, and formal power series, and attaching complex amplitudes to the transitions and productions of our automata and grammars. In a few places – notably, Lemmas 12 and 13 and Theorems 6, 7, 10, 19, 23, 24, and 25 – we introduce more original ideas.

We believe that this strategy of starting at the lowest rungs of the Chomsky hierarchy has several benefits. First, we can prove that low-lying classical and quantum computational models are different; for instance, we show here that **QCFL** $\neq$ **CFL**, i.e. that there are quantum context-free languages that are not classically context-free. Such separations are difficult to prove for more powerful models such as deterministic vs. quantum polynomial time, since they rely partly on unproved classical separations such as **P** vs. **NP**.

Second, studying the computational power of a physical system can give detailed insights into a natural system's structure and dynamics. For example, it may be the case that the spatial density of physical computation is finite. In this case, every finite quantum computer is actually a QFA. If a system does, in fact, have infinite memory, it makes sense to ask what kinds of long-time correlations it can have, such as whether its memory is stack-like or queue-like. Our QPDAs provide a way to formalize these questions.

Molecular biology suggests another example along these lines, the class of protein secondary structures coded for by RNA. To some approximation the long-range correlations between RNA nucleotide base pairs responsible for secondary structure can be modeled by parenthesis-matching grammars [28, 27]. Since RNA macromolecules are quantum mechanical objects, constructed by processes that respect atomic and molecular quantum physics, the class of secondary structures coded for by RNA may be more appropriately modeled by the quantum analogs of context-free grammars introduced here. In the same vein, DNA and RNA nucleotide sequences are recognized and manipulated by various active molecules (e.g. transcription factors and polymerases). Could their functioning be modeled by QFAs and QPDAs?

Finally, the theory of context-free languages has been extremely useful in designing compilers, parsing algorithms, and programming languages for classical computers. Is it possible that quantum context-free languages can play a similar role in the design of quantum computers and algorithms?

1.1. Quantum mechanics

First, we give a brief introduction to quantum mechanics [34].

A quantum system's state is described by a vector of complex numbers. The dimension of a quantum system is the number of complex numbers in its state vector. A column vector is written $|a\rangle$ and its *Hermitian conjugate* $|a\rangle^\dagger$, the complex conjugate of its transpose, is the row vector $\langle a|$. These vectors live in a *Hilbert space* H, which is equipped with an inner product $a \cdot b = \langle a|b\rangle$. The probability of observing a given state a is its norm $|a|^2 = \langle a|a\rangle$.

Over time, the dynamics of a quantum system rotates the state $|a\rangle$ in complex vector space by a *unitary* matrix U – one whose inverse is equal to its Hermitian conjugate, $U^\dagger = U^{-1}$. Then the total probability of the system is conserved, since if $\langle a'| = \langle a|U$, then $\langle a'|a'\rangle = \langle a|U^\dagger U|a\rangle = \langle a|a\rangle$.

The eigenvalues of a unitary matrix are of the form $e^{i\omega}$, where ω is a real-valued angle, and so are restricted to the unit circle in the complex plane. Thus, the dynamics of an n-dimensional quantum system, which is governed by an $n \times n$ unitary matrix, is simply a rotation in $\mathbb{C}^n$. In the Schrödinger equation, U is determined by the *Hamiltonian* or energy operator $\mathscr{H}$ via $U = e^{i\mathscr{H}t}$.

A measurement consists of applying an operator O to a quantum state a. We will write operators on the right, $\langle a|O$. To correspond to a classical observable, O must be *Hermitian*, $O^\dagger = O$, so that its eigenvalues are real and so "measurable". If one of its eigenvalues λ is associated with a single eigenvector u_λ, then we observe the value λ with a probability $|\langle a|u_\lambda\rangle|^2$, where $\langle a|u_\lambda\rangle$ is the component of a along u_λ.

More generally, if there is more than one eigenvector u_λ with the same eigenvalue λ, then the probability of observing $O = \lambda$ when the system is in state a is $|aP_\lambda|^2$, where P_λ is a *projection* operator such that $\langle u_\mu|P_\lambda = \langle u_\mu|$ if $\mu = \lambda$ and 0 otherwise. Thus, P_λ projects a onto the subspace of H spanned by the u_λ.

For instance, suppose that we consider a two-dimensional quantum system with Hamiltonian

$$\mathscr{H} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

Then

$$U = \begin{pmatrix} e^{it} & 0 \\ 0 & e^{-it} \end{pmatrix}.$$

The eigenvectors of $\mathscr{H}$ are $\binom{1}{0}$ and $\binom{0}{1}$, with eigenvalues $+1$ and -1, respectively. If the system is in the state $\langle a| = (\sqrt{3}/2, -i/2)$, a measurement of the energy $\mathscr{H}$ will

yield $+1$ or -1 with probabilities $\frac{3}{4}$ and $\frac{1}{4}$, respectively. The projection operators are

$$P_{+1} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \quad \text{and} \quad P_{-1} = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}.$$

1.2. Classical finite automata and regular languages

Readers familiar with basic automata theory should skip this section and the next two. An introduction can be found in [16].

If A is an *alphabet* or set of symbols, A^* is the set of all finite sequences or *words* over A and a *language* L over A is a subset of A^*. If w is a word, then $|w|$ is its length and w_i is its ith symbol. We denote the empty word by ε, the concatenation of two words u and v as uv, and w repeated k times as w^k.

A *deterministic finite-state automaton* (DFA) consists of a finite set of states S, an input alphabet A, a transition function $F : S \times A \to S$, an initial state $s_{\text{init}} \in S$, and a set of accepting states $S_{\text{accept}} \subset S$. The machine starts in s_{init} and reads an input word w from left to right. At the ith step, it reads a symbol w_i and updates its state to $s' = F(s, w_i)$. It accepts w if the final state reached after reading $w_{|w|}$ is in S_{accept}. We say the machine *recognizes* the language of accepted words.

A *nondeterministic finite-state automaton* (NFA) has a transition function into the power set of A, $F : S \times A \to \mathcal{P}(A)$, so that there may be several transitions the machine can make for each symbol. An NFA accepts if there is an allowed *computation path*, i.e. a series of allowed transitions, that leads to a state in S_{accept}.

As it turns out, DFAs and NFAs recognize exactly the same languages, since an NFA with a set of states S can be simulated by a DFA whose states correspond to subsets of S. If a language can be recognized by a DFA or NFA, it is called *regular*.

For instance, the set of words over $A = \{a, b\}$ where no two b's occur consecutively is regular. If $S = \{A, B, R\}$, $s_{\text{init}} = A$, $S_{\text{accept}} = \{A, B\}$, and

$$F(A, a) = F(B, a) = A, \quad F(A, b) = B,$$

$$F(B, b) = R, \quad F(R, a) = F(R, b) = R,$$

then we enter the 'reject' state R, and stay there, whenever we encounter the string bb. A, S, s_{init}, S_{accept}, and F constitute a DFA.

One way to view finite-state automata is with matrices and vectors. If an NFA has n states, the set of allowed transitions can be described by an $n \times n$ transition matrix M_a for each symbol $a \in A$, in which $(M_a)_{ij} = 1$ if and only if the transition from state i to state j is allowed on reading a. Then if $\boldsymbol{s}_{\text{init}}$ is the n-component column vector

$$(\boldsymbol{s}_{\text{init}})_i = \begin{cases} 1, & i = s_{\text{init}}, \\ 0, & \text{otherwise}, \end{cases}$$

and $\boldsymbol{P}_{\text{accept}}$ is the column vector

$$(\boldsymbol{P}_{\text{accept}})_i = \begin{cases} 1, & i \in S_{\text{accept}}, \\ 0, & \text{otherwise}, \end{cases}$$

then the number of accepting paths on an input w is

$$f(w) = \boldsymbol{s}_{\text{init}}^{\mathrm{T}} \cdot M_w \cdot \boldsymbol{P}_{\text{accept}}, \tag{1}$$

where M_w is shorthand for $M_{w_1} M_{w_2} \ldots M_{w_{|w|}}$. Then a word w is accepted if $f(w) > 0$, so that there is some path leading from $\boldsymbol{s}_{\text{init}}$ to the accepting subspace spanned by $s \in S_{\text{accept}}$. (We apply the matrices on the right, so that they occur in the same order as the symbols of w, instead of in reverse.) Of course, M_ε is the identity matrix, which we will denote $\mathbf{1}$.

Eq. (1) will be our starting point for defining quantum versions of finite-state automata and regular languages.

1.3. Push-down automata and context-free languages

A *push-down automaton* (PDA) is a finite-state automaton or 'control' that also has access to a *stack*, an infinite memory storing a string of symbols in some alphabet T. Its transition function $F : S \times T \times A \to \mathscr{P}(S \times T^*)$ allows it to examine its control state, the top stack symbol, and the input symbol. It then updates its control state, pops the top symbol off the stack, and pushes a (possibly empty) word onto the stack. A PDA starts with an initial state and stack configuration. After reading a word, it accepts if a computation path exists that either ends in an accepting control state or produces an empty stack.

PDAs recognize the *context-free* languages (CFLs), a name whose motivation will become clear in a moment. For instance, the Dyck language of properly nested words of brackets $\{\varepsilon, (), (()), ()(), (()()), \ldots\}$ is context-free. It is recognized by a PDA with a single stack symbol x. This PDA pushes an x onto the stack when it sees a "(" and pops one off when it sees a ")". If it ever attempts to pop a symbol off an empty stack, it enters the reject state and stays there.

A *deterministic push-down automaton* (DPDA) is one with at most one allowed transition for each combination of control state, stack symbol, and input symbol. DPDAs recognize the *deterministic context-free* languages (DCFLs), such as the Dyck language above.

1.4. Grammars, context-free and regular

A *grammar* consists of two alphabets V and T, the *variables* and *terminals*, an initial variable $I \in V$, and a set P of *productions* $\alpha \to \beta$ where $\alpha \in V^*$ and $\beta \in (V \cup T)^*$. A *derivation* $\alpha \Rightarrow \beta$ is a chain of strings, where at each step one substring is replaced with another according to one of the productions. Then the language generated by the grammar consists of those strings in T^* (consisting only of terminals) that can be derived from I with a chain of productions in P.

For example, the grammar $V = \{I\}$, $T = \{(,)\}$, and $P = \{I \to (I)I, I \to \varepsilon\}$ generates the Dyck language. Note that the left-hand side of each production consists of a single symbol and does not require any neighboring symbols to be present; hence the term *context-free*. Context-free grammars generate exactly the languages recognized by PDAs.

The Dyck language grammar is *unambiguous* in that every word has a unique derivation tree. A context-free language is unambiguous if there is an unambiguous grammar that generates it. Notably, there are *inherently ambiguous* context-free languages for which no unambiguous grammar exists.

If we restrict a grammar further so that every production is of the form $v_1 \to wv_2$ or $v_1 \to w$, where $w \in T^*$ and $v_1, v_2 \in V$, then there is never more than one variable present in the string. The result is that a derivation leaves strings of terminals behind the variable as it moves to the right. Such grammars are called *regular* and generate exactly the regular languages.

1.5. Quantum languages and automata

Since quantum systems predict observables in a probabilistic way, it makes sense to define a *quantum language* as a function mapping words to probabilities, $f : A^* \to [0,1]$. This generalizes the classical Boolean situation where each language has a *characteristic function* $\chi_L : A^* \to \{0,1\}$, defined as $\chi_L(w) = 1$ if $w \in L$ and 0 otherwise. (In fact, in order to compare our quantum language classes with the classical ones, we will occasionally abuse our terminology by identifying a Boolean language with its characteristic function, saying that a language is in a given class if its characteristic function is.)

Then in analogy to Eq. (1), we define quantum automata in the following way:

Definition. A *real-time quantum automaton* (QA) Q consists of
– a Hilbert space H,
– an initial state vector $\langle s_{\text{init}}| \in H$ with $|s_{\text{init}}|^2 = 1$,
– a subspace $H_{\text{accept}} \subset H$ and an operator P_{accept} that projects onto it,
– an input alphabet A, and
– a unitary transition matrix U_a for each symbol $a \in A$.

Then using the shorthand

$$U_w = U_{w_1} U_{w_2} \cdots U_{w_{|w|}},$$

we define the quantum language recognized by Q as the function

$$f_Q(w) = |s_{\text{init}} U_w P_{\text{accept}}|^2$$

from words in A^* to probabilities in $[0,1]$. (Again, we apply linear operators on the right, so that the symbols w_i occur in left-to-right order.)

In other words, we start with $\langle s_{\text{init}}|$, apply the unitary matrices U_{w_i} for the symbols of w in order, and measure the probability that the resulting state is in H_{accept} by applying the projection operator P_{accept} and measuring the norm. This is a *real-time* automaton since it takes exactly one step per input symbol, with no additional computation time after the word is input.

Physically, this can be interpreted as follows. We have a quantum system prepared in a superposition of initial states. We expose it over time to different influences depending

on the input symbols, one time-step per symbol. At the end of this process, we perform a measurement on the system and $f(w)$ is the probability of this measurement having an acceptable outcome, such as being in a given energy level.

Note that f is not a measure on the space of words. It is the probability of a particular measurement after a given input.

This basic setting is not new. If we restrict ourselves to real rather than complex values and replace unitarity of the transition matrices with *stochasticity* in which the elements of each row of the U_a sum to 1, we get the *stochastic automata* of Rabin [24]; see also the review in [20]. If we generalize the U_a to nonlinear maps in $\mathbb{R}^n$, we get *real-time dynamical recognizers* [22]. If we generalize the U_a to nonlinear Bayes-optimal update maps of the n-simplex, we get *ε-machine* deterministic representations of recurrent hidden Markov models [8, 36].

Note that the effect of the matrix product $U_w = U_{w_1} U_{w_2} \ldots$ is to sum over all possible paths that the machine can take. Each path has a *complex amplitude* equal to the product of the amplitudes of the transitions at each step. Each of U_w's components, representing possible paths from an initial state s_0 to a final state $s_{|w|}$, is the sum of these. That is,

$$(U_w)_{s_0,s_{|w|}} = \sum_{s_1,s_2,\ldots,s_{|w|-1}} (U_{w_1})_{s_0,s_1} (U_{w_2})_{s_1,s_2} \ldots (U_{w_{|w|}})_{s_{|w|-1},s_{|w|}}$$

over all possible choices of the intervening states $s_1, \ldots, s_{|w|-1}$. The difference from the real-valued (stochastic) case is that *destructive interference* can take place. Two paths can have opposite phases in the complex plane and cancel each other out, leaving a total probability less than the sum of the two, since $|a+b|^2 \leqslant |a|^2 + |b|^2$.

Note that paths ending in different perpendicular states in H_{accept} add noninterferingly, $|a|^2 + |b|^2$, while paths ending in the same state add interferingly, $|a+b|^2$. This will come up several times in discussion below.

In analogy with Turakainen's generalized stochastic automata [35] where the transition matrices do not necessarily preserve probability, we will sometimes find it useful to relax unitarity:

Definition. A *generalized real-time quantum automaton* is one in which the matrices U_a are not necessarily unitary and the norm of the initial state s_{init} is not necessarily 1.

We can then define different classes of quantum automata by restricting the Hilbert space H and the transition matrices U_a in various ways: first to the finite-dimensional case and then to an infinite memory in the form of a stack.

2. Quantum finite-state automata and regular languages

The quantum analog of a finite-state machine is a system with a finite-dimensional state space, so

Definition. A *quantum finite-state automaton* (QFA) is a real-time quantum automaton where H, s_{init}, and the U_a all have a finite dimensionality n. A *quantum regular language* (QRL) is a quantum language recognized by a QFA.

In this section, we will try to reproduce as many results as possible on classical regular languages in the quantum case.

2.1. Closure properties of QRLs

First, we define two operations on quantum automata that allow us to add and multiply quantum languages. The result is that the set of QRLs is closed under these operations, just as stochastic languages are [20, 23].

Definition. If u and v are vectors of dimension m and n, respectively, their *direct sum* $u \oplus v$ is the $(m+n)$-dimensional vector $(u_1, \ldots, u_m, v_1, \ldots, v_n)$. If M and N are matrices, then

$$M \oplus N = \left(\begin{array}{c|c} M & 0 \\ \hline 0 & N \end{array}\right).$$

Then if Q and R are quantum automata with the same input alphabet, and if a and b are complex numbers such that $|a|^2 + |b|^2 = 1$, the *weighted direct sum* $aQ \oplus bR$ has initial state $s'_{\text{init}} = as^Q_{\text{init}} \oplus bs^R_{\text{init}}$, projection operator $P'_{\text{accept}} = P^Q_{\text{accept}} \oplus P^R_{\text{accept}}$, and transition matrices $U'_x = U^Q_x \oplus U^R_x$ for each input symbol x.

Lemma 1. *If Q and R are QFAs and if $|a|^2 + |b|^2 = 1$, then $aQ \oplus bR$ is a QFA and $f_{aQ \oplus bR} = |a|^2 f_Q + |b|^2 f_R$. Therefore, if $f_1, f_2, \ldots, f_k$ are QRLs, then $\sum_{i=0}^k c_i f_i$ is a QRL for any real constants $c_i > 0$ such that $\sum_{i=0}^k c_i = 1$.*

Proof. Clearly $|s'_{\text{init}}|^2 = |as^Q_{\text{init}}|^2 + |bs^R_{\text{init}}|^2 = |a|^2 + |b|^2 = 1$. The direct sum of two subspaces is a subspace, the direct sum of unitary matrices is unitary, and the direct sum of two finite-dimensional quantum automata is finite-dimensional, so $aQ \oplus bR$ is a QFA.

Furthermore, $U'_w = U^Q_w \oplus U^R_w$ and

$$f_{aQ \oplus bR}(w) = |as^Q_{\text{init}} U^Q_w P^Q_{\text{accept}}|^2 + |bs^R_{\text{init}} U^R_w P^R_{\text{accept}}|^2 = |a|^2 f_Q(w) + |b|^2 f_R(w).$$

(Note that the phases of a and b do not matter, only their norms.) By induction we can sum any k QRLs in this way, as long as $\sum_{i=0}^k c_i = 1$. □

Definition. If u and v are vectors of dimension m and n, respectively, then their *tensor product* $u \otimes v$ is the mn-dimensional vector $w_{\langle i,j \rangle} = u_i v_j$ where $\langle i, j \rangle = n(i-1) + j$, say, is a pairing function. If M and N are $m \times m$ and $n \times n$ matrices, $M \otimes N$ is the $mn \times mn$ matrix $O_{\langle i,k \rangle, \langle j,l \rangle} = M_{ij} N_{kl}$. Then if Q and R are quantum automata with the same input alphabet, $Q \otimes R$ is defined by taking the tensor products of their respective s_{init}, P_{accept}, and the U_a.

Lemma 2. *If Q and R are QFAs, then $Q \otimes R$ is a QFA and $f_{Q\otimes R} = f_Q f_R$. Therefore, the product of any number of QRLs is a QRL.*

Proof. It is easy to show that if a and c are m-dimensional vectors and b and d are n-dimensional vectors, then $\langle a \otimes b | c \otimes d \rangle = \langle a|c\rangle\langle b|d\rangle$. Therefore $|s'_{\text{init}}|^2 = |s^Q_{\text{init}}|^2 \, |s^R_{\text{init}}|^2 = 1$. The tensor product of finite-dimensional unitary matrices is unitary and finite-dimensional, so $Q \otimes R$ is a QFA.

Furthermore, $U'_w = U^Q_w \otimes U^R_w$ and

$$f_{Q\otimes R}(w) = |s^Q_{\text{init}} U^Q_w P^Q_{\text{accept}}|^2 \cdot |s^R_{\text{init}} U^R_w P^R_{\text{accept}}|^2 = f_Q(w) f_R(w).$$

By induction we can multiply any number of QRLs in this way. □

Lemma 3. *For any $c \in [0,1]$, the constant function $f(w) = c$ is a QRL.*

Proof. Just choose any s_{init} and P_{accept} such that $|s_{\text{init}} P_{\text{accept}}|^2 = c$, and let $U_a = \mathbf{1}$ for all a. □

Since we can add and multiply QRLs, we have

Corollary. *Let f_i be QRLs and let c_i be a set of constants such that $\sum_{i=0}^{k} c_i \leqslant 1$. Then any polynomial $\sum_j c_j g_j$, where each g_j is a product of a finite number of f_i's, is a QRL.*

In a sense, closure under (weighted) addition and multiplication are complex-valued analogs of OR and AND. Classical regular languages are closed under both these Boolean operations, as well as complementation:

Lemma 4. *If f is a QRL, then $\overline{f} = 1 - f$ is a QRL.*

Proof. Let H'_{accept} be the subspace of H perpendicular to H_{accept} and P'_{accept} the projection operator onto it. Since $P_{\text{accept}} + P'_{\text{accept}} = \mathbf{1}$, $P_{\text{accept}} P'_{\text{accept}} = 0$, the U_w are unitary, and $|s_{\text{init}}|^2 = 1$, we have

$$\begin{aligned} 1 &= |s_{\text{init}} U_w|^2 = |s_{\text{init}} U_w (P_{\text{accept}} + P'_{\text{accept}})|^2 \\ &= |s_{\text{init}} U_w P_{\text{accept}}|^2 + |s_{\text{init}} U_w P'_{\text{accept}}|^2 \\ &= f(w) + \overline{f}(w), \end{aligned}$$

where $\overline{f}(w) = |s_{\text{init}} U_w P'_{\text{accept}}|^2$. □

Another property of classical regular languages is closure under inverse homomorphism [16]:

Definition. A *homomorphism* $h : A^* \to A^*$ is a function that replaces symbols with words. For instance, if $h(a) = b$ and $h(b) = ab$, then $h(bab) = abbab$. If f is a quantum

language, then its *inverse image* under h is the language $(f \circ h)(w) = f(h(w))$. (This looks wrong, but it is in fact the proper form for the characteristic function of the inverse image of a set. Formally, the mapping from sets to characteristic functions acts like a contravariant functor.)

Lemma 5. *If f is a QRL and h is a homomorphism, then the inverse image $f \circ h$ is a QRL.*

Proof. Simply replace each U_a with $U_{h(a)}$. Recall that the composition of unitary matrices is unitary.

2.2. The pumping lemma for QRLs

The following is a well-known classical result [16]:

Lemma (Pumping lemma for regular languages). If L is a regular language, then any sufficiently long word $w \in L$ can be written $w = xyz$ such that $xy^k z \in L$ for all $k \geqslant 0$.

Proof. If an NFA has n states, then any path longer than n transitions contains a loop, which can be repeated as many times as desired. □

Because of unitarity, we have a slightly stronger result for QRLs in that any subword can be 'pumped'. However, unlike the classical case, we cannot repeat a word arbitrarily many times. Rather, the dynamics is like an irrational rotation of a circle, so that for any $\varepsilon > 0$, there is some k such that k rotations brings us back to within a distance ε from where we started.

Theorem 6 (Pumping for QRLs). *If f is a QRL, then for any word w and any $\varepsilon > 0$, there is a k such that $|f(uw^k v) - f(uv)| \leqslant \varepsilon$ for any words u, v. Moreover, if f's automaton is n-dimensional, there is a constant c such that $k \leqslant (c\varepsilon)^{-n}$.*

Proof. In its diagonal basis, U_w rotates n complex numbers on the unit circle by n different angles ω_i for $1 \leqslant i \leqslant n$. We can think of this as a rotation of a n-dimensional torus. If $V = (c\delta)^n$ is the volume of a n-dimensional ball of radius δ, then U_w^k is within a distance δ of the identity matrix for some number of iterations $k \leqslant 1/V$. We illustrate this in Fig. 1.

Then we can write $U_w^k = \mathbf{1} + \delta J$, where J is a diagonal matrix for which $\sum_{i=0}^{n} |J_{ii}|^2 \leqslant 1$, and

$$\begin{aligned} f(uw^k v) &= |s_{\text{init}} U_u (\mathbf{1} + \delta J) U_v P_{\text{accept}}|^2 \\ &\leqslant \left(|s_{\text{init}} U_u U_v P_{\text{accept}}| + \delta\, |s_{\text{init}} U_u J U_v P_{\text{accept}}| \right)^2 \\ &= f(uv) + 2\delta \sqrt{f(uv)\, j} + \delta^2 j \\ &\leqslant f(uv) + 3\delta, \end{aligned}$$

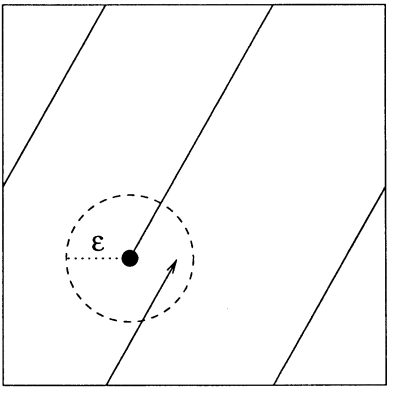

Fig. 1. Iterating the unitary matrix U_w is equivalent to rotating a torus. If a ball of radius ε has volume V, then after at most $1/V$ iterations the state must return to within a distance ε of its initial position.

since $\delta \leqslant 1$, $f(uv) \leqslant 1$, and $j \leqslant 1$ where

$$j = |s_{\text{init}} U_u J U_v P_{\text{accept}}|^2 \leqslant |s_{\text{init}}|^2 \sum_{i=0}^{n} |J_{ii}|^2.$$

We can prove $f(uw^k v) \geqslant f(uv) - 3\delta$ similarly. Then $|f(uw^k v) - f(uv)| \leqslant 3\delta$ and the theorem is proved with $\delta = \varepsilon/3$. □

If m of the angles ω_i are rational fractions $2\pi p/q$, then we return to a $(n-m)$-dimensional torus every q steps and $k \leqslant q(c\varepsilon)^{-(n-m)}$.

In the case where a unitary QFA recognizes a classical language (which we identify with its characteristic function), this gives the following:

Theorem 7. *If a regular language L is a QRL, then the transition matrices M_a of the minimal DFA recognizing L generate a group $\{M_w\}$. Therefore, there are regular languages that are not QRLs.*

Proof. Any set of matrices forms a semigroup, so we just have to show that every sequence of transitions M_w has an inverse.

Define two words as equivalent, $u \sim v$, if they can be followed by the same suffixes, $uw \in L$ if and only if $vw \in L$. It is well known [16] that the states of L's minimal DFA are in one-to-one correspondence with $\sim$'s equivalence classes.

Then if L's characteristic function χ_L is a QRL, setting $\varepsilon < 1$ in Theorem 6 shows that for every w, there exists a k such that, for all u and v,

$$\chi_L(uw^k v) = \chi_L(uv)$$

which implies $uw^k \sim u$ for all u. Then $M_w^k = \mathbf{1}$ in L's minimal DFA since it returns any u to its original equivalence class, and M_w has an inverse M_w^{k-1}. So $\{M_w\}$ is a group.

Most regular languages do not have this property. Consider the language L given in the introduction with the subword bb forbidden. Inserting bb anywhere in an allowed word makes it disallowed, and this cannot be undone by following bb with any other subword. Thus, M_{bb} has no inverse in $\{M_w\}$, and L is not a QRL. □

In contrast, in the generalized case where the U_a do not have to be unitary, we have

Lemma 8. *Any regular language is a generalized QRL.*

Proof. Let the U_a be the Boolean transition matrices of L's DFA. Then there is exactly one allowed path for each allowed word, so $f(w) = \chi_L(w)$. □

Combining this with the previous corollary gives the following:

Corollary. *The QRLs are a proper subclass of the generalized QRLs.*

2.3. QRLs are rational

In classical language theory, we are often interested in the *generating function* of a language, $g_L(z) = \sum_{w\in L} z^{|w|}$ or equivalently $\sum_n N_n z^n$, where N_n is the number of words of length n in L. More generally, if we think of the symbols $a \in A$ as noncommuting variables, we can write a formal power series $G_L = \sum_{w\in L} w$, whereupon setting $a = z$ for all $a \in A$ gives $G_L = g_L(z)$.

A beautiful theory of such series is given in [18]. In particular, the generating function of a regular language is always *rational*, i.e. the quotient of two polynomials. To see this, sum Eq. (1) over all lengths, labelling transitions with their respective symbols. Using a DFA with one computation path per word, if we define $M = \sum_{a\in A} aM_a$ and rewrite the sum over all words as a sum over all lengths, we have

$$\begin{aligned} G_L &= \sum_w (\boldsymbol{s}_{\text{init}}^{\mathrm{T}} \cdot M_w \cdot \boldsymbol{P}_{\text{accept}})\, w \\ &= \boldsymbol{s}_{\text{init}}^{\mathrm{T}} \cdot \sum_{n=0}^{\infty} M^n \cdot \boldsymbol{P}_{\text{accept}} \\ &= \boldsymbol{s}_{\text{init}}^{\mathrm{T}} \cdot (\mathbf{1} - M)^{-1} \cdot \boldsymbol{P}_{\text{accept}} \end{aligned}$$

which is rational in each symbol a since each component of $(\mathbf{1} - M)^{-1}$ is. Then restricting to $a = z$ for all a gives a rational $g_L(z)$ as well.

For instance, for the regular language given above with bb forbidden,

$$M = \begin{pmatrix} a & b \\ a & 0 \end{pmatrix}, \quad \boldsymbol{s}_{\text{init}} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

and $\boldsymbol{P}_{\text{accept}} = \mathbf{1}$. Here $\begin{pmatrix} 0 \\ 0 \end{pmatrix}$ represents the reject state. Then the reader can check that

$$(\mathbf{1} - M)^{-1} = \frac{1}{1 - a - ab} \begin{pmatrix} 1 & b \\ a & 1 - a \end{pmatrix}$$

and

$$G_L = \frac{1+b}{1-a-ab} = 1 + a + b + aa + ab + ba + \cdots,$$

where the empty word is now denoted by 1. Setting $a = b = z$ gives

$$g_L(z) = \frac{1+z}{1-z-z^2} = 1 + 2z + 3z^2 + 5z^3 + \cdots$$

recovering the well-known fact that the number of words of length n is the nth Fibonacci number.

The obvious generalization of this is

Definition. If f is a quantum language, then its *generating function* G_f is the formal sum $\sum_{w \in A^*} f(w)\, w$.

Theorem 9. *If f is a generalized QRL, then G_f is rational.*

Proof. We first consider generating functions g based on complex amplitudes rather than total probabilities. The accepting subspace H_{accept} is spanned by a finite number of perpendicular unit vectors h_i. Then if we define $g_i = \sum_w \langle s_{\text{init}} | U_w | h_i \rangle\, w$ and $U = \sum_{a \in A} a U_a$, we have

$$g_i = \langle s_{\text{init}} \,|\, (\mathbf{1} - U)^{-1} \,|\, h_i \rangle$$

and the g_i are rational.

The *Hadamard product* of two series $C = \sum_w c_w w$ and $D = \sum_w d_w w$ is the series formed by multiplying their coefficients term by term, $C \odot D = \sum_w c_w d_w w$. Since $|vP_{\text{accept}}|^2 = \sum_i |\langle v | h_i \rangle|^2$ for any vector v, i.e. the probability of being in H_{accept} is the (noninterfering) sum of the squares of the amplitudes along each of the h_i, we have

$$G_f = \sum_i g_i^* \odot g_i.$$

The class of rational series is closed under both addition and Hadamard product [18], so G_f is rational. (These closure properties are generalizations of the closure of the class of regular languages under union and intersection.) □

The theory of rational generating functions has also been used in the recognition of languages by neural networks [32].

2.4. Real representation and stochastic automata

We should investigate the relationship between quantum and real-valued stochastic automata, since the latter have been extensively studied. We alluded to the following in the introduction [23, 35]:

Definition. A *generalized stochastic function* is a function from words over an alphabet A to real numbers, $f : A^* \to \mathbb{R}$, for which there are real-valued vectors π and η and real-valued matrices M_a for each $a \in A$ such that f is a bilinear form,

$$f(w) = \pi^{\mathrm{T}} \cdot M_w \cdot \eta,$$

where $M_w = M_{w_1} M_{w_2} \ldots M_{w_{|w|}}$ as before. We will call such a function *n-dimensional* if π, η and the M_a are n-dimensional.

If the components of η are 0 and 1 denoting nonaccepting and accepting states and if π and the rows of the M_a have nonnegative entries that sum to 1 so that probability is preserved, then f is a *stochastic function*. If we allow negative entries but still require that π and the rows of the M_a sum to 1, then f is *pseudo-stochastic*.

It is well known that complex numbers $c = a + b\mathrm{i}$ can be represented by 2×2 real matrices

$$\boldsymbol{c} = \begin{pmatrix} a & b \\ -b & a \end{pmatrix}.$$

The reader can check that multiplication is faithfully reproduced and that $\boldsymbol{c}^{\mathrm{T}}\boldsymbol{c} = |c|^2\mathbf{1}$. In the same way, an $n \times n$ complex matrix can be simulated by a $2n \times 2n$ real-valued matrix. Moreover, this matrix is unitary if the original matrix is.

Using this representation, we can show the following:

Theorem 10. *Any generalized QRL recognized by an n-dimensional generalized QFA is a $2n^2$-dimensional generalized stochastic function.*

Proof. First, we transform our automaton so that the output $f(w)$ is a bilinear, rather than quadratic, function of the machine's state. As before, let h_i be a set of perpendicular unit vectors spanning H_{accept}. Then

$$\begin{aligned} f(w) &= \sum_{i=0}^{n} |\langle s_{\mathrm{init}} \mid U_w \mid h_i \rangle|^2 \\ &= \sum_{i=0}^{n} \langle s_{\mathrm{init}}^* \otimes s_{\mathrm{init}} \mid U_w^* \otimes U_w \mid h_i^* \otimes h_i \rangle \\ &= \langle s_{\mathrm{init}}^* \otimes s_{\mathrm{init}} \mid U_w^* \otimes U_w \mid \sum_{i=0}^{n} h_i^* \otimes h_i \rangle. \end{aligned}$$

This has the form $\pi^{\mathrm{T}} \cdot M_w \cdot \eta$ with $\pi = s_{\mathrm{init}}^* \otimes s_{\mathrm{init}}$, $M_a = U_a^* \otimes U_a$ for all $a \in A$, and $\eta = \sum_i h_i^* \otimes h_i$. Since these are the tensor products of n-dimensional objects, they have n^2 dimensions. However, their entries are still complex valued.

Using the representation above, we transform π^{T}, M_a, and η into $2 \times 2n^2$, $2n^2 \times 2n^2$, and $2n^2 \times 2$ real-valued matrices $\overline{\pi}^{\mathrm{T}}$, $\overline{M}_w$, and $\overline{\eta}$, respectively, and

$$\overline{\pi}^{\mathrm{T}} \cdot \overline{M}_w \cdot \overline{\eta} = f(w) \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Letting π and η be the top row of $\overline{\pi}$ and the left column of $\overline{\eta}$, respectively, gives the desired real-valued, bilinear form. □

This expression of a QRL as a generalized stochastic function gives us transition matrices that are unitary but neither stochastic nor pseudo-stochastic. A logical question, then, is whether the class of QRLs is contained in the class of stochastic functions, or vice versa, and similarly for the pseudo-stochastic functions. Since the only matrices that are both pseudo-stochastic and unitary are permutation matrices, it seems more likely that the QRLs are incomparable with both these classes. In that case, their intersection would be the stochastic quantum regular languages (SQuRLs) [25].

If a generalized stochastic function f is the characteristic function of some language L, then L can be defined as $L = \{w \mid f(w) > 0\}$. Turakainen [35] showed that f can be replaced with a stochastic function, in which case L is a *0-stochastic language*. Bukharaev [5] has shown that any such language is regular, so we have a converse to Lemma 8:

Corollary. *If the characteristic function of a language L is a generalized QRL, then L is regular.*

3. Quantum context-free languages

3.1. Quantum push-down automata (QPDAs)

Next, we define quantum push-down automata and show that several modifications to the definition result in equivalent machines.

Definition. A *quantum push-down automaton* (QPDA) is a real-time quantum automaton where H is the tensor product of a finite-dimensional space Q, whose basis vectors are states of a finite-state control, and an infinite dimensional *stack space* Σ, whose basis vectors correspond to finite words over a stack alphabet T. We also require that s_{init}, which is now infinite dimensional, be a superposition of a finite number of different initial control and stack states.

Because of the last-in, first-out structure of a stack, only certain transitions can occur. If $q_1, q_2 \in Q$ are control states and $\sigma_1, \sigma_2 \in T^*$ are stack states, then the transition amplitude $\langle (q_1, \sigma_1) | U_a | (q_2, \sigma_2) \rangle$ can be nonzero only if $t\sigma_1 = \sigma_2$, $\sigma_1 = t\sigma_2$, or $\sigma_1 = \sigma_2$ for some $t \in T$. In other words, transitions can only push or pop single symbols on or off the stack or leave the stack unchanged. Furthermore, transition amplitudes can depend on the control state and the stack, but only on the top (leftmost) symbol of σ_1 and σ_2, or on whether or not the stack is empty.

Finally, for acceptance we demand that the QPDA end in both an accepting control state and with an empty stack. That is, $H_{\text{accept}} = Q_{\text{accept}} \otimes \{\varepsilon\}$ for some subspace $Q_{\text{accept}} \subset Q$.

This definition differs in several ways from that of classical PDAs [16]. First of all, the amplitude of a popping transition can depend both on the top stack symbol and the one below it, since the one below it is the top symbol of the stack we're making a transition to. We do this for the sake of unitarity and time-symmetry, since the amplitude of a pushing transition depends on both the top symbol and the symbol pushed. Similarly, popping transition amplitudes can depend on whether the stack will be empty afterwards.

In the generalized case where the transition matrices are not constrained to be unitary, we can easily get rid of this dependence:

Lemma 11. *A generalized QPDA can be simulated by a generalized QPDA whose transition amplitudes do not depend on the second-topmost stack symbol.*

Proof. Simply expand the stack alphabet to $T' = T \cup T^2$. Let each stack symbol also inform the QPDA of the symbol below it or that it is the bottom symbol. For instance, the stack *stu* becomes $(s,t)(t,u)u$. □

However, we believe Lemma 11 holds only in the generalized case. While the machine's dynamic is still unitary on the subset of the stack space that we will actually visit, we see no way to extend it to the entire stack space, including nonsense stacks like $(s,t)(u,w)$, in a unitary, time-symmetric way.

Again, for time-symmetry's sake, since we can only pop one symbol at a time, we only allow ourselves to push one symbol at a time. We next show that allowing us to push words of arbitrary length adds no additional power, just as for classical PDAs, at least in the generalized case:

Lemma 12. *A generalized QPDA that is allowed to push words of arbitrary length on the stack can be simulated by a generalized QPDA as defined above, for which every move pushes or pops one symbol or leaves the stack unchanged.*

Proof. In the classical case, we can do this simply by adding extra control states that push the word on one symbol at a time (Lemma 10.1 of [16]). However, this allows several steps per input symbol and thus violates our real-time restriction, so we need a slightly more subtle construction.

Suppose the old QPDA pushes words γ of length at most k. Then we expand the stack alphabet to composite symbols $T' = T^k \times \{1,\ldots,k\}$, which we will denote (β,m), and expand the set of control states to $Q' = Q \times \{1,\ldots,k\}$, which we will denote (q,m_0).

We represent the old QPDA's stack as shown in Fig. 2. If the stack of the new QPDA is $(\beta_1,m_1)(\beta_2,m_2)\cdots(\beta_s,m_s)$, then each β_i represents a chunk of the old QPDA's stack, starting with β_i's m_{i-1}th symbol. Alternately, each m_i is a pointer telling us to skip to the m_ith symbol of β_{i+1}. The pointer m_0 to β_1 is stored in the control state.

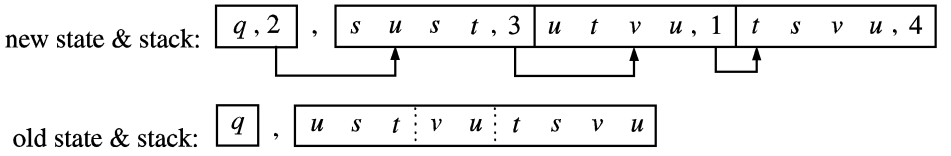

Fig. 2. Simulating a QPDA that can push words of length $\leqslant 4$ on the stack with one that only pushes or pops single symbols. The counter m_i in each stack symbol (β_i, m_i) acts as a pointer to the first relevant symbol in β_{i+1}. The pointer for β_1 is stored in the control state. The symbols to the left of each pointer are either dummies or symbols that have been popped off the original QPDA's stack.

Using Lemma 11, we assume that the old QPDA's transition amplitudes depend only on its top stack symbol. We operate the new QPDA as follows, replacing the transitions of the old QPDA with new ones of the same amplitude:

- To pop the top symbol, i.e. the m_0th symbol of β_1, change the control state by incrementing m_0. If $m_0 = k$, pop (β_1, m_1) off the stack and set $m_0 = m_1$ in the control state.
- To push a nonempty word γ of length $n \leqslant k$, choose a dummy symbol a and push $(a^{k-n}\gamma, m_0)$ on the stack, padding γ out to length k. Then set $m_0 = k - n + 1$ in the control state.

This converts a QPDA into one where each transition pushes or pops one symbol, or changes the topmost symbol of the stack by popping when $m_0 = k$ and then pushing a nonempty γ.

This simulation preserves our real-time restriction, and creates a QPDA which pushes or pops one symbol, or changes the top symbol, at each step. To complete the proof, we need to convert this QPDA into one that pushes, pops, or leaves the stack unchanged. This can be done by making the top symbol part of the control state, $Q'' = Q' \times T'$, so that we can change the top symbol by changing the state instead (as in Lemma 10.2 of [16]). □

Like Lemma 11, we believe Lemma 12 holds only in the generalized case. Unitarity appears to be lost even on the set of stacks actually visited. The stack state of the old QPDA is represented by many stack states of the new QPDA, depending on the intervening computation, and some of these receive less probability than others.

In the classical case, acceptance by control state and by empty stack are equivalent. We can prove this in one direction, in both the unitary and generalized case:

Lemma 13. *If a quantum language is accepted by a (generalized) QPDA by empty stack, then it is accepted by a (generalized) QPDA by control state.*

Proof. The standard construction (Theorem 5.1 of [16]) simply allows the PDA to empty its stack at the end of its computation, without reading any additional input. Since this violates our real-time restriction of one step per input symbol, we use a slightly different construction that also preserves unitarity.

First, double the number of control states to $Q' = Q \oplus \overline{Q}$, with a marked control state $\overline{q} \in \overline{Q}$ for each state $q \in Q$. Marked control states will denote an empty stack. Then replace transitions of the old QPDA, that pop to or push on an empty stack, with new transitions, with the same amplitudes, as follows:
– Replace pops of the form $(q_1, t) \to (q_2, \varepsilon)$ with $(q_1, t) \to (\overline{q}_2, \varepsilon)$.
– Replace pushes of the form $(q_1, \varepsilon) \to (q_2, t)$ with $(\overline{q}_1, \varepsilon) \to (q_2, t)$.
– Replace transitions on an empty stack $(q_1, \varepsilon) \to (q_2, \varepsilon)$ with $(\overline{q}_1, \varepsilon) \to (\overline{q}_2, \varepsilon)$.
Require all states (q, ε) (an unmarked control state and an empty stack) and $(\overline{q}, \sigma)$ (a marked control state and a nonempty stack) to make transitions only to themselves with amplitude 1. Finally, let s_{init} have nonzero components only along states $(\overline{q}, \varepsilon)$ that are marked and empty and (q, σ) that are unmarked and nonempty.

Then the new QPDA will be in a marked control state if and only if the stack is empty, so we accept with $H_{\text{accept}} = \overline{Q}_{\text{accept}} \otimes \Sigma$. The new transition matrices are direct sums of the old ones (with the basis vectors (q, ε) replaced by $(\overline{q}, \varepsilon)$) with an identity matrix (on the space generated by the (q, ε) and $(\overline{q}, \sigma)$). Thus, if the old QPDA is unitary, the new one is too. □

Unfortunately, we believe that a QPDA accepting by control state without regard to the stack cannot, in general, be simulated by one accepting by empty stack. The accepting subspace $H_{\text{accept}} = Q_{\text{accept}} \otimes \Sigma$ is infinite dimensional, allowing for an infinite number of different paths that add in a noninterfering way. We see no way to map this into a finite-dimensional subspace of the form $Q_{\text{accept}} \otimes \{\varepsilon\}$. Perhaps the reader can find a proof of this.

The last difference between QPDAs and classical PDAs is that, depending on its precise definition, a classical PDA either halts and accepts as soon as its stack becomes empty or rejects if it is asked to pop off an empty stack. In our case, we allow a QPDA to sense whether the stack is empty and act accordingly. We do this because of our strict real-time constraint, in which the only time the QPDA is allowed to talk back to us is when we perform a measurement at the end of the input process. Therefore, we have to tell the machine what to do if its stack is already empty and it receives more input.

3.2. Quantum context-free grammars

We now propose a definition of quantum grammars, in which each production has a set of complex amplitudes and multiple derivations of a word can interfere with each other constructively or destructively. We show that in the context-free case, these grammars generate exactly the languages recognized by quantum PDAs.

Definition. A *quantum grammar* G consists of two alphabets V and T, the *variables* and *terminals*, an initial variable $I \in V$, and a finite set P of *productions* $\alpha \to \beta$, where $\alpha \in V^*$ and $\beta \in (V \cup T)^*$. Each production in P has a set of complex amplitudes $c_k(\alpha \to \beta)$ for $1 \leqslant k \leqslant n$, where n is the *dimensionality* of the grammar.

We define the kth amplitude c_k of a derivation $\alpha \Rightarrow \beta$ as the product of the c_k's for each productions in the chain and $c_k(\alpha \Rightarrow \beta)$ as the sum of the c_k's of all derivations of β from α. Then the amplitudes of a word $w \in T^*$ are $c_k(w) = c_k(I \Rightarrow w)$ and the probability associated with w is the norm of its vector of amplitudes, summed over each dimension of the grammar, $f(w) = \sum_{k=1}^{n} |c_k(w)|^2$. We say G *generates* the quantum language f.

Finally, a quantum grammar is *context-free* if only productions where α is a single variable v have nonzero amplitudes. A *quantum context-free language* (QCFL) is one generated by some quantum context-free grammar.

The main result of this section is that a quantum language is context-free if and only if it is recognized by a generalized QPDA. We prove this with a series of lemmas that track the standard proof almost exactly. Our only innovation is attaching complex amplitudes to the productions and transitions, and showing that they match. A similar proof in the real-valued case is given for *probabilistic tree automata* in [12].

The multiple amplitudes c_k attached to each production seem rather awkward. As we will see below, they are needed so that paths ending in perpendicular states in Q_{accept} can add in a noninterfering way. If we had only one amplitude, then all paths would interfere with each other. In the grammars we actually construct, the c_k's will be equal, except for a few productions.

Definition. Two quantum grammars G_1 and G_2 are *equivalent* if they generate the same quantum language, $f_1(w) = f_2(w)$ for all w.

Definition. A quantum context-free grammar is in *Greibach normal form* if only productions of the form $v \to a\gamma$ where $a \in T$ and $\gamma \in V^*$ can have nonzero amplitudes, i.e. every product β consists of a terminal followed by a (possibly empty) string of variables.

We will also find the following requirement useful, although it is not the same as requiring that the grammar generates a finite word with probability 1, i.e. $\sum_w f(w) = 1$:

Definition. A quantum grammar is *terminating* if all infinite derivation trees have zero amplitudes.

Lemma 14. *Any terminating quantum context-free grammar is equivalent to one in Greibach normal form.*

Proof. This is essentially the same proof as in [12] for the real-valued case.

Clearly, G' is equivalent to G if for each derivation in G of a terminal word, there is exactly one derivation in G' with the same set of amplitudes. Then summing the amplitudes over all derivations will give the same answer for both grammars. All we need to do, then, is to attach amplitudes to the standard proof for classical

grammars (Lemmas 4.1–4.4 and Theorems 4.1–4.6 of [16]) and show that they are carried through correctly. As shorthand, we will refer to c_k and c'_k for all k as simply c and c', respectively.

First, Theorem 4.4 of [16] shows how to eliminate *unit productions* of one variable by another, $v_1 \to v_2$. If G has such productions, then for every production $v_i \to \beta$ in G where β is not a single variable, give G' the productions

$$c'(v_i \to \beta) = c(v_i \Rightarrow \beta) = \sum_j c(v_i \Rightarrow v_j)\, c(v_j \to \beta)$$

for all i, where

$$c(v_i \Rightarrow v_j) = \sum_{n=0}^{\infty} (M^n)_{ij} = (\mathbf{1} - M)^{-1}_{ij}$$

sums over all paths from v_i to v_j with n unit productions, and $M_{ij} = c(v_i \to v_j)$. This sum works if $\lim_{n\to\infty} M^n = 0$; but this is true if the grammar is terminating, since infinite chains of unit productions must have zero amplitude. Then setting $c'(v_i \to v_j) = 0$ leaves G' with no unit productions.

Second, Theorem 4.5 of [16] converts a grammar to *Chomsky normal form*, in which β consists of either a single terminal or two variables. For any production $v \to \beta$ in G where β consists of m variables $b_1 b_2 \cdots b_m$, introduce additional variables $d_1, d_2, \ldots d_{m-2}$ and allow the productions $v \to b_1 d_1$, $d_1 \to b_2 d_2$, …, $d_{m-2} \to b_{m-1} b_m$ in G'. Then give G' the productions

$$c'(v \Rightarrow \beta) = c'(v \to b_1 d_1) \cdot \prod_{i=1}^{m-3} c'(d_i \to b_{i+1} d_{i+1}) \cdot c'(d_{m-2} \to b_{m-1} b_m)$$

which we can make equal to $c(v \to \beta)$ by choosing the c' on the right-hand site appropriately, e.g. with $c'(v \to b_1 d_1) = c(v \to \beta)$ and the others set to 1.

Finally, Lemma 4.4 of [16] eliminates productions of the form $v \to v\alpha$. If G has such productions and v's other productions in G are $v \to \beta$, add a variable b and give G' the productions

$$c'(b \to \alpha) = c'(b \to \alpha b) = c(v \to v\alpha),$$
$$c'(v \to \beta) = c'(v \to \beta b) = c(v \to \beta)$$

for all α and β. Then

$$\begin{aligned} c'(v \Rightarrow \beta\alpha_1\alpha_2 \cdots \alpha_m) &= c'(v \to \beta b) \cdot \prod_{i=1}^{m-1} c'(b \to \alpha_i b) \cdot c'(b \to \alpha_m) \\ &= c(v \to \beta) \cdot \prod_{i=1}^{m} c(v \to v\alpha_i) \\ &= c(v \Rightarrow \beta\alpha_1\alpha_2 \cdots \alpha_m), \end{aligned}$$

where the derivation tree for G' now produces the α_i from left to right rather than from right to left.

The reader can easily check that the rest of the proof of Theorem 4.6 of [16] can be rewritten this way, so that G and G' have derivations with all the same complex amplitudes. □

Greibach normal form is useful because the derivation trees it generates create a terminal symbol on the left with every production. Each such tree corresponds to a computation of a real-time PDA that accepts with an empty stack. Adding complex amplitudes gives us the quantum version of Theorem 5.3 of [16]:

Theorem 15. *Any QCFL is recognized by a generalized QPDA.*

Proof. Convert the QCFL's grammar into Greibach normal form. Then construct a QPDA with the terminals T as its input symbols, with the variables V as its stack alphabet, and with one control state q_k for each dimension of the grammar, $1 \leqslant k \leqslant n$.

Let the QPDA's transitions be as follows. For each production $v \to a\gamma$ where $a \in T$ and $\gamma \in V^*$, if the control state is q_k and the top stack symbol is v, let U_a pop v and push γ on the stack with amplitude $c_k(v \to a\gamma)$. Always leave the control state unchanged.

Then as we read the input symbols a, the QPDA guesses a derivation tree and ends with an empty stack. The amplitude of a computation path with control state q_k is equal to the kth amplitude of the corresponding derivation. Summing over all paths is equivalent to summing over all derivations. If the QPDA's initial control state vector is $q_{\text{init}} = (1, 1, \ldots, 1)$, the initial stack is I, and $Q_{\text{accept}} = Q$, then projecting onto $H_{\text{accept}} = Q \otimes \{\varepsilon\}$ sums over all k and gives the norm $f(w) = \sum_k |c_k(w)|^2$.

This gives us a QPDA that pushes whole words on the stack. Using Lemma 12, we can convert it into one that pushes or pops one symbol or leaves the stack unchanged, and we're done. □

Conversely, by assigning the correct amplitudes to the productions in Theorem 5.4 of [16], we can make each derivation match a computation path of a QPDA:

Theorem 16. *Any quantum language recognized by a generalized QPDA is a QCFL.*

Proof. By Lemma 11, we will assume that the QPDA's transition amplitudes do not depend on the second-topmost stack symbol.

Our variables will be of the form $[q_1, t, q_2]$, where $q_1, q_2 \in Q$ and $t \in \Sigma \cup \{\varepsilon\}$. The leftmost variable will tell us that the QPDA is in control state q_1 with top symbol t (or an empty stack if $t = \varepsilon$) and will be in state q_2 by the time t is popped. As in the previous theorem, the terminals will be the input symbols of the QPDA, and the k'th amplitude c_k of the derivation will be the amplitude of all paths that end with a final state q_k. Thus the dimensionality of the grammar is equal to that of Q_{accept}.

To start us off, we guess the QPDA's final state q_k, initial state q_1, and initial stack β, and what states $q_2, \ldots, q_{|\beta|}$ we will go through as we pop the symbols of β. For

each allowed control state $q_k \in Q_{\text{accept}}$, for each state-stack pair (q_1, β) with nonzero amplitude in s_{init}, and for all possible chains of control states $q_2, \ldots, q_{|\beta|} \in Q$, allow the production

$$I \to [q_1, \beta_1, q_2]\,[q_2, \beta_2, q_3] \cdots [q_{|\beta|}, \beta_{|\beta|}, q_k]$$

with amplitudes $c_k = \langle s_{\text{init}} | (q_1, \beta) \rangle$ and $c_j = 0$ for all $j \neq k$. (These will be our only productions for which c_k depends on k.)

Then reading an input symbol $a \in A$, pushing a symbol s on the stack, and entering state q_3 is represented by a production of the form

$$[q_1, t, q_2] \to a\,[q_3, s, q_4]\,[q_4, t, q_2] \tag{2}$$

whose amplitudes c_k are all equal to the amplitude $\langle (q_1, \sigma) | U_a | (q_3, s\sigma) \rangle$ of this QPDA transition. This production is allowed for any q_4, which is the state we guess that we will pass through after popping s at some later time.

Similarly, reading an input symbol a, popping t off the stack, and entering state q_2 is represented by

$$[q_1, t, q_2] \to a \tag{3}$$

whose amplitudes c_k are all equal to the amplitude $\langle (q_1, t\sigma) | U_a | (q_2, \sigma) \rangle$ of this transition. Changing the state to q_3 while leaving the stack unchanged is represented by

$$[q_1, t, q_2] \to a\,[q_3, t, q_2] \tag{4}$$

with amplitudes $c_k = \langle (q_1, \sigma) | U_a | (q_3, \sigma) \rangle$.

Then, if we apply our productions always to the leftmost variable, we see that each derivation tree corresponds to a computation path of the QPDA with the same amplitude as the derivation. Summing over derivations sums over computation paths. $c_k(w) = \langle s_{\text{init}} | U_a | (q_k, \varepsilon) \rangle$ is the amplitude of all paths that end with the QPDA in control state q_k with an empty stack. Then $f(w) = \sum_{k=1}^{n} |c_k(w)|^2$ sums over all $q_k \in Q_{\text{accept}}$ and the theorem is proved. □

This representation of the control state, in which every control state occurs in two variables, is necessary to enforce a consistent series of transitions, since symbols in a context-free derivation have no way of communicating with each other once they are created.

An alternate approach would be to give our productions *matrix-valued* amplitudes, so that their transitions can keep track of the state. Our current definition, in which the c_k are simply multiplied componentwise, is equivalent to using diagonal matrices. Since matrices do not commute in general, we would have to choose an order in which to multiply the production amplitudes to define a derivation's amplitude. A leftmost depth-first search of a derivation in Greibach normal form would still correspond to a computation path of a QPDA. However, our proof of Greibach normal form breaks down because of the way Lemma 4.4 of [16] changes the shape of the tree. If such

grammars can be put in Greibach normal form, then Theorem 15 works and they are equivalent to QPDAs. If they cannot, they may be more powerful.

The productions in the above proof look nonunitary because they produce either too much probability, since (2) is allowed for any choice of q_4, or too little, since (3) and (4) may not correspond to transitions that are allowed at all. Let us define

Definition. A QCFL is *unitary* if it is recognized by a unitary QPDA.

It is not clear what constraints a quantum grammar needs to meet to be unitary. Nor is it clear whether these constraints can be put in a simple form that is preserved by the kinds of transformations we use in Lemma 14. Perhaps a grammar's productions affect unitarity in a similar way to the rule table of a quantum cellular automaton. An algorithm to tell whether a quantum CA is unitary is given in [11].

Finally, we note that theorems 15 and 16 have the following corollaries:

Corollary. *Any quantum context-free grammar is equivalent to one in which the production amplitudes c_k do not depend on k except for productions from the initial variable. Any generalized QPDA can be simulated by one whose transitions never change its control state, for which $Q_{\text{accept}} = Q$, and whose only initial stack consists of a single symbol.*

It is not clear whether the latter is true in the unitary case.

3.3. Closure properties of QCFLs

Classical context-free languages are closed under intersection with a regular language. The quantum version of this follows easily:

Lemma 17. *If f is a (unitary) QCFL and g is a QRL, then fg is a (unitary) QCFL.*

Proof. We simply form the tensor product of the two automata. If f and g have finite-dimensional state spaces Q and R, construct a new QPDA with control states $Q \otimes R$, transition matrices $U'_a = U_a^f \otimes U_a^g$ (recall that $\otimes$ preserves unitarity), and accepting subspace $H'_{\text{accept}} = Q_{\text{accept}} \otimes R_{\text{accept}} \otimes \{\varepsilon\}$. □

Classical CFLs are also closed under union, which as before becomes addition:

Lemma 18. *If f and g are QCFLs, then $f + g$ is a QCFL.*

Proof. We define a direct sum of two grammars as follows. Suppose the grammars generating f and g have m and n dimensions, variables V and W, and initial variables I and J. We will denote their amplitudes by c_k^f and c_k^g. Then create a new grammar with $m+n$ dimensions, variables $V \cup W \cup \{K\}$, and initial variable K, with the productions $K \to I$ and $K \to J$ allowed with amplitudes $c_k = 1$. Other productions are allowed

with $c_k = c_k^f$ for $1 \leqslant k \leqslant m$ and $c_k = c_{k-m}^g$ for $m+1 \leqslant k \leqslant m+n$. The reader can easily check that this grammar generates $f + g$. □

We would like to say that a weighted sum $af + bg$, where $a + b = 1$, of unitary QCFLs is unitary. This is true if the QPDAs accepting f and g have stack alphabets of the same size. Just take the direct sum of their control state spaces and let both sets of states interpret the stack as if it were their own. However, if one stack alphabet is bigger than the other, we have to figure out how to handle the dynamics in a unitary way when one of f's states tries to read one of g's stack symbols. We leave this as a question for the reader.

3.4. The generating functions of QCFLs

If we define a generating function of a context-free language L that counts multiple derivations, $G_L = \sum_{w \in L} n(w)\, w$, where $n(w)$ is the number of derivations of w in L's grammar, then G_L is algebraic. That is, it is a solution to a finite set of polynomial equations in noncommuting variables [18]. If we don't count multiple derivations and define $G_L = \sum_{w \in L} w$ instead, then G_L is algebraic for unambiguous context-free languages since each word has a unique derivation [16].

For instance, the Dyck language is generated by the unambiguous grammar $P = \{I \to aIbI,\ I \to \varepsilon\}$, where we have replaced left and right brackets with a and b, respectively. Then its generating function obeys the quadratic equation in noncommuting variables

$$G = aGbG + 1.$$

If we set $a = b = z$, this becomes

$$g(z) = z^2 g^2 + 1$$

whose solution is

$$g(z) = \frac{1 - \sqrt{1 - 4z^2}}{2z^2} = 1 + z^2 + 2z^4 + 5z^6 + 14z^8 + \cdots$$

whose z^{2k} coefficient is the *Catalan number* $\binom{2k}{k}/(k+1)$.

The closest we can come to this in the quantum case is the following.

Definition The *Hadamard square* of a formal power series g is the Hadamard product $g^* \odot g$.

Theorem 19. *If f is a QCFL, then G_f is a restriction of the Hadamard square of an algebraic power series.*

Proof. As in Theorem 9, we start with generating functions weighted with complex amplitudes rather than probabilities. For each dimension k of the grammar write c for

c_k and define

$$g_v = \sum_{w \in T^*} c(v \Rightarrow w)\, w.$$

This is the generating function of the terminal words $w \in T^*$ that can be derived from a variable $v \in V$, weighted by the kth amplitudes of each derivation. For a terminal $a \in T$, we define $g_a = a$ since a can only produce itself. We also use the shorthand

$$g_\beta = g_{\beta_1} g_{\beta_2} \cdots g_{\beta_{|\beta|}},$$

since the words that can be derived from a word β are simply concatenations of those that can be derived from each of β's symbols.

Then the g_v obey the following equations, with one term for each production:

$$g_v = \sum_{\beta \in (V \cup T)^*} c(v \to \beta)\, g_\beta$$

each of which is a polynomial of order $\max_{\beta | c(v \to \beta) \neq 0} |\beta|$. This system of equations has an algebraic solution g_I.

If we call the g_I based on the k'th amplitude g_k, then G_f is the sum of their Hadamard squares

$$G_f = \sum_w f(w)\, w = \sum_w \sum_{k=1}^{n} |c_k(w)|^2\, w = \sum_{k=1}^{n} g_k^* \odot g_k.$$

We can write this as a single Hadamard square in the following way. For each dimension k of the grammar, introduce a new symbol x_k. Then if we define $g = \sum_{k=1}^{n} x_k g_k$, we have

$$g^* \odot g = \sum_{k=1}^{n} x_k\, (g_k^* \odot g_k)$$

and $G_f = g^* \odot g$ in the restriction $x_k = 1$ for all k. □

Unfortunately, unlike the class of rational series, the class of algebraic series is not closed under Hadamard product. This corresponds to the fact that the context-free languages are not closed under intersection. In fact, the set of accepting computations of a Turing machine is the intersection of two CFLs, so it is undecidable whether two algebraic series have a nonzero Hadamard product [16].

This also means that the Hadamard square of an algebraic series can be transcendental. Let A and B be two algebraic series such that $A \odot B$ is transcendental. Then if $C = (A + B)/2$ and $D = (A - B)/2$, we have $A \odot B = (C \odot C) - (D \odot D)$ and at least one of $C \odot C$ and $D \odot D$ must be transcendental. As a concrete example, $g(z) = \sum_{z=0}^{\infty} \binom{2n}{n} z^n$ is algebraic, but $g \odot g$ can be shown to be transcendental using the asymptotic techniques in [13].

Ideally, this result could be used to show that certain inherently ambiguous context-free languages, whose generating functions are not the Hadamard square of an algebraic function, are not QCFLs. Unfortunately, it is not obvious how to prove this, even in the case where all the $f(w)$ are 0 or 1.

3.5. Regular grammars

Although it is painfully obvious at this point, we include the following for completeness.

Definition. A quantum grammar is *regular* if only productions of the form $v_1 \to wv_2$ and $v_1 \to w$ have nonzero amplitudes, where $v_1, v_2 \in V$ are variables and $w \in T^*$ is a (possibly empty) word of terminals.

Theorem 20. *A quantum language is a generalized QRL if and only if it is generated by a regular quantum grammar.*

Proof. First, we show that the language f generated by a regular quantum grammar is a generalized QRL. Using the techniques of Lemma 14, we can convert any regular grammar into one where $|w| = 1$, i.e. all productions are of the form $v_1 \to av_2$ or $v_1 \to a$, where $v_1, v_2 \in V$ and $a \in T$.

If there are m variables, then for each dimension k of the grammar we can define a set of $(m+1)$-dimensional transition matrices $U_a^{(k)}$:

$$(U_a^{(k)})_{ij} = \begin{cases} c_k(v_i \to av_j) & 1 \leqslant i, j \leqslant m, \\ c_k(v_i \to a) & j = m+1, \\ 0 & i = m+1. \end{cases}$$

Then $|c_k(w)| = |s_{\text{init}} U_w^{(k)} P_{\text{accept}}|$, where s_{init} is the unit vector $(s_{\text{init}})_i = 1$ if $v_i = I$ and 0 otherwise; and $uP_{\text{accept}} = u_{m+1}$, i.e. P_{accept} projects onto a vector's $(m+1)$th component. Then each $f_k = |c_k(w)|^2$ is a QRL and by Lemma 1 so is their sum $f(w) = \sum_{k=1}^{n} f_k(w) = \sum_{k=1}^{n} |c_k(w)|^2$.

Conversely, let f be a generalized QRL. Its state space is spanned by a set of unit vectors that we identify with the variables V. The accepting subspace H_{accept} is spanned by a set of unit vectors h_k as in Theorem 9, each of which corresponds to one dimension of the grammar. Then define the production amplitudes as follows:

$$\begin{aligned} c_k(I \to v) &= \langle s_{\text{init}} | v \rangle, \\ c_k(v_i \to av_j) &= (U_a)_{ij}, \\ c_k(v_j \to \varepsilon) &= \langle v_j | h_k \rangle. \end{aligned}$$

Then $\sum_{k=1}^{n} |c_k(w)|^2 = \sum_{k=1}^{n} |\langle s_{\text{init}} | U_w | h_k \rangle|^2 = |\langle s_{\text{init}} | U_w | P_{\text{accept}} \rangle|^2$ and the theorem is proved. □

Since only the last of the amplitudes in Theorem 20 depend on k, we can add the following corollary:

Corollary. *Any regular grammar is equivalent to one in which the c_k do not depend on k except for productions of the form $v \to \varepsilon$.*

Just as the regular languages are a proper subclass of the context-free languages, we can show that the QRLs are a proper subclass of the QCFLs, in both the unitary and nonunitary cases:

Theorem 21. *The QRLs are a proper subclass of the unitary QCFLs, and the generalized QRLs are a proper subclass of the QCFLs.*

Proof. Containment is given in both cases by using the control state of a (unitary) QPDA to simulate a (unitary) QFA while leaving its stack alone. It is proper because the language $L_=$ of words in $\{a,b\}$ with an equal number of a's and b's is a unitary QCFL (or rather, its characteristic function is) but not a generalized QRL, as we will now show.

Consider a QPDA with two control states A and B and one stack symbol x. The stack will indicate how many excess a's or b's we have, with the control state indicating which dominates. Then starting with an empty stack $s_{\text{init}} = (A,\varepsilon)$, we can recognize $L_=$ with the transition matrices

$$U_a = \begin{array}{c|ccccccc} & (A,\varepsilon) & (A,x) & (B,x) & (A,xx) & (B,xx) & (A,xxx) & (B,xxx) & \cdots \\ \hline (A,\varepsilon) & & 1 & & & & & & \\ (A,x) & & & & 1 & & & & \\ (B,x) & 1 & & & & & & & \\ (A,xx) & & & & & & 1 & & \\ (B,xx) & & & 1 & & & & & \\ (A,xxx) & & & & & & & & \ddots \\ (B,xxx) & & & & & 1 & & & \\ \vdots & & & & & & \ddots & & \end{array}$$

(with all other entries zero and (B,ε) left unchanged and unused) and $U_b = U_a^\dagger = U_a^{-1}$. Since both U_a and U_b are unitary, this is a QPDA and $L_=$ is a unitary QCFL.

On the other hand, $L_=$'s generating function

$$g(z) = \sum_{n=0}^{\infty} \binom{2n}{n} z^{2n} = \frac{1}{\sqrt{1-4z^2}}$$

is algebraic but not rational, so $L_=$ is not a generalized QRL by Theorem 9. □

Since regular grammars are also context-free, Theorem 20 is another proof that the generalized QRLs are a subclass of the QCFLs.

3.6. QCFLs and CFLs

Finally, we will compare our quantum classes to their classical counterparts. Lemma 7 states that any regular language is a generalized QRL. Similarly, we have (again conflating a language with its characteristic function):

Lemma 22. *Any unambiguous context-free language is a QCFL. More specifically, for any unambiguous CFL L there is a quantum grammar of dimensionality* 1 *such that* $c(w) = \chi_L(w)$.

Proof. Simply give allowed and disallowed productions amplitudes 1 and 0, respectively. Since L is unambiguous, each allowed word has exactly one derivation, so $c(w) = \chi_L(w)$. Since 0 and 1 are their own squares, we also have $f(w) = |c(w)|^2 = \chi_L(w)$. □

Using the quantum effect of destructive interference, we can get the following nonclassical result, showing that quantum context-free grammars and QPDAs are strictly more powerful than classical ones:

Theorem 23. *If L_1 and L_2 are unambiguous context-free languages, their symmetric difference $L_1 \triangle L_2 = (L_1 \cup L_2) - (L_1 \cap L_2)$ is a QCFL.*

Proof. If L_1 and L_2 are generated by grammars with initial variables I_1 and I_2, then create a new initial variable I and allow the productions $I \to I_1$ and $I \to I_2$ with amplitudes 1 and -1, respectively. Then $f = |c_1(w) + c_2(w)|^2 = 1$ if w is in L_1 or L_2, but not both. □

Corollary. *If L_1 is an unambiguous context-free language, its complement $\overline{L_1}$ is a QCFL.*

Proof. Let $L_2 = A^*$. □

Theorem 24. *There are QCFLs that are not context-free.*

Proof. Let $L_1 = \{a^i b^i c^j\}$ and $L_2 = \{a^i b^j b^j\}$, both of which are unambiguous context-free. Then

$$L_1 \triangle L_2 = \{a^i b^j c^k \mid i = j \text{ or } j = k, \text{ but not both}\}$$

is a QCFL, but it can be shown to be noncontext-free using the pumping lemma for context-free languages [16]. □

This proof that **QCFL** $\neq$ **CFL** relies simply on the non-closure of the CFLs under $\triangle$. The same idea could be used for some other low-lying complexity classes.

We can use interference in another amusing way:

Theorem 25. *If L_1, L_2, and L_3 are unambiguous context-free languages, then $(L_1 \cup L_2 \cup L_3) - (L_1 \cap L_2 \cap L_3)$ is a QCFL.*

Proof. Create a new initial variable I and allow the productions $I \to I_1$, $I \to I_2$, and $I \to I_3$ with amplitudes 1, $e^{2\pi i/3}$, and $e^{4\pi i/3}$, respectively. Since these are 120° apart,

$f = |c_1(w) + c_2(w) + c_3(w)|^2$ if w is in one or two, but not all three, of the three languages. □

Corollary. *If L_1 and L_2 are unambiguous context-free languages, then $L_1 \cup L_2$ and $\overline{L_1 \cap L_2}$ are QCFLs.*

Proof. Let $L_3 = A^*$ and $L_3 = \emptyset$, respectively. □

Unfortunately, there are no sets of four or more vectors with norm 1 such that the sum of any subset of them has norm 1, so this is as far as this argument goes.[2]

This gives us the following undecidability result, analogous to the undecidability of whether $L = A^*$ for a classical CFL.

Theorem 26. *If f is a QCFL, it is undecidable whether $f(w) = 1$ for all words w. Therefore, it is also undecidable whether two QCFLs are equivalent.*

Proof. Given two CFLs L_1 and L_2, it is undecidable whether their intersection is empty [16]. By the above corollary, the characteristic function of $\overline{L_1 \cap L_2}$ is a QCFL. □

The question of whether a QCFL has nonempty support, i.e. whether $f(w) \neq 0$ for any w, is decidable for classical CFLs [16]. It is not so clear for QCFLs, since destructive interference can cancel out all derivations of a word. We leave this as an open question.

It would be nice to have examples of a QCFL which uses interference in a more fundamental way, all the way down the derivation tree, rather than just joining two or three trees of different phases together at the top. Perhaps the reader can come up with such examples. It would also be nice to use theorem 19 to show that some inherently ambiguous CFLs, with transcendental generating functions, are not QCFLs, in which case the CFLs and QCFLs would be incomparable.

4. Conclusion and directions for further work

We have defined quantum versions of finite-state automata, push-down automata, and context-free grammars. While many classical results carry over into the quantum case, we have shown that classical and quantum CFLs are provably different.

We leave the reader with a set of open questions, some of which have already been mentioned above:

1. What happens when we remove the real-time restriction, allowing the machine to choose when to read an input symbol? This adds no power to classical DFAs and PDAs [16]. Does it in the quantum case?

[2] We are indebted to Jan-Christoph Puchta, David Joyner, Benjamin Lotto, and Dan Asimov for providing proofs of this fact.

2. What about two-way automata, 2QFAs and 2QPDAs, that can choose to move left or right on the input? Kondacs and Watrous [17] have shown that a 2QFA can recognize the nonregular language $\{a^n b^n\}$, but their model allows the user to perform a measurement of the machine at each step. Are 2QFAs more powerful than their classical counterparts when restricted to a single measurement after a specified time?
3. Is there a natural quantum analog of rational transductions [3], under which QRLs and QCFLs are closed without losing unitarity?
4. Are QRLs incomparable with stochastic and pseudo-stochastic functions?
5. Is each QRL recognized by a unique QFA (up to isomorphism) with the minimal number of dimensions? It might be possible to determine the eigenvalues of U_w for all w by Fourier analysis of $f(uw^k v)$. We could then reconstruct the U_a, since any set of matrices is determined by their eigenvalues and those of their products [15].
6. Can grammars with noncommuting matrix-valued amplitudes be defined in a consistent way and put in Greibach normal form?
7. Is there a simple way of determining whether a quantum context-free grammar generates a unitary QCFL?
8. Can a QPDA be simulated by one that never changes its control state, and for which $Q_{\text{accept}} = Q$, without losing unitarity?
9. Is a weighted sum of unitary QCFLs a unitary QCFL, even when their QPDAs have stack alphabets of different sizes?
10. Is there a quantum analog to the Dyck languages D_k and to Chomsky's theorem that every CFL is a homomorphic image of the intersection of D_k with a regular language?
11. Is nonemptiness decidable for QCFLs?
12. Are QCFLs context-sensitive?
13. Are there CFLs that are not QCFLs?
14. Can we define quantum versions of other real-time recognizer classes, such as queue automata [6], counter automata, and real-time Turing machines [2, 10]?
15. Are valid computations of real-time QTMs the product of two QCFLs, analogous to the classical case [16]?
16. We can easily define quantum context-sensitive grammars. Do they correspond to a quantum version of linear-bounded Turing machines?

We hope that quantum grammars and automata will be fruitful areas of research and that they will be useful to people studying quantum computation.

Acknowledgements

We are grateful to Bruce Litow, Philippe Flajolet, and Christophe Reutenauer for the proof that the Hadamard square of an algebraic series can be transcendental; Bruce Reznick, Jan-Christoph Puchta, Alf van der Poorten, Timothy Chow and Robert Israel

for advice on rational generating functions; John Baez for pointing out Ref. [15]; Ioan Macarie and Eduardo Sontag for a reading of the manuscript; Christian "Ducky" Reidys for help on functors; Umesh Vazirani for helpful discussions; and the referees for several corrections and suggestions. C.M. also wishes to thank Molly Rose and Café du Nord for inspiration. This work was supported at UC Berkeley by ONR Grant N00014-96-1-0524 and at the Santa Fe Institute by ONR Grant N00014-95-1-0975 and NSF grant ASC-9503162.

References

[1] L.M. Adleman, Molecular computation of solutions to combinatorial problems, Science 266 (1994) 1021–1023.
[2] P. Benioff, Quantum mechanical Hamiltonian models of Turing machines that dissipate no energy, Phys. Rev. Lett. 48 (1982) 1581–1585 and J. Statist. Phys. 29 (1982) 515–546.
[3] J. Berstel, Transductions and Context-Free Languages, Teubner Studienbücher, Berlin, 1978.
[4] L. Blum, M. Shub, S. Smale, On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines, Bull. Amer. Math. Soc. 21 (1989) 1–46.
[5] R. Bukharaev, On the representability of events in probabilistic automata, Probab. Meth. Cybernet. V Kazan (1967) 7–20 (Russian).
[6] A. Cherubini, C. Citrini, S.C. Reghizzi, D. Mandrioli, QRT FIFO automata, breadth-first grammars and their relations, Theoret. Comput. Sci. 85 (1991) 171–203.
[7] J.I. Cirac, P. Zoller, Quantum computations with cold trapped ions, Phys. Rev. Lett. 74 (1995) 4091–4094.
[8] J.P. Crutchfield, The calculi of emergence: computation, dynamics, and induction, Physica D 75 (1994) 11–54.
[9] J.P. Crutchfield, M. Mitchell, The evolution of emergent computation, Proc. Natl. Acad. Sci. 92 (1995) 10742–10746.
[10] D. Deutsch, Quantum theory, the Church–Turing principle and the universal quantum computer, Proc. R. Soc. London Ser. A 400 (1985) 97–117.
[11] C. Dürr, M. Santha, A decision procedure for unitary linear quantum cellular automata, Proc. 37th Symp. on Foundations of Computer Science, 1996, pp. 38–45.
[12] C.A. Ellis, Probabilistic languages and automata, Ph.D. Thesis, University of Illinois, Urbana, 1969.
[13] P. Flajolet, Analytic models and ambiguity of context-free languages, Theoret. Comput. Sci. 49 (1987) 283–309.
[14] N.A. Gershenfeld, I.L. Chuang, Bulk spin-resonance quantum computation, Science 275 (1997) 350–356.
[15] R. Giles, Reconstruction of gauge potentials from Wilson loops, Phys. Rev. D 24(8) (1981) 2160–2168.
[16] J.E. Hopcroft, J.D. Ullman, Introduction to Automata Theory, Languages, and Computation, Addison-Wesley, New York, 1979.
[17] A. Kondacs, J. Watrous, On the power of quantum finite state automata, Proc. 38th Symp. on Foundations of Computer Science, 1997, To appear.
[18] W. Kuich, A. Salomaa, Semirings, Automata, Languages, EATCS Monographs on Theoretical Computer Science, vol. 5. Springer, Berlin, 1986.
[19] S. Lloyd, A potentially realizable quantum computer, Science 261 (1993) 1569-1571.
[20] I. Macarie, Closure properties of stochastic languages, University of Rochester Computer Science Technical Report 441, 1993.
[21] C. Moore, Unpredictability and undecidability in dynamical systems, Phys. Rev. Lett. 64 (1990) 2354–2357 and Nonlinearity 4 (1991) 199–230.
[22] C. Moore, Dynamical recognizers: real-time language recognition by analog computers, Theoret. Comput. Sci. 201 (1998) 99–136.
[23] A. Paz, Introduction to Probabilistic Automata, Academic Press, New York, 1971.
[24] M.O. Rabin, Probabilistic automata, Inform. Control 6 (1963) 230–245.

[25] El Rodento Diablo, personal communication.
[26] A.L. Rosenberg, Real-time definable languages, J. ACM 14 (1967) 645–662.
[27] Y. Sakakibara, M. Brown, R. Hughey, I.S. Mian, K. Sjolander, R.C. Underwood, D. Haussler, Recent methods for RNA modeling using stochastic context-free grammars, in: M. Crochemore, D. Gusfield, (Eds.), Combinatorial Pattern Matching, 5th Annual Symp., Springer, Berlin, 1994, pp. 289-306.
[28] D.B. Searls, The linguistics of DNA, Am. Scientist 80 (1992) 579–591.
[29] P.W. Shor, Algorithms for quantum computation: discrete logarithms and factoring, Proc. 35th Symp. on Foundations of Computer Science, 1994, pp. 124–134.
[30] P.W. Shor, Fault-tolerant quantum computation, Proc. 37th Symp. on Foundations of Computer Science, 1996, pp. 56–65.
[31] H. Siegelmann, E.D. Sontag, Analog computation via neural networks, Theoret. Comput. Sci. 131 (1994) 331–360.
[32] H. Siegelmann, E.D. Sontag, L. Giles, The complexity of language recognition by neural networks, in: J. van Leeuwen, (Ed.), Algorithms, Software, Architecture, North-Holland, Amsterdam, 1992, pp. 329–335.
[33] A.M. Steane, Active stabilization, quantum computation, and quantum state synthesis, Phys. Rev. Lett. 78 (1997) 2252–2255.
[34] J.S. Townsend, A Modern Approach to Quantum Mechanics, McGraw-Hill, New york 1992.
[35] P. Turakainen, On stochastic languages, Inform. Control 12 (1968) 304–313.
[36] D.R. Upper, Theory and algorithms for hidden Markov models, and generalized hidden Markov models, Ph.D. Thesis, Mathematics Department, University of California, Berkeley, 1997.