Estimating Jones Polynomials is a Complete Problem for One Clean Qubit

Peter W. Shor* and Stephen P. Jordan†

Abstract

It is known that evaluating a certain approximation to the Jones polynomial for the plat closure of a braid is a BQP-complete problem. That is, this problem exactly captures the power of the quantum circuit model[12, 3, 1]. The one clean qubit model is a model of quantum computation in which all but one qubit starts in the maximally mixed state. One clean qubit computers are believed to be strictly weaker than standard quantum computers, but still capable of solving some classically intractable problems [20]. Here we show that evaluating a certain approximation to the Jones polynomial at a fifth root of unity for the trace closure of a braid is a complete problem for the one clean qubit complexity class. That is, a one clean qubit computer can approximate these Jones polynomials in time polynomial in both the number of strands and number of crossings, and the problem of simulating a one clean qubit computer is reducible to approximating the Jones polynomial of the trace closure of a braid.

1 One Clean Qubit

The one clean qubit model of quantum computation originated as an idealized model of quantum computation on highly mixed initial states, such as appear in NMR implementations[20, 4]. In this model, one is given an initial quantum state consisting of a single qubit in the pure state $|0\rangle$, and n qubits in the maximally mixed state. This is described by the density matrix

$$\rho = |0\rangle\langle 0| \otimes \frac{I}{2^n}.$$

One can apply any polynomial-size quantum circuit to ρ, and then measure the first qubit in the computational basis. Thus, if the quantum circuit implements the unitary transformation U, the probability of measuring $|0\rangle$ will be

$$p_0 = \mathrm{Tr}[(|0\rangle\langle 0| \otimes I)U\rho U^\dagger] = 2^{-n}\mathrm{Tr}[(|0\rangle\langle 0| \otimes I)U(|0\rangle\langle 0| \otimes I)U^\dagger]. \quad (1)$$

Computational complexity classes are typically described using decision problems, that is, problems which admit yes/no answers. This is mathematically convenient, and the implications for the complexity of non-decision problems are usually straightforward to obtain (*c.f.* [22]). The one clean qubit complexity class consists of the decision problems which can be solved in polynomial time by a one clean qubit machine with correctness probability of at least 2/3. The experiment described in equation 1 can be repeated polynomially many times. Thus, if $p_1 \geq 1/2 + \epsilon$ for instances to which the answer is yes, and $p_1 \leq 1/2 - \epsilon$ otherwise, then by repeating the experiment poly$(1/\epsilon)$ times and taking the majority vote one can achieve 2/3 probability of correctness. Thus, as long as ϵ is at least an inverse polynomial in the problem size, the problem is contained in the one clean qubit complexity class. Following [20], we will refer to this complexity class as DQC1.

A number of equivalent definitions of the one clean qubit complexity class can be made. For example, changing the pure part of the initial state and the basis in which the final measurement is performed does

*Mathematics Department, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139. *shor@math.mit.edu*
†Center for Theoretical Physics, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139. *sjordan@mit.edu*

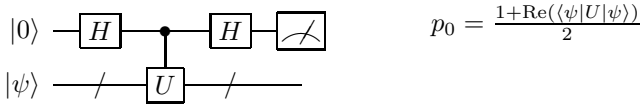


Figure 1: This circuit implements the Hadamard test. The probability p_0 of measuring $|0\rangle$ is as shown above. Thus, one can obtain the real part of $\langle\psi| U |\psi\rangle$ to precision ϵ by making $O(1/\epsilon^2)$ measurements and counting what fraction of the measurement outcomes are $|0\rangle$. Similarly, if the control bit is initialized to $|1\rangle$ instead of $|0\rangle$, one can estimate the imaginary part of $\langle\psi| U |\psi\rangle$.

not change the resulting complexity class. Less trivially, allowing logarithmically many clean qubits results in the same class, as discussed below. It is essential that on a given copy of ρ, measurements are performed only at the end of the computation. Otherwise, one could obtain a pure state by measuring ρ thus making all the qubits "clean" and re-obtaining BQP.

The study of quantum computation on high entropy initial states is partly motivated by the fact that low entropy initial states are experimentally difficult to achieve in many systems. It may appear artificial that in the one clean qubit model, there are some pure qubits and some maximally mixed qubits. However, given an arbitrary n-qubit state ρ, one can use the technique of "algorithmic cooling"[23] to unitarily separate this into $n - s$ pure qubits and s maximally mixed qubits, where s is the von Neumann entropy of ρ.

Any $2^n \times 2^n$ unitary matrix can be decomposed as a linear combination of n-fold tensor products of Pauli matrices. As discussed in[20], the problem of estimating a coefficient in the Pauli decomposition of a quantum circuit to polynomial accuracy is a DQC1-complete problem. Estimating the normalized trace of a quantum circuit is a special case of this, and it is also DQC1-complete. This point is discussed in[24]. To make our presentation self-contained, we will sketch here a proof that trace estimation is DQC1-complete. Technically, we should consider the decision problem of determining whether the trace is greater than a given threshold. However, the trace estimation problem is easily reduced to its decision version by the method of binary search, so we will henceforth ignore this point.

First we'll show that trace estimation is contained in DQC1. Suppose we are given a quantum circuit on n qubits which consists of polynomially many gates from some finite universal gate set. Given a state $|\psi\rangle$ of n qubits, there is a standard technique for estimating $\langle\psi| U |\psi\rangle$, called the Hadamard test [3], as shown in figure 1. Now suppose that we use the circuit from figure 1, but choose $|\psi\rangle$ uniformly at random from the 2^n computational basis states. Then the probability of getting outcome $|0\rangle$ for a given measurement will be

$$p_0 = \frac{1}{2^n} \sum_{x \in \{0,1\}^n} \frac{1 + \mathrm{Re}(\langle x| U |x\rangle)}{2} = \frac{1 + \mathrm{Re}(\mathrm{Tr}\ U)}{2^{n+1}}.$$

Choosing $|\psi\rangle$ uniformly at random from the 2^n computational basis states is exactly the same as using the density matrix $I/2^n$ for this register. Thus, the only clean qubit is the control qubit. Trace estimation is therefore achieved in the one clean qubit model by converting the given circuit for U into a circuit for controlled-U and conjugating the control bit with Hadamard gates. One can convert a circuit for U into a circuit for controlled-U by replacing each gate G with a circuit for controlled-G. The overhead incurred is thus bounded by a constant factor [21].

Next we'll show that trace estimation is hard for DQC1. Suppose we are given a classical description of a quantum circuit implementing some unitary transformation U on n qubits. As shown in equation 1, the probability of obtaining outcome $|0\rangle$ from the one clean qubit computation of this circuit is proportional to the trace of the non-unitary operator $(|0\rangle\langle 0| \otimes I)U(|0\rangle\langle 0| \otimes I)U^\dagger$, which acts on n qubits. Estimating this

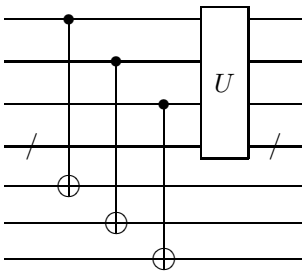
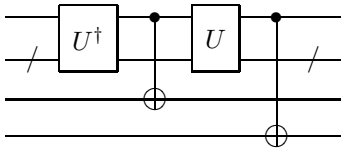


Figure 2: Here CNOT gates are used to simulate 3 clean ancilla qubits.

can be achieved by estimating the trace of

$U' =$ 

which is a unitary operator on $n+2$ qubits. This suffices because

$$\mathrm{Tr}[(|0\rangle\langle 0| \otimes I)U(|0\rangle\langle 0| \otimes I)U^\dagger] = \frac{1}{4}\mathrm{Tr}[U'].$$

To see this, we can think in terms of the computational basis:

$$\mathrm{Tr}[U'] = \sum_{x\in\{0,1\}^n} \langle x| U' |x\rangle .$$

If the first qubit of $|x\rangle$ is $|1\rangle$, then the rightmost CNOT will flip the lowermost qubit. The resulting state will be orthogonal to $|x\rangle$ and the corresponding matrix element will not contribute to the trace. Thus this CNOT gate simulates the initial projector $|0\rangle\langle 0| \otimes I$. Similarly, the other CNOT simulates the other projector.

The preceding analysis shows that, given a description of a quantum circuit implementing a unitary transformation U on n-qubits, the problem of approximating $\frac{1}{2^n}\mathrm{Tr}\, U$ to within $\pm\frac{1}{\mathrm{poly}(n)}$ precision is DQC1-complete.

Some unitaries may only be efficiently implementable using ancilla bits. That is, to implement U on n-qubits using a quantum circuit, it may be most efficient to construct a circuit on $n+m$ qubits which acts as $U \otimes I$, provided that the m ancilla qubits are all initialized to $|0\rangle$. These ancilla qubits are used as work bits in intermediate steps of the computation. To estimate the trace of U, one can construct a circuit U_a on $n+2m$ qubits by adding CNOT gates controlled by the m ancilla qubits and acting on m extra qubits, as shown in figure 2. This simulates the presence of m clean ancilla qubits, because if any of the ancilla qubits is in the $|1\rangle$ state then the CNOT gate will flip the corresponding extra qubit, resulting in an orthogonal state which will not contribute to the trace.

With one clean qubit, one can estimate the trace of U_a to a precision of $\frac{2^{n+2m}}{\mathrm{poly}(n,m)}$. By construction, $\mathrm{Tr}[U_a] = 2^m\mathrm{Tr}[U]$. Thus, if m is logarithmic in n, then one can obtain $\mathrm{Tr}[U]$ to precision $\frac{2^n}{\mathrm{poly}(n)}$, just as can be obtained for circuits not requiring ancilla qubits. This line of reasoning also shows that the k-clean qubit model gives rise to the same complexity class as the one clean qubit model, for any constant k, and even for k growing logarithmically with n.

It seems unlikely that the trace of these exponentially large unitary matrices can be estimated to this precision on a classical computer in polynomial time. Thus it seems unlikely that DQC1 is contained in P. (For more detailed analysis of this point see [9].) However, it also appears unlikely that DQC1 contains all of P. By applying Barrington's theorem[7], it has been shown that DQC1 contains NC1, the class of problems solvable by logarithmic depth classical circuits [4]. These relationships are summarized in figure 3.

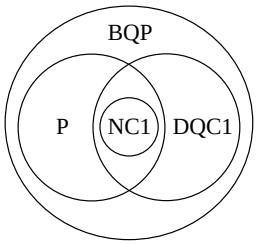


Figure 3: Summary of known relationships between DQC1 and other complexity classes.

Figure 4: Shown from left to right are the unknot, another representation of the unknot, an oriented trefoil knot, and the Hopf link. Broken lines indicate undercrossings.

2 Jones Polynomials

A knot is defined to be an embedding of the circle in $\mathbb{R}^3$ considered up to continuous transformation (isotopy). More generally, a link is an embedding of one or more circles in $\mathbb{R}^3$ up to isotopy. In an oriented knot or link, one of the two possible traversal directions is chosen for each circle. Some examples of knots and links are shown in figure 4. One of the fundamental tasks in knot theory is, given two representations of knots, which may appear superficially different, determine whether these both represent the same knot. In other words, determine whether one knot can be deformed into the other without ever cutting the strand.

Reidemeister showed in 1927 that two knots are the same if and only if one can be deformed into the other by some sequence constructed from three elementary moves, known as the Reidemeister moves, shown in figure 5. This reduces the problem of distinguishing knots to a combinatorial problem, although one for which no efficient solution is known. In some cases, the sequence of Reidemeister moves needed to show equivalence of two knots involves intermediate steps that increase the number of crossings. Thus, it is very difficult to show upper bounds on the number of moves necessary. The most thoroughly studied knot equivalence problem is the problem of deciding whether a given knot is equivalent to the unknot. Even showing the decidability of this problem is highly nontrivial. This was achieved by Haken in 1961[13]. In 1998 it was shown by Hass, Lagarias, and Pippenger that the problem of recognizing the unknot is contained in NP[14].

Starting with the Alexander polynomial, discovered in 1928, a number of knot invariants have been discovered. These are functions which are invariant under Reidemeister moves, thus a knot invariant will always take the same value for different representations of the same knot, such as the two representations of the unknot shown in figure 4. In general, there can be distinct knots which a knot invariant fails to distinguish.

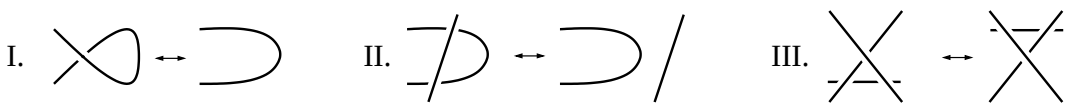


Figure 5: Two knots are the same if and only if one can be deformed into the other by some sequence of the three Reidemeister moves shown above.

One of the best known knot invariants is the Jones polynomial, discovered in 1985 by Vaughan Jones[17]. To any oriented knot or link, it associates a Laurent polynomial in the variable $t^{1/2}$. The Jones polynomial has a degree in t which grows at most linearly with the number of crossings in the link. The coefficients are all integers, but they may be exponentially large. Exact evaluation of Jones polynomials at all but a few special values of t is #P-hard[15]. The Jones polynomial can be defined recursively by a simple "skein" relation. However, for our purposes it will be more convenient to use a definition in terms of a representation of the braid group, as discussed below.

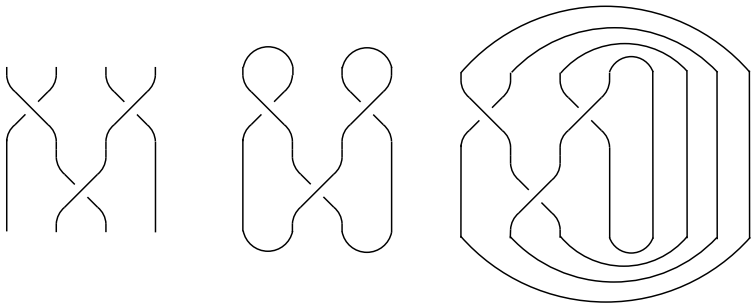


Figure 6: Shown from left to right are a braid, its plat closure, and its trace closure.

To describe in more detail the computation of Jones polynomials we must specify how the knot will be represented on the computer. Although an embedding of a circle in $\mathbb{R}^3$ is a continuous object, all the topologically relevant information about a knot can be described in the discrete language of the braid group. Links can be constructed from braids by joining the free ends. Two ways of doing this are taking the plat closure and the trace closure, as shown in figure 6. Alexander's theorem states that any link can be constructed as the trace closure of some braid. Any link can also be constructed as the plat closure of some braid. This can be easily proven as a corollary to Alexander's theorem, as shown in figure 7.

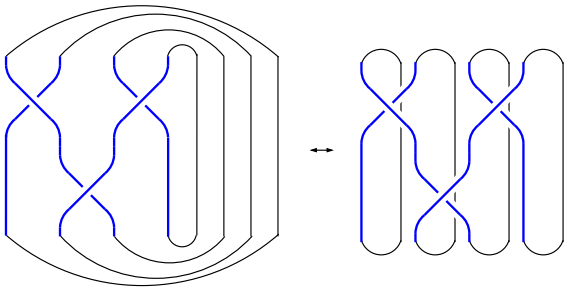


Figure 7: A trace closure of a braid on n strands can be converted to a plat closure of a braid on $2n$ strands by moving the "return" strands into the braid.

Given that the trace closure provides a correspondence between links and braids, one may attempt to find functions on braids which yield link invariants via this correspondence. Markov's theorem shows that a function on braids will yield a knot invariant provided it is invariant under the two Markov moves, shown in figure 8. Thus the Markov moves provide an analogue for braids of the Reidemeister moves on links. The constraints imposed by invariance under the Reidemeister moves are enforced in the braid picture jointly by invariance under Markov moves and by the defining relations of the braid group.

A linear function f satisfying $f(AB) = f(BA)$ is called a trace. The ordinary trace on matrices is one such function. Taking a trace of a representation of the braid group yields a function on braids which is invariant under Markov move I. If the trace and representation are such that the resulting function is also

Figure 8: Shown are the two Markov moves. Here the boxes represent arbitrary braids. If a function on braids is invariant under these two moves, then the corresponding function on links induced by the trace closure is a link invariant.

invariant under Markov move II, then a link invariant will result. The Jones polynomial can be obtained in this way.

In [3], Aharonov, *et al.* show that an additive approximation to the Jones polynomial of the plat or trace closure of a braid at $t = e^{i2\pi/k}$ can be computed on a quantum computer in time which scales polynomially in the number of strands and crossings in the braid and in k. In [1, 27], it is shown that for plat closures, this problem is BQP-complete. The complexity of approximating the Jones polynomial for trace closures was left open, other than showing that it is contained in BQP.

The results of [3, 1, 27] reformulate and generalize the previous results of Freedman *et al.* [12, 11], which show that certain approximations of Jones polynomials are BQP-complete. The work of Freedman *et al.* in turn builds upon Witten's discovery of a connection between Jones polynomials and topological quantum field theory [26]. Recently, Aharonov *et al.* have generalized further, obtaining an efficient quantum algorithm for approximating the Tutte polynomial for any planar graph, at any point in the complex plane, and also showing BQP-hardness at some points [2]. As special cases, the Tutte polynomial includes the Jones polynomial, other knot invariants such as the HOMFLY polynomial, and partition functions for some physical models such as the Potts model.

The algorithm of Aharonov *et al.* works by obtaining the Jones polynomial as a trace of the path model representation of the braid group. The path model representation is unitary and, as shown in [3], can be efficiently implemented by quantum circuits. For computing the trace closure of a braid the necessary trace is similar to the ordinary matrix trace except that only a subset of the diagonal elements of the unitary implemented by the quantum circuit are summed, and there is an additional weighting factor. For the plat closure of a braid the computation instead reduces to evaluating a particular matrix element of the quantum circuit. Aharonov *et al.* also use the path model in their proof of BQP-completeness.

Given a braid b, we know that the problem of approximating the Jones polynomial of its plat closure is BQP-hard. By Alexander's theorem, one can obtain a braid b' whose trace closure is the same link as the plat closure of b. The Jones polynomial depends only on the link, and not on the braid it was derived from. Thus, one may ask why this doesn't immediately imply that estimating the Jones polynomial of the trace closure is a BQP-hard problem. The answer lies in the degree of approximation. As discussed in section 6, the BQP-complete problem for plat closures is to approximate the Jones polynomial to a certain precision which depends exponentially on the number of strands in the braid. The number of strands in b' can be larger than the number of strands in b, hence the degree of approximation obtained after applying Alexander's theorem may be too poor to solve the original BQP-hard problem.

The fact that computing the Jones polynomial of the trace closure of a braid can be reduced to estimating a generalized trace of a unitary operator and the fact that trace estimation is DQC1-complete suggest a connection between Jones polynomials and the one clean qubit model. Here we find such a connection by showing that evaluating a certain approximation to the Jones polynomial Jones polynomial of the trace closure of a braid at a fifth root of unity is DQC1-complete. The main technical difficulty is obtaining the Jones polynomial as a trace over the entire Hilbert space rather than as a summation of some subset of the diagonal matrix elements. To do this we will not use the path model representation of the braid group, but rather the Fibonacci representation, as described in the next section.

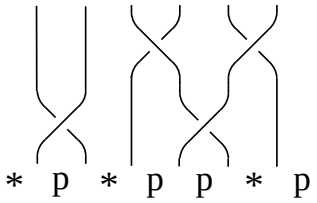


Figure 9: For an n-strand braid we can write a length $n+1$ string of p and $*$ symbols across the base. The string may have no two stars in a row, but can be otherwise arbitrary.

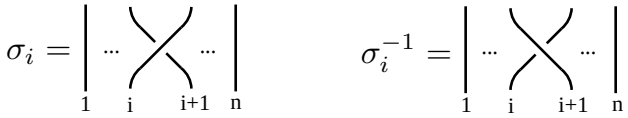


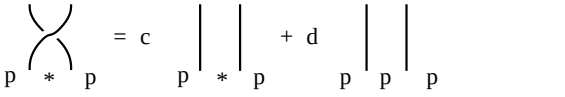
Figure 10: σ_i denotes the elementary crossing of strands i and $i+1$. The braid group on n strands B_n is generated by $\sigma_1 \ldots \sigma_{n-1}$, which satisfy the relations $\sigma_i\sigma_j = \sigma_j\sigma$ for $|i-j| > 1$ and $\sigma_{i+1}\sigma_i\sigma_{i+1} = \sigma_{i+1}\sigma_i\sigma_{i+1}$ for all i. The group operation corresponds to concatenation of braids.

3 Fibonacci Representation

The Fibonacci representation $\rho_F^{(n)}$ of the braid group B_n is described in [18] in the context of Temperley-Lieb recoupling theory. This is a mathematical framework, which in this case describes two species of idealized "particles" denoted by p and $*$. We will not delve into the conceptual and mathematical underpinnings of Temperley-Lieb recoupling theory. For present purposes, it will be sufficient to regard it as a formal procedure for obtaining a particular unitary representation of the braid group whose trace yields the Jones polynomial at $t = -e^{i3\pi/5}$. Throughout most of this paper it will be more convenient to express the Jones polynomial in terms of $A = e^{i3\pi/5}$, with t defined by $t = A^{-4}$.

Given an n-strand braid $b \in B_n$, we can write a length $n+1$ string of p and $*$ symbols across the base as shown in figure 9. These strings have the restriction that no two $*$ symbols can be adjacent. The number of such strings is f_{n+3}, where f_n is the n^{th} Fibonacci number, defined so that $f_1 = 1$, $f_2 = 1$, $f_3 = 2, \ldots$ Thus the formal linear combinations of such strings form an f_{n+3}-dimensional vector space. For each n, the Fibonacci representation $\rho_F^{(n)}$ is a homomorphism from B_n to the group of unitary linear transformations on this space. We will describe the Fibonacci representation in terms of its action on the elementary crossings which generate the braid group, as shown in figure 10.

The elementary crossings correspond to linear operations which mix only those strings which differ by the symbol beneath the crossing. The linear transformations have a local structure, so that the coefficients for the symbol beneath the crossing to be changed or unchanged depend only on that symbol and its two neighbors. For example, using the notation of [18],



(2)

which means that the elementary crossing σ_i corresponds to a linear transformation which takes any string whose i^{th} through $(i+2)^{\text{th}}$ symbols are $p * p$ to the coefficient c times the same string plus the coefficient d times the same string with the $*$ at the $(i+1)^{\text{th}}$ position replaced by p. (As shown in figure 9, the i^{th} crossing is over the $(i+1)^{\text{th}}$ symbol.) To compute the linear transformation that the representation of a given braid applies to a given string of symbols, one can write the symbols across the base of the braid, and then apply rules of the form 2 until all the crossings are removed, and all that remains are various coefficients

for different strings to be written across the base of a set of straight strands.

For compactness, we will use $(p\widehat{*}p) = c(p*p) + d(ppp)$ as a shorthand for equation 2. In this notation, the complete set of rules is as follows.

$$\begin{aligned}
(*\widehat{p}p) &= a(*pp) \\
(*\widehat{p}*) &= b(*p*) \\
(p\widehat{*}p) &= c(p*p) + d(ppp) \\
(p\widehat{p}*) &= a(pp*) \\
(p\widehat{p}p) &= d(p*p) + e(ppp),
\end{aligned} \tag{3}$$

where

$$\begin{aligned}
a &= -A^4 \\
b &= A^8 \\
c &= A^8\tau^2 - A^4\tau \\
d &= A^8\tau^{3/2} + A^4\tau^{3/2} \\
e &= A^8\tau - A^4\tau^2 \\
A &= e^{i3\pi/5} \\
\tau &= 2/(1+\sqrt{5}).
\end{aligned} \tag{4}$$

Using these rules we can calculate any matrix from the Fibonacci representation of the braid group. Notice that this is a reducible representation. These rules do not allow the rightmost symbol or leftmost symbol of the string to change. Thus the vector space decomposes into four invariant subspaces, namely the subspace spanned by strings which begin and end with p, and the $*\ldots*$, $p\ldots*$, and $*\ldots p$ subspaces. As an example, we can use the above rules to compute the action of B_3 on the $*\ldots p$ subspace.

$$\rho^{(3)}_{*p}(\sigma_1) = \begin{bmatrix} b & 0 \\ 0 & a \end{bmatrix} \begin{matrix} *\text{p}* \text{p} \\ *\text{pp} \end{matrix} \qquad \rho^{(3)}_{*p}(\sigma_2) = \begin{bmatrix} c & d \\ d & e \end{bmatrix} \begin{matrix} *\text{p}* \text{p} \\ *\text{pp} \end{matrix} \tag{5}$$

In appendix A we prove that the Jones polynomial evaluated at $t = -e^{i3\pi/5}$ can be obtained as a weighted trace of the Fibonacci representation over the $*\ldots*$ and $*\ldots p$ subspaces.

4 Computing the Jones Polynomial in DQC1

As mentioned previously, the Fibonacci representation acts on the vector space of formal linear combinations of strings of p and $*$ symbols in which no two $*$ symbols are adjacent. The set of length n strings of this type, P_n, has f_{n+2} elements, where f_n is the n^{th} Fibonacci number: $f_1 = 1$, $f_2 = 1$, $f_3 = 2$, and so on. As shown in appendix C, one can construct a bijective correspondence between these strings and the integers from 0 to $f_{n+2} - 1$ as follows. If we think of $*$ as 1 and p as 0, then with a string $s_n s_{n-1} \ldots s_1$ we associate the integer

$$z(s) = \sum_{i=1}^{n} s_i f_{i+1}. \tag{6}$$

This is known as the Zeckendorf representation.

Representing integers as bitstrings by the usual method of place value, we thus have a correspondence between the elements of P_n and the bitstrings of length $b = \lceil \log_2(f_{n+2}) \rceil$. This correspondence will be a key element in computing the Jones polynomial with a one clean qubit machine. Using a one clean qubit machine, one can compute the trace of a unitary over the entire Hilbert space of 2^n bitstrings. Using CNOT gates as above, one can also compute with polynomial overhead the trace over a subspace whose dimension is a

polynomially large fraction of the dimension of the entire Hilbert space. However, it is generally not possible to compute the trace over subspaces whose dimension is an exponentially small fraction of the dimension of the total Hilbert space. For this reason, directly mapping the strings of p and $*$ symbols to strings of 1 and 0 will not work. In contrast, the correspondence described in equation 6 maps P_n to a subspace whose dimension is at least half the dimension of the full 2^b-dimensional Hilbert space.

In outline, the DQC1 algorithm for computing the Jones polynomial works as follows. Using the results described in section 1, we will think of the quantum circuit as acting on b maximally mixed qubits plus $O(1)$ clean qubits. Thinking in terms of the computational basis, we can say that the first b qubits are in a uniform probabilistic mixture of the 2^b classical bitstring states. By equation 6, most of these bitstrings correspond to elements of P_n. In the Fibonacci representation, an elementary crossing on strands i and $i-1$ corresponds to a linear transformation which can only change the value of the i^{th} symbol in the string of p's and $*$'s. The coefficients for changing this symbol or leaving it fixed depend only on the two neighboring symbols. Thus, to simulate this linear transformation, we will use a quantum circuit which extracts the $(i-1)^{\text{th}}$, i^{th}, and $(i+1)^{\text{th}}$ symbols from their bitstring encoding, writes them into an ancilla register while erasing them from the bitstring encoding, performs the unitary transformation prescribed by equation 3 on the ancillas, and then transfers this symbol back into the bitstring encoding while erasing it from the ancilla register. Constructing one such circuit for each crossing, multiplying them together, and performing DQC1 trace-estimation yields an approximation to the Jones polynomial.

Performing the linear transformation demanded by equation 3 on the ancilla register can be done easily by invoking gate set universality (*cf.* Solovay-Kitaev theorem [21]) since it is just a three-qubit operation. The harder steps are transferring the symbol values from the bitstring encoding to the ancilla register and back.

It may be difficult to extract an arbitrary symbol from the bitstring encoding. However, it is relatively easy to extract the leftmost "most significant" symbol, which determines whether the Fibonacci number f_n is present in the sum 6. This is because, for a string s of length n, $z(s) \geq f_{n-1}$ if and only if the leftmost symbol is $*$. Thus, starting with a clean $|0\rangle$ ancilla qubit, one can transfer the value of the leftmost symbol into the ancilla as follows. First, check whether $z(s)$ (as represented by a bitstring using place value) is $\geq f_{n-1}$. If so flip the ancilla qubit. Then, conditioned on the value of the ancilla qubit, subtract f_{n-1} from the bitstring. (The subtraction will be done $\mod 2^b$ for reversibility.)

The basic operations of arithmetic and comparison for integers can all be done classically by NC1 circuits [25]. It follows that this procedure can be done in DQC1. More specifically, Krapchenko's algorithm for adding two n-bit numbers has depth $\lceil \log n \rceil + O(\sqrt{\log n})$ [25]. A lower bound of depth $\log n$ is also known, so this is essentially optimal [25]. Barrington's construction [7] yields a sequence of 2^{2d} gates on 3 clean ancilla qubits [4] to simulate a circuit of depth d. Thus we obtain an addition circuit which has quadratic size (up to a subpolynomial factor). Subtraction can be obtained analogously, and one can determine whether $a \geq b$ can be done by subtracting a from b and looking at whether the result is negative.

Although the construction based on Barrington's theorem has polynomial overhead and is thus sufficient for our purposes, it seems worth noting that it is possible to achieve better efficiency. As shown by Draper [10], there exist ancilla-free quantum circuits for performing addition and subtraction, which succeed with high probability and have nearly linear size. Specifically, one can add or subtract a hardcoded number a into an n-qubit register $|x\rangle$ modulo 2^n by performing quantum Fourier transform, followed by $O(n^2)$ controlled-rotations, followed by an inverse quantum Fourier transform. Furthermore, using approximate quantum Fourier transforms[6], [10] describes an approximate version of the circuit, which, for any value of parameter m, uses a total of only $O(mn \log n)$ gates to produce an output having an inner product with $|x + a \mod 2^n\rangle$ of $1 - O(2^{-m})$ [1].

Because they operate modulo 2^n, Draper's quantum circuits for addition and subtraction do not immediately yield fast ancilla-free quantum circuits for comparison, unlike the classical case. Instead, start with an n-bit number x and then introduce a single clean ancilla qubit initialized to $|0\rangle$. Then subtract an n-bit hardcoded number a from this register modulo 2^{n+1}. If $a > x$ then the result will wrap around into the

[1] A linear-size quantum circuit for exact ancilla-free addition is known, but it does not generalize easily to the case of hardcoded summands [8].

1	2	3	5	13	8	5	3	2	1	$\leftrightarrow$	$(6,5)$
$*$	p	p	$*$	p	p	$*$	p	p	p		

Figure 11: Here we make a correspondence between strings of p and $*$ symbols and ordered pairs of integers. The string of 9 symbols is split into substrings of length 4 and 5, and each one is used to compute an integer by adding the $(i+1)^{\text{th}}$ Fibonacci number if $*$ appears in the i^{th} place. Note the two strings are read in different directions.

range $[2^n, 2^{n+1}-1]$, in which case the leading bit will be 1. If $a \leq x$ then the result will be in the range $[0, 2^n - 1]$. After copying the result of this leading qubit and uncomputing the subtraction, the comparison is complete. Alternatively, one could use the linear size quantum comparison circuit devised by Takahashi and Kunihiro, which uses n uninitialized ancillas but no clean ancillas.

Unfortunately, most crossings in a given braid will not be acting on the leftmost strands. However, we can reduce the problem of extracting a general symbol to the problem of extracting the leftmost symbol. Rather than using equation 6 to make a correspondence between an string from P_n and a single integer, we can split the string at some chosen point, and use equation 6 on each piece to make a correspondence between elements of P_n and ordered pairs of integers, as shown in figure 11. To extract the i^{th} symbol, we thus convert encoding 6 to the encoding where the string is split between the i^{th} and $(i-1)^{\text{th}}$ symbols, so that one only needs to extract the leftmost symbol of the second string. Like equation 6, this is also an efficient encoding, in which the encoded bitstrings form a large fraction of all possible bitstrings.

To convert encoding 6 to a split encoding with the split at an arbitrary point, we can move the split rightward by one symbol at a time. To introduce a split between the leftmost and second-to-leftmost symbols, one must extract the leftmost symbol as described above. To move the split one symbol to the right, one must extract the leftmost symbol from the right string, and if it is $*$ then add the corresponding Fibonacci number to the left string. This is again a procedure of addition, subtraction, and comparison of integers. Note that the computation of Fibonacci numbers in NC1 is not necessary, as these can be hardcoded into the circuits. Moving the split back to the left works analogously. As crossings of different pairs of strands are being simulated, the split is moved to the place that it is needed. At the end it is moved all the way leftward and eliminated, leaving a superposition of bitstrings in the original encoding, which have the correct coefficients determined by the Fibonacci representation of the given braid.

Lastly, we must consider the weighting in the trace, as described by equation 9. Instead of weight W_s, we will use W_s/ϕ so that the possible weights are 1 and $1/\phi$ both of which are ≤ 1. We can impose any weight ≤ 1 by doing a controlled rotation on an extra qubit. The CNOT trick for simulating a clean qubit which was described in section 1 can be viewed as a special case of this. All strings in which that qubit takes the value $|1\rangle$ have weight zero, as imposed by a $\pi/2$ rotation on the extra qubit. The weighting will cause only a polynomial overhead in the number of measurements needed to get a given precision provided that $\sum_s |W_s|^2$ is only polynomially small. This is clearly the case for W_s as defined in 9.

5 DQC1-hardness of Jones Polynomials

As mentioned in section 3, the Fibonacci representation $\rho_F^{(n)}$ is reducible. Let $\rho_{**}^{(n)}$ denote the representation of the braid group B_n defined by the action of $\rho_F^{(n)}$ on the vector space spanned by strings which begin and end with $*$. As shown in appendix B, $\rho_{**}^{(n)}(B_n)$ taken modulo phase is a dense subgroup of $SU(f_{n-1})$. Similarly, $\rho_{*p}^{(n)}(B_n)$ and $\rho_{p*}^{(n)}(B_n)$ are dense subgroups of $SU(f_n)$ modulo phase. However, $\rho_{pp}^{(n)}(B_n)$ is not a dense subgroup of the corresponding unitary group. This necessitates some additional care in proving the DQC1-hardness result.

Given a quantum circuit, we will construct a braid whose Fibonacci representation performs essentially a gate-by-gate simulation of the circuit. To do this, we will use a version of the Solovay-Kitaev theorem which appears in [19].

Theorem 1 (Solovay-Kitaev) *Suppose matrices $U_1, \ldots, U_r$ generate a dense subgroup in $SU(d)$. Then, given a desired unitary $U \in SU(d)$, and a precision parameter $\delta > 0$, there is an algorithm to find a product V of $U_1, \ldots, U_r$ and their inverses such that $\|V - U\| \leq \delta$. The length of the product and the runtime of the algorithm are both polylogarithmic in $1/\delta$ and polynomial in d and r.*

Next, we must specify an encoding, that is, a map $\eta : Q_n \to S_m$ from the set Q_n of strings of p and $*$ symbols which start with $*$ and have no two $*$ symbols in a row, to S_m, the set of bitstrings of length m. Unlike in section 4, we will not use a one to one encoding between bit strings and strings of p and $*$ symbols. All we require is that a sum over all strings of p and $*$ symbols corresponds to a sum over bitstrings in which each bitstring appears an equal number of times. Equivalently, all bitstrings $b \in S_m$ must have a preimage $\eta^{-1}(b)$ of the same size. This insures an unbiased trace in which no bitstrings are overweighted. To achieve this we can use the encoding

$$\begin{array}{lcl} \text{ppp} & \to & 0 \\ \text{p*p} & \to & 1 \end{array} \tag{7}$$

The strings other than ppp and p$*$p do not correspond to any bit value. Since both the encoded 1 and the encoded 0 begin and end with p, they can be preceded and followed by any allowable string. Thus, changing an encoded 1 to an encoded zero does not change the number of allowed strings of p and $*$ consistent with that encoded bitstring. Thus the condition that $|\eta^{-1}(b)|$ be independent of b is satisfied.

We would also like to know *a priori* where in the string of p and $*$ symbols a given bit is encoded. This way, when we need to simulate a gate acting on a given bit, we would know which strands the corresponding braid should act on. If we were to simply divide our string of symbols into blocks of three and write down the corresponding bit string (skipping every block which is not in one of the two coding states ppp and p$*$p) then this would not be the case. Thus, to encode n bits, we will instead divide the string of symbols into n blocks, each of size $c \log n$ for some constant c. To decode a block, scan it from left to right until you reach either a ppp string or a p$*$p string. The first such string encountered determines whether the block encodes a 1 or a 0, according to equation 7. Now imagine we choose a string randomly from $Q_{cn \log n}$. By choosing the constant prefactor c in our block size we can ensure that in the entire $cn \log n$ string of symbols, the probability of there being any noncoding block which contains neither ppp nor p$*$p anywhere within it is polynomially small. If this is the case, then these noncoding strings will contribute only a polynomially small additive error to the estimate of the circuit trace, on par with the other sources of error.

The gate set consisting of the CNOT, Hadamard, and $\pi/8$ gates is known to be universal for BQP [21]. Thus, it suffices to consider the simulation of 1-qubit and 2-qubit gates. Furthermore, it is sufficient to imagine the qubits arranged on a line and to allow 2-qubit gates to act only on neighboring qubits. This is because qubits can always be brought into neighboring positions by applying a series of SWAP gates to nearest neighbors. By our encoding a unitary gate applied to qubits i and $i+1$ will correspond to a unitary transformation on symbols $ic \log n$ through $(i+3)c \log n - 1$.

By the Solovay-Kitaev theorem, one can efficiently construct arbitrary unitaries in $SU(d)$ where d is polynomial. Hence, we can obtain arbitrary unitaries on strings of logarithmically many p and $*$ symbols. Thus if a $*$ symbol were known to be within logarithmic distance of the target symbols then by the density results of appendix B and the Solovay-Kitaev theorem, a braid corresponding to the desired gate could be efficiently constructed. However, this will not be the case for all strings in the trace. Thus we need a way to unitarily bring $*$ symbols to where they are needed.

To do this we'll use an "inchworm" structure which brings a pair of $*$ symbols rightward to where they are needed. Specifically, suppose we have a pair of blocks which each have a $*$ in their exact center[2]. The presence of the left $*$ and the density of ρ_{*p} allow us to use the Solovay-Kitaev theorem to unitarily move the right $*$ one block to the right by adding polynomially many crossings to the braid. Then, the presence of the right $*$ and the density of ρ_{p*} allow us to similarly move the left $*$ one block to the right, thus bringing it into the block adjacent to the one which contains the right $*$. To move the inchworm to the left we use the inverse operation.

[2]We can always choose the block size to be odd.

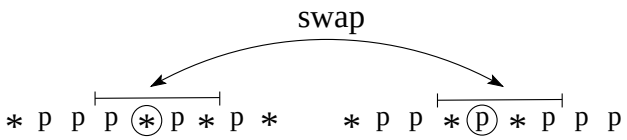


Figure 12: This unitary procedure starts with a $*$ in the current block and brings it to the center of the target block.

To simulate a given gate, one first uses the previously described procedure to make the inchworm crawl to the blocks just to the left of the blocks which encode the qubits on which the gate acts. Then, by the density of ρ_{*p} and the Solovay-Kitaev theorem, the desired gate can be simulated using polynomially many braid crossings.

It remains to specify the exact unitary operations which move the inchworm. Suppose we have a current block and a target block. The current block contains a $*$ in its center, and the target block is the next block to the right or left. We wish to move the $*$ to the center of the target block. To do this, we can select the smallest segment around the center such that in each of these blocks, the segment is bordered on its left and right by p symbols. This segment can then be swapped, as shown in figure 12.

For some possible strings this procedure will not be well defined. Specifically there may not be any segment which contains the center and which is bordered by p symbols in both blocks. On such strings we define the operation to act as the identity. For random strings, the probability of this decreases exponentially with the block size. Thus, by choosing c sufficiently large we can make this negligible for the entire computation.

As the inchworm moves rightward, it leaves behind a trail. Due to the swapping, the blocks are not in their original state after the inchworm has passed. However, because the operations are unitary, when the inchworm moves back to the left, the modifications to the blocks get undone. Thus the inchworm can shuttle back and forth, moving where it is needed to simulate each gate, always stopping just to the left of the blocks corresponding to the encoded qubits.

To get this process started, the leftmost two blocks must each contain a $*$ at their center. This occurs with constant probability. The strings in which this is not the case can be prevented from contributing to the trace by a technique analogous to that used in section 1 to simulate logarithmically many clean ancillas. Namely, an extra encoded qubit can be conditionally flipped if the first two blocks do not both have $*$ symbols at their center. This can always be done using the Solovay-Kitaev theorem, since the leftmost symbol in the string is always $*$, and the ρ_{*p} and ρ_{**} representations are both dense.

The only remaining detail to consider is that the trace appearing in the Jones polynomial is weighted depending on whether the last symbol is p or $*$, whereas the DQC1-complete trace estimation problem is for completely unweighted traces. This problem is easily solved. Just introduce a single extra logarithmic size block of symbols at the end of the string. After bringing the inchworm adjacent to the last block, apply a unitary which performs a conditional rotation on the qubit encoded by this block. The rotation will be by an angle so that the inner product of the rotated qubit with its original state is $1/\phi$ where ϕ is the golden ratio. This will be done only if the last symbol is p. This exactly cancels out the weighting which appears in the formula for the Jones polynomial, as described in appendix A.

Thus, for appropriate ϵ, approximating the Jones polynomial of the trace closure of a braid to within $\pm\epsilon$ is DQC1-hard.

6 Conclusion

The preceding sections show that the problem of approximating the Jones polynomial of the trace closure of a braid with n strands and m crossings to within $\pm\epsilon$ at $t = -e^{i3\pi/5}$ is a DQC1-complete problem for appropriate ϵ. The proofs are based on the problem of evaluating the Markov trace of the Fibonacci representation of a braid to $\frac{1}{\mathrm{poly}(n,m)}$ precision. By equation 10, we see that this corresponds to evaluating the Jones polynomial with $\pm\frac{|D^{n-1}|}{\mathrm{poly}(n,m)}$ precision, where $D = -A^2 - A^{-2} = 2\cos(6\pi/5)$. Whereas approximating

the Jones polynomial of the plat closure of a braid was known[1] to be BQP-complete, it was previously only known that the problem of approximating the Jones polynomial of the trace closure of a braid was in BQP. Understanding the complexity of approximating the Jones polynomial of the trace closure of a braid to precision $\pm\frac{|D^{n-1}|}{\mathrm{poly}(n,m)}$ was posed as an open problem in [3]. This paper shows that for $A = e^{i3\pi/5}$, this problem is DQC1-complete. Such a completeness result improves our understanding of both the difficulty of the Jones polynomial problem and the power one clean qubit computers, by finding an equivalence between the two.

It is generally believed that DQC1 is not contained in P and does not contain all of BQP. The DQC1-completeness result shows that if this belief is true, it implies that approximating the Jones polynomial of the trace closure of a braid is not so easy that it can be done classically in polynomial time, but is not so difficult as to be BQP-hard.

To our knowledge, the problem of approximating the Jones polynomial of the trace closure of a braid is one of only two known candidates for classically intractable problems solvable on a one clean qubit computer. The other is estimating the Pauli decomposition of the unitary matrix corresponding to a polynomial-size quantum circuit[3], as described in [20, 24] and recounted in section 1. Since the Pauli decomposition problem is explicitly formulated in terms of quantum circuits, the Jones polynomial problem is thus the only candidate problem which comes from outside of quantum computation.

7 Acknowledgements

The authors thank David Vogan, Pavel Etingof, Wim van Dam, Aram Harrow, and Daniel Nagaj for useful discussions. PS gratefully acknowledges support from the W. M. Keck foundation and from the NSF under grant number CCF-0431787. SJ gratefully acknowledges support from ARO/DTO's QuaCGR program.

A Jones Polynomials by Fibonacci Representation

For any braid $b \in B_n$ we will define $\widetilde{\mathrm{Tr}}(b)$ by:

$$\widetilde{\mathrm{Tr}}(b) = \frac{1}{\phi f_n + f_{n-1}} \sum_{s \in Q_{n+1}} W_s \quad \begin{array}{c} \mathrm{s} \\ \diagup \\ \boxed{\mathrm{b}} \\ \diagup \\ \mathrm{s} \end{array} \tag{8}$$

We will use | to denote a strand and $\nmid$ to denote multiple strands of a braid (in this case n). Q_{n+1} is the set of all strings of $n+1$ p and $*$ symbols which start with $*$ and contain no two $*$ symbols in a row. The symbol

$$\begin{array}{c} \mathrm{s} \\ \diagup \\ \boxed{\mathrm{b}} \\ \diagup \\ \mathrm{s} \end{array}$$

denotes the s, s matrix element of the Fibonacci representation of braid b. The weight W_s is

$$W_s = \begin{cases} \phi & \text{if } s \text{ ends with } p \\ 1 & \text{if } s \text{ ends with } *. \end{cases} \tag{9}$$

ϕ is the golden ratio $(1+\sqrt{5})/\sqrt{2}$.

As discussed in [3], the Jones polynomial of the trace closure of a braid b is given by

$$V_{b^{\mathrm{tr}}}(A^{-4}) = (-A)^{3w(b^{\mathrm{tr}})} D^{n-1} \mathrm{Tr}(\rho_A(b^{\mathrm{tr}})). \tag{10}$$

[3]This includes estimating the trace of the unitary as a special case.

b^{tr} is the link obtained by taking the trace closure of braid b. $w(b^{\mathrm{tr}})$ is denotes the writhe of the link b^{tr}. For an oriented link, one assigns a value of $+1$ to each crossing of the form  , and the value -1 to each crossing of the form  . The writhe of a link is defined to be the sum of these values over all crossings. D is defined by $D = -A^2 - A^{-2}$. $\rho_A : B_n \to \mathrm{TL}_n(D)$ is a representation from the braid group to the Temperley-Lieb algebra with parameter D. Specifically,

$$\rho_A(\sigma_i) = AE_i + A^{-1}\mathbb{1} \tag{11}$$

where $E_1 \ldots E_n$ are the generators of $\mathrm{TL}_n(D)$, which satisfy the following relations.

$$E_iE_j = E_jE_i \quad \text{for } |i-j| > 1 \tag{12}$$
$$E_iE_{i\pm1}E_i = E_i \tag{13}$$
$$E_i^2 = DE_i \tag{14}$$

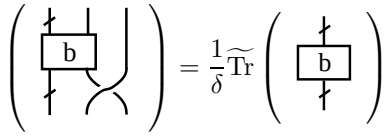
The Markov trace on $\mathrm{TL}_n(D)$ is a linear map $\mathrm{Tr} : \mathrm{TL}_n(D) \to \mathbb{C}$ which satisfies

$$\mathrm{Tr}(\mathbb{1}) = 1 \tag{15}$$
$$\mathrm{Tr}(XY) = \mathrm{Tr}(YX) \tag{16}$$
$$\mathrm{Tr}(XE_{n-1}) = \frac{1}{D}\mathrm{Tr}(X') \tag{17}$$

On the left hand side of equation 17, the trace is on $\mathrm{TL}_n(D)$, and X is an element of $\mathrm{TL}_n(D)$ not containing E_{n-1}. On the right hand side of equation 17, the trace is on $\mathrm{TL}_{n-1}(D)$, and X' is the element of $\mathrm{TL}_{n-1}(D)$ which corresponds to X in the obvious way since X does not contain E_{n-1}.

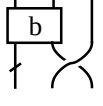
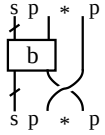
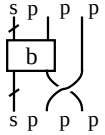
We'll show that the Fibonacci representation satisfies the properties implied by equations 11, 12, 13, and 14. We'll also show that $\widetilde{\mathrm{Tr}}$ on the Fibonacci representation satisfies the properties corresponding to 15, 16, and 17. It was shown in [3] that properties 15, 16, and 17, along with linearity, uniquely determine the map Tr. It will thus follow that $\widetilde{\mathrm{Tr}}(\rho_F^{(n)}(b)) = \mathrm{Tr}(\rho_A(b))$, which proves that the Jones polynomial is obtained from the trace $\widetilde{\mathrm{Tr}}$ of the Fibonacci representation after multiplying by the appropriate powers of D and $(-A)$ as shown in equation 10. Since these powers are trivial to compute, the problem of approximating the Jones polynomial at $A = e^{i3\pi/5}$ reduces to the problem of computing this trace.

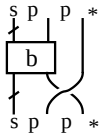
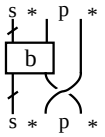
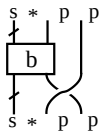
$\widetilde{\mathrm{Tr}}$ is equal to the ordinary matrix trace on the subspace of strings ending in $*$ plus ϕ times the matrix trace on the subspace of strings ending in p. Thus the fact that the matrix trace satisfies property 16 immediately implies that $\widetilde{\mathrm{Tr}}$ does too. Furthermore, since the dimensions of these subspaces are f_{n-1} and f_n respectively, we see from equation 8 that $\widetilde{\mathrm{Tr}}(\mathbb{1}) = 1$. To address property 17, we'll first show that

$\widetilde{\mathrm{Tr}}$  $= \frac{1}{\delta}\widetilde{\mathrm{Tr}}$ (18)

for some constant δ which we will calculate. We will then use equation 11 to relate δ to D.

Using the definition of $\widetilde{\mathrm{Tr}}$ we obtain

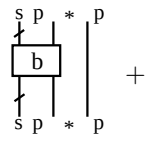
$\widetilde{\mathrm{Tr}}$  $= \frac{1}{f_n\phi + f_{n-1}}\Big[\phi \sum_{s\in Q_{n-2}}$  $+\phi \sum_{s\in Q_{n-2}}$ 

$+ \sum_{s\in Q_{n-2}}$  $+ \sum_{s\in Q'_{n-2}}$  $+\phi \sum_{s\in Q'_{n-2}}$  $\Big]$

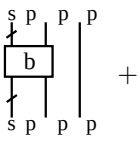
where Q'_{n-2} is the set of length $n-2$ strings of $*$ and p symbols which begin with $*$, end with p, and have no two $*$ symbols in a row.

Next we expand according to the braiding rules described in equations 2 and 3.

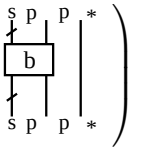
$$= \frac{1}{f_n\phi + f_{n-1}} \left[\sum_{s \in Q_{n-2}} \left(\phi c \; \right. \right.$$



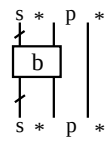
$$+ \phi e$$



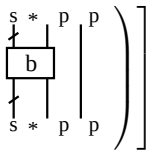
$$+ a$$



$$\left. \left) + \sum_{s \in Q'_{n-2}} \left(b \; \right. \right.$$



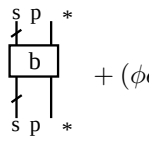
$$+ \phi a$$



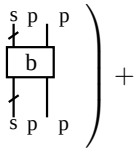
$$\left. \left) \right] \right.$$

We know that matrix elements in which differing string symbols are separated by unbraided strands will be zero. To obtain the preceding expression we have omitted such terms. Simplifying yields

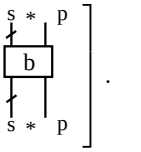
$$= \frac{1}{f_n + \phi f_{n-1}} \left[\sum_{s \in Q_{n-2}} \left(\phi c \; \right. \right.$$



$$+ (\phi e + a)$$



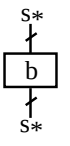
$$\left. \left) + \sum_{s \in Q'_{n-2}} (b + \phi a) \right. \right.$$



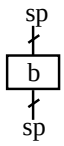
$$\left. \right] .$$

By the definitions of A, a, b, and e, given in equation 4, we see that $\phi e + a = b + \phi a$. Thus the above expression simplifies to

$$= \frac{1}{f_n\phi + f_{n-1}} \left[\sum_{s \in Q'_{n-1}} \phi c \right.$$



$$+ \sum_{s \in Q_{n-1}} (\phi e + a)$$



$$\left. \right]$$

Now we just need to show that

$$\frac{\phi c}{f_n\phi + f_{n-1}} = \frac{1}{\delta} \frac{1}{f_{n-1}\phi + f_{n-2}} \tag{19}$$

and

$$\frac{\phi e + a}{f_n\phi + f_{n-1}} = \frac{1}{\delta} \frac{\phi}{f_{n-1}\phi + f_{n-2}}. \tag{20}$$

The Fibonacci numbers have the property

$$\frac{f_n\phi + f_{n-1}}{f_{n-1}\phi + f_{n-2}} = \phi$$

for all n. Thus equations 19 and 20 are equivalent to

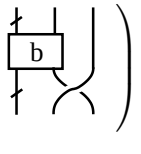
$$\phi c = \frac{1}{\delta}\phi \tag{21}$$

and

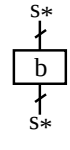
$$\phi e + a = \frac{1}{\delta}\phi^2 \tag{22}$$

respectively. For $A = e^{i3\pi/5}$ these both yield $\delta = A - 1$. Hence

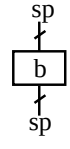
$$\widetilde{\mathrm{Tr}} \left(\right.$$



$$\left. \right) = \frac{1}{\delta} \frac{1}{f_{n-1}\phi + f_{n-2}} \left[\sum_{s \in Q'_{n-1}} \right.$$



$$+ \sum_{s \in Q_{n-1}} \phi$$



$$\left. \right]$$

$$= \frac{1}{\delta}\widetilde{\mathrm{Tr}}(b),$$

thus confirming equation 18.

Now we calculate D from δ. Solving 11 for E_i yields

$$E_i = A^{-1}\rho_A(\sigma_i) - A^{-2}\mathbb{1} \tag{23}$$

Substituting this into 17 yields

$$\mathrm{Tr}(X(A^{-1}\rho_A(\sigma_i) - A^{-2}\mathbb{1})) = \frac{1}{D}\mathrm{Tr}(X)$$

$$\Rightarrow A^{-1}\mathrm{Tr}(X\rho_A(\sigma_i)) - A^{-2}\mathrm{Tr}(X) = \frac{1}{D}\mathrm{Tr}(X).$$

Comparison to our relation $\mathrm{Tr}(X\rho_A(\sigma_i)) = \frac{1}{\delta}\mathrm{Tr}(X)$ yields

$$A^{-1}\frac{1}{\delta} - A^{-2} = \frac{1}{D}.$$

Solving for D and substituting in $A = e^{i3\pi/5}$ yields

$$D = \phi.$$

This is also equal to $-A^2 - A^{-2}$ consistent with the usage elsewhere.

Thus we have shown that $\widetilde{\mathrm{Tr}}$ has all the necessary properties. We will next show that the image of the representation ρ_F of the braid group B_n also forms a representation of the Temperley-Lieb algebra $TL_n(D)$. Specifically, E_i is represented by

$$E_i \to A^{-1}\rho_F^{(n)}(\sigma_i) - A^{-2}\mathbb{1}. \tag{24}$$

To show that this is a representation of $TL_n(D)$ we must show that the matrices described in equation 24 satisfy the properties 12, 13, and 14. By the theorem of [3] which shows that a Markov trace on any representation of the Temperley-Lieb algebra yields the Jones polynomial, it will follow that the trace of the Fibonacci representation yields the Jones polynomial.

Since ρ_F is a representation of the braid group and $\sigma_i\sigma_j = \sigma_j\sigma_i$ for $|i-j| > 1$, it immediately follows that the matrices described in equation 24 satisfy condition 12. Next, we'll consider condition 14. By inspection of the Fibonacci representation as given by equation 3, we see that by appropriately ordering the basis[4] we can bring $\rho_A(\sigma_i)$ into block diagonal form, where each block is one of the following 1×1 or 2×2 possibilities.

$$[a] \quad [b] \quad \begin{bmatrix} c & d \\ d & e \end{bmatrix}$$

Thus, by equation 24, it suffices to show that

$$\left(A^{-1}\begin{bmatrix} c & d \\ d & e \end{bmatrix} - A^{-2}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right)^2 = D\left(A^{-1}\begin{bmatrix} c & d \\ d & e \end{bmatrix} - A^{-2}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right),$$

$$(A^{-1}a - A^{-2})^2 = D(A^{-1}a - A^{-2}),$$

and

$$(A^{-1}b - A^2)^2 = D(A^{-1}b - A^2),$$

i.e. each of the blocks square to D times themselves. These properties are confirmed by direct calculation.

[4] We will have to choose different orderings for different σ_i's.

Now all that remains is to check that the correspondence 24 satisfies property 13. Using the rules described in equation 3 we have

$$\rho_F^{(3)}(\sigma_1) = \begin{bmatrix} b & 0 & & & & & & \\ 0 & a & & & & & & \\ & & a & & & & & \\ & & & e & 0 & d & & \\ & & & 0 & a & 0 & & \\ & & & d & 0 & c & & \\ & & & & & & e & d \\ & & & & & & d & c \end{bmatrix} \begin{matrix} *\text{p}*\text{p} \\ *\text{ppp} \\ *\text{pp}* \\ \text{pppp} \\ \text{pp}*\text{p} \\ \text{p}*\text{pp} \\ \text{ppp}* \\ \text{p}*\text{p}* \end{matrix} \qquad \rho_F^{(3)}(\sigma_2) = \begin{bmatrix} c & d & & & & & & \\ d & e & & & & & & \\ & & a & & & & & \\ & & & e & d & 0 & & \\ & & & d & c & 0 & & \\ & & & 0 & 0 & a & & \\ & & & & & & a & 0 \\ & & & & & & 0 & b \end{bmatrix} \begin{matrix} *\text{p}*\text{p} \\ *\text{ppp} \\ *\text{pp}* \\ \text{pppp} \\ \text{pp}*\text{p} \\ \text{p}*\text{pp} \\ \text{ppp}* \\ \text{p}*\text{p}* \end{matrix}$$

(Here we have considered all four subspaces unlike in equation 5.) Substituting this into equation 24 yields matrices which satisfy condition 14. It follows that equation 24 yields a representation of the Temperley-Lieb algebra. This completes the proof that

$$V_{b^{\text{tr}}}(A^{-4}) = (-A)^{3w(b^{\text{tr}})} D^{n-1} \widetilde{\text{Tr}}(\rho_F^{(n)}(b^{\text{tr}}))$$

for $A = e^{i3\pi/5}$.

B Density of the Fibonacci representation

In this appendix we will show that $\rho_{**}^{(n)}(B_n)$ is a dense subgroup of $SU(f_{n-1})$ modulo phase, and that $\rho_{*p}^{(n)}(B_n)$ and $\rho_{*p}^{(n)}(B_n)$ are dense subgroups of $SU(f_n)$ modulo phase. Similar results regarding the path model representation of the braid group were proven in [1]. Our proofs will use many of the techniques introduced there.

We'll first show that $\rho_{**}^{(4)}(B_4)$ modulo phase is a dense subgroup of $SU(2)$. We can then use the bridge lemma from [1] to extend the result to arbitrary n.

Proposition 1 *$\rho_{**}^{(4)}(B_4)$ modulo phase is a dense subgroup of $SU(2)$.*

Proof: Using equation 3 we have:

$$\rho_{**}^{(4)}(\sigma_1) = \rho_{**}^{(4)}(\sigma_3) = \begin{bmatrix} b & 0 \\ 0 & a \end{bmatrix} \begin{matrix} *\text{p}*\text{p}* \\ *\text{ppp}* \end{matrix} \qquad \rho_{**}^{(4)}(\sigma_2) = \begin{bmatrix} c & d \\ d & e \end{bmatrix} \begin{matrix} *\text{p}*\text{p}* \\ *\text{ppp}* \end{matrix}$$

We do not care about global phase so we will take

$$\rho_{**}^{(n)}(\sigma_i) \quad \rightarrow \quad \frac{1}{\left(\det \rho_{**}^{(n)}(\sigma_i)\right)^{1/f_{n-1}}} \rho_{**}^{(n)}(\sigma_i)$$

to project into $SU(f_{n-1})$. Thus we must show the group $\langle A, B \rangle$ generated by

$$A = \frac{1}{\sqrt{ab}} \begin{bmatrix} b & 0 \\ 0 & a \end{bmatrix} \qquad B = \frac{1}{\sqrt{ce - d^2}} \begin{bmatrix} c & d \\ d & e \end{bmatrix} \tag{25}$$

is a dense subgroup of $SU(2)$. To do this we will use the well known surjective homomorphism $\phi : SU(2) \rightarrow SO(3)$ whose kernel is $\{\pm \mathbb{1}\}$ (*cf.* [5], pg. 276). A general element of $SU(2)$ can be written as

$$\cos\left(\frac{\theta}{2}\right) \mathbb{1} + \sin\left(\frac{\theta}{2}\right) [x\sigma_x + y\sigma_y + z\sigma_z]$$

where σ_x, σ_y, σ_z are the Pauli matrices, and x, y, z are real numbers satisfying $x^2 + y^2 + z^2 = 1$. ϕ maps this element to the rotation by angle θ about the axis

$$\vec{x} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}.$$

Using equations 25 and 4, one finds that $\phi(A)$ and $\phi(B)$ are both rotations by $7\pi/5$. These rotations are about different axes which are separated by angle

$$\theta_{12} = \cos^{-1}\left(\frac{-4\sqrt{ab(ce-d^2)}(e-c)}{(a-b)((c-e)^2+2d^2)}\right) \simeq 0.739447844528\ldots$$

To show that $\rho_{**}^{(n)}(B_4)$ modulo phase is a dense subgroup of $SU(2)$ it suffices to show that $\phi(A)$ and $\phi(B)$ generate a dense subgroup of $SO(3)$. To do this we take advantage of the fact that the finite subgroups of $SO(3)$ are completely known.

Theorem 2 ([5] pg. 184) *Every finite subgroup of $SO(3)$ is one of the following:*

C_k: the cyclic group of order k

D_k: the dihedral group of order k

T: the tetrahedral group (order 12)

O: the octahedral group (order 24)

I: the icosahedral group (order 60)

If we can show that $\langle\phi(A), \phi(B)\rangle$ is not contained in any one of these groups then $\langle\phi(A), \phi(B)\rangle$ must be infinite and we are done.

Since $\phi(A)$ and $\phi(B)$ are rotations about different axes we know that $\langle\phi(A), \phi(B)\rangle$ is not C_k or D_k. Next, we note that $R = \phi(A)^5\phi(B)^5$ is a rotation by $2\theta_{12}$. By direct calculation, $2\theta_{12}$ is not an integer multiple of $2\pi/k$ for $k = 1, 2, 3, 4$, or 5. Thus R has order greater than 5. As mentioned on pg. 262 of [16], T, O, and I do not have any elements of order greater than 5. Thus, $\langle\phi(A), \phi(B)\rangle$ is not contained in C, O, or I, which completes the proof. Alternatively, using more arithmetic and less group theory, we can see that $2\theta_{12}$ is not any integer multiple of $2\pi/k$ for any $k \leq 30$, thus R cannot be in T, O, or I since its order does not divide the order of any of these groups. □

Next we'll consider $\rho_{**}^{(n)}$ for larger n. These will be matrices acting on the strings of length $n+1$. These can be divided into those which end in pp$*$ and those which end in $*$p$*$. The space upon which $\rho_{**}^{(n)}$ acts can correspondingly be divided into two subspaces which are the span of these two sets of strings. From equation 3 we can see that $\rho_{**}^{(n)}(\sigma_1)\ldots\rho_{**}^{(n)}(\sigma_{n-3})$ will leave these subspaces invariant. Thus if we order our basis to respect this grouping of strings, $\rho_{**}^{(n)}(\sigma_1)\ldots\rho_{**}^{(n)}(\sigma_{n-3})$ will appear block-diagonal with a block corresponding to each of these subspaces.

The possible prefixes of $*$p$*$ are all strings of length $n-2$ that start with $*$ and end with p. Now consider the strings acted upon by $\rho_{**}^{(n-2)}$. These have length $n-1$ and must end in $*$. The possible prefixes of this $*$ are all strings of length $n-2$ that begin with $*$ and end with p. Thus these are in one to one correspondence with the strings acted upon by $\rho_{**}^{(n)}$ that end in $*$p$*$. Furthermore, since the rules 3 depend only on the three symbols neighboring a given crossing, the block of $\rho_{**}^{(n)}(\sigma_1)\ldots\rho_{**}^{(n)}(\sigma_{n-3})$ corresponding to the $*$p$*$ subspace is exactly the same as $\rho_{**}^{(n-2)}(\sigma_1)\ldots\rho_{**}^{(n-2)}(\sigma_{n-3})$. By a similar argument, the block of $\rho_{**}^{(n)}(\sigma_1)\ldots\rho_{**}^{(n)}(\sigma_{n-3})$ corresponding to the pp$*$ is exactly the same as $\rho_{**}^{(n-1)}(\sigma_1)\ldots\rho_{**}^{(n-1)}(\sigma_{n-3})$.

For any $n > 3$, $\rho_{**}^{(n)}(\sigma_{n-2})$ will not leave these subspaces invariant. This is because the crossing σ_{n-2} spans the $(n-1)^{\text{th}}$ symbol. Thus if the $(n-2)^{\text{th}}$ and n^{th} symbols are p, then by equation 3, $\rho_{**}^{(n)}$ can flip

the value of the $(n-1)^{\text{th}}$ symbol. The n^{th} symbol is guaranteed to be p, since the $(n+1)^{\text{th}}$ symbol is the last one and is therefore $*$ by definition. For any $n > 3$, the space acted upon by $\rho_{**}^{(n)}(\sigma_{n-1})$ will include some strings in which the $(n-2)^{\text{th}}$ symbol is p.

As an example, for five strands:

$$\rho_{**}^{(5)}(\sigma_1) = \begin{bmatrix} b & 0 & 0 \\ 0 & a & 0 \\ 0 & 0 & a \end{bmatrix} \begin{matrix} *\text{p}*\text{pp}* \\ *\text{pppp}* \\ *\text{pp}*\text{p}* \end{matrix} \qquad \rho_{**}^{(5)}(\sigma_2) = \begin{bmatrix} c & d & 0 \\ d & e & 0 \\ 0 & 0 & a \end{bmatrix} \begin{matrix} *\text{p}*\text{pp}* \\ *\text{pppp}* \\ *\text{pp}*\text{p}* \end{matrix}$$

$$\rho_{**}^{(5)}(\sigma_3) = \begin{bmatrix} a & 0 & 0 \\ 0 & e & d \\ 0 & d & c \end{bmatrix} \begin{matrix} *\text{p}*\text{pp}* \\ *\text{pppp}* \\ *\text{pp}*\text{p}* \end{matrix} \qquad \rho_{**}^{(5)}(\sigma_4) = \begin{bmatrix} a & 0 & 0 \\ 0 & a & 0 \\ 0 & 0 & b \end{bmatrix} \begin{matrix} *\text{p}*\text{pp}* \\ *\text{pppp}* \\ *\text{pp}*\text{p}* \end{matrix}$$

We recognize the upper 2×2 blocks of $\rho_{**}^{(5)}(\sigma_1)$, and $\rho_{**}^{(5)}(\sigma_2)$ from equation 5. The lower 1×1 block matches $\rho_{**}^{(3)}(\sigma_1)$ and $\rho_{**}^{(3)}(\sigma_2)$, which are both easily calculated to be $[a]$. $\rho_{**}^{(5)}(\sigma_3)$ mixes these two subspaces.

We can now use the preceding observations about the recursive structure of $\{\rho_{**}^{(n)} | n = 4, 5, 6, 7 \ldots\}$ to show inductively that $\rho_{**}^{(n)}(B_n)$ forms a dense subgroup of $SU(f_{n-1})$ for all n. To perform the induction step we use the bridge lemma and decoupling lemma from [1].

Lemma 1 (Bridge Lemma) *Let $C = A \oplus B$ where A and B are vector spaces with $\dim B > \dim A \geq 1$. Let $W \in SU(C)$ be a linear transformation which mixes the subspaces A and B. Then the group generated by $SU(A)$, $SU(B)$, and W is dense in $SU(C)$.*

Lemma 2 (Decoupling Lemma) *Let G be an infinite discrete group, and let A and B be two vector spaces with $\dim(A) \neq \dim(B)$. Let $\rho_a : G \to SU(A)$ and $\rho_b : G \to SU(B)$ be homomorphisms such that $\rho_a(G)$ is dense in $SU(A)$ and $\rho_b(G)$ is dense in $SU(B)$. Then for any $U_a \in SU(A)$ there exist a series of G-elements α_n such that $\lim_{n\to\infty} \rho_a(\alpha_n) = U_a$ and $\lim_{n\to\infty} \rho_b(\alpha_n) = \mathbb{1}$. Similarly, for any $U_b \in SU(B)$, there exists a series $\beta_n \in G$ such that $\lim_{n\to\infty} \rho_a(\beta_n) = \mathbb{1}$ and $\lim_{n\to\infty} \rho_a(\beta_n) = U_b$.*

With these in hand we can prove the main proposition of this appendix.

Proposition 2 *For any $n \geq 3$, $\rho_{**}^{(4)}(B_4)$ modulo phase is a dense subgroup of $SU(f_{n-1})$.*

Proof: As mentioned previously, the proof will be inductive. The base cases are $n = 3$ and $n = 4$. As mentioned previously, $\rho_{**}^{(3)}(\sigma_1) = \rho_{**}^{(3)}(\sigma_2) = [a]$. Trivially, these generate a dense subgroup of (indeed, all of) $SU(1) = \{\mathbb{1}\}$ modulo phase. By proposition 1, $\rho_{**}^{(4)}(\sigma_1)$, and $\rho_{**}^{(4)}(\sigma_2)$ generate a dense subgroup of $SU(2)$ modulo phase. Now for induction assume that $\rho_{**}^{(n-1)}(B_{n-1})$ is a dense subgroup of $SU(f_{n-2})$ and $\rho_{**}^{(n-2)}(B_{n-2})$ is a dense subgroup of $SU(f_{n-3})$. As noted above, these correspond to the upper and lower blocks of $\rho_{**}^{(n)}(\sigma_1) \ldots \rho_{**}^{(n)}(\sigma_{n-2})$. Thus, by the decoupling lemma, $\rho_{**}^{(n)}(B_n)$ contains an element arbitrarily close to $U \oplus \mathbb{1}$ for any $U \in SU(f_{n-2})$ and an element arbitrarily close to $\mathbb{1} \oplus U$ for any $U \in SU(f_{n-3})$. Since, as observed above, $\rho_{**}^{(n)}(\sigma_{n-1})$ mixes these two subspaces, the bridge lemma shows that $\rho_{**}^{(n)}(B_n)$ is dense in $SU(f_{n-1})$. $\square$

From this, the density of $\rho_{*p}^{(n)}$ and $\rho_{p*}^{(n)}$ easily follow.

Corollary 1 *$\rho_{*p}^{(n)}(B_n)$ and $\rho_{p*}^{(n)}(B_n)$ are dense subgroups of $SU(f_n)$ modulo phase.*

Proof: It is not hard to see that

$$\begin{aligned} \rho_{*p}^{(n)}(\sigma_1) &= \rho_{**}^{(n+1)}(\sigma_1) \\ &\vdots \\ \rho_{*p}^{(n)}(\sigma_{n-1}) &= \rho_{**}^{(n+1)}(\sigma_{n-1}) \end{aligned} .$$

As we saw in the proof of proposition 2, $\rho_{**}^{(n+1)}(\sigma_n)$ is not necessary to obtain density in $SU(f_n)$, that is, $\langle \rho_{**}^{(n+1)}(\sigma_1), \ldots, \rho_{**}^{(n+1)}(\sigma_{n-1}) \rangle$ is a dense subgroup of $SU(f_n)$ modulo phase. Thus, the density of $\rho_{*p}^{(n)}$ in $SU(f_n)$ follows immediately from the proof of proposition 2. By symmetry, $\rho_{p*}^{(n)}(B_n)$ is isomorphic to $\rho_{*p}^{(n)}(B_n)$, thus this is a dense subgroup of $SU(f_n)$ modulo phase as well. □

C Zeckendorf Representation

Following [18], to construct the Fibonacci representation of the braid group, we use strings of p and $*$ symbols such that no two $*$ symbols are adjacent. There exists a bijection z between such strings and the integers, known as the Zeckendorf representation. Let P_n be the set of all such strings of length n. To construct the map $z : P_n \to \{0, 1, \ldots, f_{n+2}\}$ we think of $*$ as one and p as zero. Then, for a given string $s = s_n s_{n-1} \ldots s_1$ we associate the integer

$$z(s) = \sum_{i=1}^{n} s_i f_{i+1}, \tag{26}$$

where f_i is the i^{th} Fibonacci number: $f_1 = 1, f_2 = 1, f_3 = 2$, and so on. In this appendix we'll show the following.

Proposition 3 *For any n, the map $z : P_n \to \{0, \ldots, f_{n+2}\}$ defined by $z(s) = \sum_{i=1}^{n} s_i f_{i+1}$ is bijective.*

Proof: We'll inductively show that the following two statements are true for every $n \geq 2$.

$\mathbf{A_n}$: *z maps strings of length n starting with p bijectively to $\{0, \ldots, f_{n+1} - 1\}$.*

$\mathbf{B_n}$: *z maps strings of length n starting with $*$ bijectively to $\{f_{n+1}, \ldots, f_{n+2} - 1\}$.*

Together, A_n and B_n imply that z maps P_n bijectively to $\{0, \ldots, f_{n+2} - 1\}$. As a base case, we can look at $n = 2$.

$$\begin{aligned} pp &\leftrightarrow 0 \\ p* &\leftrightarrow 1 \\ *p &\leftrightarrow 2 \end{aligned}$$

Thus A_2 and B_2 are true. Now for the induction. Let $s_{n-1} \in P_{n-1}$. By equation 26,

$$z(ps_{n-1}) = z(s_{n-1}).$$

Since s_{n-1} follows a p symbol, it can be any element of P_{n-1}. By induction, z is bijective on P_{n-1}, thus A_n is true. Similarly, by equation 26

$$z(*s_{n-1}) = f_{n+1} + z(s_{n-1}).$$

Since s_{n-1} here follows a $*$, its allowed values are exactly those strings which start with p. By induction, A_{n-1} tells us that z maps these bijectively to $\{0, \ldots, f_n - 1\}$. Since $f_{n+1} + f_n = f_{n+2}$, this implies B_n is true. Together, A_n and B_n for all $n \geq 2$, along with the trivial $n = 1$ case, imply proposition 3. □

References

[1] Dorit Aharonov and Itai Arad. The BQP-hardness of approximating the Jones polynomial. 2006. arXiv:quant-ph/0605181.

[2] Dorit Aharonov, Itai Arad, Elad Eban, and Zeph Landau. Polynomial quantum algorithms for additive approximations of the Potts model and other points of the Tutte plane. *arXiv:quant-ph/0702008*, 2007.

[3] Dorit Aharonov, Vaughan Jones, and Zeph Landau. A polynomial quantum algorithm for approximating the Jones polynomial. *STOC 06*, 2006. arXiv:quant-ph/0511096.

[4] Andris Ambainis, Leonard Schulman, and Umesh Vazirani. Computing with highly mixed states. *Journal of the ACM*, 53(3):507–531, May 2006.

[5] Michael Artin. *Algebra*. Prentice Hall, 1991.

[6] Adriano Barenco, Artur Ekert, Kalle-Antti Suominen, and Päivi Törmä. Approximate quantum Fourier transform and decoherence. *Phyical Review A*, 54(1):139–146, 1996. arXiv:quant-ph/9601018.

[7] David A. Barrington. Bounded-width polynomial-size brancing programs recognize exactly those languages in NC^1. *Journal of Computer and System Sciences*, 38:150–164, 1989.

[8] Steven A. Cuccaro, Thomas G. Draper, Samuel A. Kutin, and David Petrie Moulton. A new quantum ripple-carry addition circuit. *arXiv:quant-ph/0410184*, 2004.

[9] Animesh Datta, Steven T. Flammia, and Carlton M. Caves. Entanglement and the power of one qubit. *Physical Review A*, 72(042316), 2005. arXiv:quant-ph/0505213.

[10] Thomas Draper. Addition on a quantum computer. *arXiv:quant-ph/0008033*, 2000.

[11] Michael Freedman, Alexei Kitaev, and Zhenhan Wang. Simulation of topological field theories by quantum computers. *Communications in Mathematical Physics*, 227:587–603, 2002.

[12] Michael Freedman, Michael Larsen, and Zhenghan Wang. A modular functor which is universal for quantum computation. 2000. arXiv:quant-ph/0001108.

[13] W. Haken. Theorie der normalflachen, ein isotopiekriterium für den kreisknoten. *Acta Mathematica*, (105):245–375, 1961.

[14] Joel Hass, Jeffrey Lagarias, and Nicholas Pippenger. The computational complexity of knot and link problems. *Journal of the ACM*, 46(2):185–211, 1999. arXiv:math.GT/9807016.

[15] F. Jaeger, D. L. Vertigan, and D. J. A. Welsh. On the computational complexity of the Jones and Tutte polynomials. *Mathematical Proceedings of the Cambridge Philosophical Society*, (108):35–53, 1990.

[16] Vaughan F. R. Jones. Braid groups, Hecke algebras and type II_1 factors. In *Geometric methods in operator algebras, US-Japan Seminar*, pages 242–273, Kyoto, July 1983.

[17] Vaughan F. R. Jones. A polynomial invariant for knots via von Neumann algebras. *Bulletin of the American Mathematical Society*, 12:103–111, 1985.

[18] Louis H. Kauffman and Samuel J. Lomonaco Jr. q-deformed spin networks knot polynomials and anyonic topological quantum computation. *arXiv:quant-ph/0606114*, 2006.

[19] Alexei Yu. Kitaev. Quantum computations: algorithms and error correction. *Russian Mathematical Surveys*, 52(6):1191–1249, 1997.

[20] E. Knill and R. Laflamme. Power of one bit of quantum information. *Physical Review Letters*, 81(25):5672–5675, 1998. arXiv:quant-ph/9802037.

[21] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[22] Christos Papadimitriou. *Computational Complexity*. Addison Wesley, 1994.

[23] L. J. Schulman and U. Vazirani. Molecular scale heat engines and scalable quantum computation. *STOC 99*, pages 322–329, 1999.

[24] Dan Shepherd. Computation with unitaries and one pure qubit. 2006. arXiv:quant-ph/0608132.

[25] Ingo Wegener. *The Complexity of Boolean Functions*. Wiley, 1987.

[26] Edward Witten. Quantum field theory and the Jones polynomial. *Communications in Mathematical Physics*, 121(3):351–399, 1989.

[27] Pawel Wocjan and Jon Yard. The Jones polynomial: quantum algorithms and applications in quantum complexity theory. *arXiv:quant-ph/0603069*, 2006.