DQC1 complexity class and Jones polynomials

Ohad Barta

April 17, 2015

Table of contents I

Table of contents II

DQC1 class definition

DQC1 class is the class of decidable languages with algorithm A such that:

- A starts with one clean qubit in state $|0\rangle$, and n qubits in the maximally mixed state
- A may perform any unitary operation
- A can only perform a measurement of the clean qubit at the end of the algorithm
- A has access to a classical computer, so $P \subset DQC1$
- A can't be invoked many times in parallel
- A runs in polynomial time
- $\forall x$, A decides if $x \in L$ correctly with probability of at least $\frac{2}{3}$

Complete Problems definition

Reminder: in general, a language L is said to be "complete" in the class DQC1, if:

- $L \in DQC1$
- $\forall L_0 \in DQC1$ there is a reduction from L_0 to L, such that the reduction algorithm is in DQC1

In the next few slides, we show that calculating an estimate of the trace of a unitary matrix is DQC1-complete

trace estimation algorithm (1)

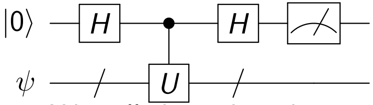
The start state of any DQC1 problem is one clean qubit (state $|0\rangle$), and n-qubits in the maximally mixed state. That is, the start state is

$$\rho = |0\rangle\langle 0| \otimes \frac{I}{2^n} \tag{1}$$

We can use the hadamard test, which gets as input this state, in order to acurratly estimate a trace of unitary operation U.

The Hadamard Test

The description of the Hadamard test for some unitary matrix U is:



We will show that this circuit indeed calculates the trace of U

- after the first hadamard gate, the state is $|+\rangle\,\psi = \frac{1}{\sqrt{2}}|0\rangle|\psi\rangle + \frac{1}{\sqrt{2}}|1\rangle|\psi\rangle$
- after the C-U operation, the new state is $\frac{1}{\sqrt{2}}|0\rangle|\psi\rangle + \frac{1}{\sqrt{2}}|1\rangle U|\psi\rangle$
- after the final hadamard operation, the final state becomes $\frac{1}{2}|0\rangle|\psi\rangle + \frac{1}{2}|1\rangle|\psi\rangle + \frac{1}{2}|0\rangle U|\psi\rangle - \frac{1}{2}|1\rangle U|\psi\rangle = \frac{|\psi\rangle+U|\psi\rangle}{2}|0\rangle + \frac{|\psi\rangle-U|\psi\rangle}{2}|1\rangle$

The Hadamard Test (2)

Therefore, the propability for measure 0 in the end is:
$\rho_0 = (\frac{\langle\psi| + \langle\psi| U^\dagger}{2})(\frac{|\psi\rangle + U|\psi\rangle}{2}) = \frac{1}{4}(\langle\psi| |\psi\rangle + \langle\psi| U^\dagger |\psi\rangle + \langle\psi| U |\psi\rangle + \langle\psi| U^\dagger U |\psi\rangle) = \frac{1}{2} + \frac{1}{4}(\langle\psi| U^\dagger |\psi\rangle + \langle\psi| U |\psi\rangle) = \frac{1}{2} + \frac{1}{2} Re(\langle\psi| U |\psi\rangle)$
When we remember that in our case ψ is actually the completely mixed state, we get that the probability is:

$$\sum_{\psi \in 0,1^n} \frac{1 + Re(\langle\psi| U |\psi\rangle)}{2} = \frac{1}{2} + \frac{Re(TrU)}{2^{n+1}}$$

Therefore, the problem of trace estimation can be solved with one clean quibit.

The hardness of trace estimation

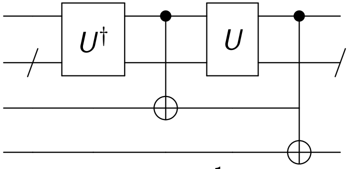
Next, we will want to show that trace estimation is hard in DQC1. Suppose we have some language $L \in DQC1$, and an some x, and we want to decide if $x \in L$.
$L \in DQC1$, therefore its start state obeys equation 1. on this start state, we imply some unitary matrix U, and get the state $\rho_{final} = U\rho U^{\dagger} = U|0\rangle\langle 0| \frac{I}{2^n} U^{\dagger}$ Therefore, the propability to measure 0 equals to the trace of the final matrix, when we enforce the first bit to be zero, or:

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

Unfortunatly - this matrix isn't unitary!!

The hardness of trace estimation (2)

To resove that issue, we exmine the following quantom circuit C:



Proposition 1.1: $\frac{1}{4}$tr[C]$= Tr[(|0\rangle\langle 0| \otimes I)U(|0\rangle\langle 0| \otimes IU^\dagger)]$

Proof:

- first, lets remember that $tr[C] = \sum_{\psi \in 0,1^n} \langle\psi| C |\psi\rangle$ and in a similar way, $Tr[(|0\rangle\langle 0| \otimes I)U(|0\rangle\langle 0| \otimes IU^\dagger)] = \sum_{\psi \in 0,1^n} \langle\psi| (|0\rangle\langle 0| \otimes I)U(|0\rangle\langle 0| \otimes IU^\dagger) |\psi\rangle$

The hardness of trace estimation (3)

- that is, some state ψ contribute to the trace of D iff after imply $U^\dagger$ it has some component with zero in its first qubit, and then after implying U on that component, we still remain with some non-zero component in the first qubit
- Similarly, in C, if, say, the result of implying $U^\dagger$ on ψ give a non-zero component, the one of the two last qubits will "flip", creating a state orthogonal to the original state (and similarly on U).
- therefore, the two circuits traces has the exact same components and are equal, up to factor of 4 (which come from the "free choice" in the values of the two last qbits in C)

adding few more clean bits dont give extra power

We showed that computing the matrix trace is DQC1-complete problem. We notice now that actually we didnt computed it accuratly. Since only the expectation off the algorithm is the trace, we rather get an approximation. According to the Chernoff inequality, the approximation will be exponentially good with polynomial number of repretitions. However here the matrix is exponentially large, so we get an approximation of $\frac{1}{poly(n)}$ to the expression $\frac{Tr(U)}{2^{n+1}}$, or in other words we have a $\frac{2^n}{poly(n)}$ additive-approximation to the trace.

adding few more clean bits dont give extra power (2)

We will define now the DQCK complexity class: DQCK class is the class of decidable languages with algorithm A such that:

- A runs in polynomial time
- $\forall x$, A decide correctly if $x \in L$, with probabilty of at least $\frac{2}{3}$. In case that $x \in L$, we expect all the clean bits to be 0 at the end of the algorithm
- A start state includes K clean qubits (K can be a costant, or a function of n) (state $|0\rangle$), and n-qubits in the maximally mixed state
- A may perform any unitary operation on his start state
- A can measure only the clean qubits, only at the end of the algorithm
- We assume that A is invoked cant be invoked many times in paralel
- A may have access to classical computer, so $P \in DQCK$

adding few more clean bits dont give extra power (3)

We will now prove that for $k \leq \log n$, estimate the trace of unitary matrix with the same precision is still a complete problem, thus proving that adding logarithmic number of clean bits doesn't change the power.
Obviously we can calculate the trace of unitary matirx with $\log n$ bits, since we can do it just with one.

adding few more clean bits dont give extra power (3)

As for the less trivial direction, assume we have some quantum algorithm in DQCK. Similarly to the one-qubit option, final state is: $\rho_{final} = U\rho U^{\dagger} = U|0\rangle\langle 0|^{\otimes k}\frac{I}{2^n}U^{\dagger}$ and therefore the probability of measuring 0 at the end is:
$p_0 = Tr[|0\rangle\langle 0|^{\otimes k} \otimes I\rho_{final}] = 2^{-n}Tr[(|0\rangle\langle 0| \otimes I)U(|0\rangle\langle 0|^{\otimes k} \otimes IU^{\dagger})$
Now, we have the same problem at estimatimg this matrix: its not unitary! To resolve that issue, we build circuit similar to the one in the 1-clean quibit process, but now we add 2k ancila qubits, when there is a CNOT gate between each i-th qubit and (i+k)-th qubit.

adding few more clean bits dont give extra power (4)

Now, we can see (similary to the proposition 1.1), that the trace of the new circuit U^* follows the rule: $Tr[U^*] = 2^k Tr[U]$. Thus, in polynomial number of executions we can compute its trace up to a percision of $\frac{2^{n+k}}{poly(n,k}$, but this equals to $\frac{2^n}{poly(n}$ when $k \leq \log n$, which means that in this case the precision is good enough to decide the original problem.

Knots

A knot is one strand, in R^3. The knot is invariant to moving or stretching, but we cant cut the knot. An old topology question, is to know if two knots are actually the same knot.Reidemeister showed in 1927 the "Reidemeister moves", which doesn't change the knot.

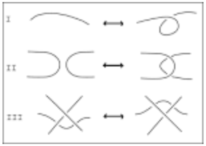


Figure: Reidemeister moves

Braid groups

Definition

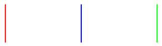
Consider two horizontal bars, one on top of the other, with n-points each. A n-stand braid, is n strands, such that:

- each strand has exactly one peg attached to it on each bar
- the strands may cross one over another
- at every point, the strand direction has non-zero component directed down

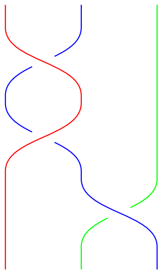
We call to the collection of all such braids the braid group B_n, with identity element and generators that will immediately follow, when the operation between two brands is put them one below the other, and connect all the button-pegs of the first, with the top-pegs of the second.

examples

The identity braid:

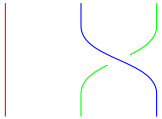


Some other braid:



braid group generators

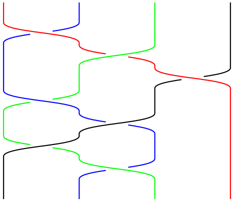
$\forall 1 \leq i \leq n$, we denote by σ_i, the braid which takes the i-th strand to the (i+1) place, the (i+1)-strand to the i-th place, and leaves all the other strands in place. For example, this is σ_2 with 3-strands:



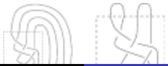
Notice that when $1 < |i-j|$ then σ_i, σ_j act on completely different strands, so they are commutative: $\sigma_i \sigma_j = \sigma_j \sigma_i$
Furthermore, it holds that $\sigma_i \sigma_{i+1} \sigma_i = \sigma_{i+1} \sigma_i \sigma_{i+1}$ (both switch the i-th strand and the (i+2)-strand, while leave all the others in place)

braid groups and knots

Braids can create a knot by "joining the loose ends" all together. there are two ways of doing it: The plat closure, where we join neighbour pegs in the top and bottom, and the trace closure, where we connect each top peg to the corresponding bottom peg, without creating more loops. for example, for the following braid:



has this closures:

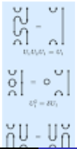


Tempely Lieb algebra

In a similar way to the braid groups, we can define the Tempely Lieb n,d algebra. Tempely Lieb n,d algebra group consists of two rows of pegs (button and top), and n-strabds, such that:

- each strand connects to exactly two pegs (but they can be from the same side!)
- the strands cannot intersect between them.
- the operator of two objects like this is simply put them one below the other, while erasing circuits created, but multiply the final result by d, for each circuit removed

an example for this operator is:



Tempely Lieb algebra generators

In a similar way to the braid groups, we can define generators to the this algebra, such that the i-th generator send the ith-strand on the buttom and top the the i+1 peg, and leaves all the others in place.

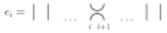


Figure: tempely lieb generators

Notice that these generators obeys the following rules (as demonstrated in the previuos slide figure):

- $E_iE_j = E_jE_i$, when $2 \leq |i - j$
- $E_iE_{i+1}E_i = E_i$, $E_iE_{i-1}E_i = E_i$
- ${E_i}^2 = dE_i$

From braid groups to tempely-lieb algebra

Now, we observe that a connection between the braid groups and the tempely-lieb algebra can be made by defining homomorphism to the braid groups generators. we will define: $\rho_A(\sigma_i) = AE_i + A^{-1}I$, when I is the identity in the lieb-algebra, and A is a number that satisfies $A^2 + A^{-2} = d$. We can show that this is indeed a representation of the braid groups, if we show that the relations of the braid group generators still hold. As for the first relation, it holds that for $2 \leq |i - j|$, $\rho_A(\sigma_i), \rho_A(\sigma_j)$ commutes since in this case E_i, E_j commutes as well.

From braid groups to tempely-lieb algebra (2)

For the second relation, to show that :

$$\rho_A(\sigma_i)\rho_A(\sigma_{i+1})\rho_A(\sigma_i) = \rho_A(\sigma_{i+1})\rho_A(\sigma_i)\rho_A(\sigma_{i+1})$$

or:
$A^3E_iE_{i+1}E_i + AE_iE_{i+1} + AE_i^2 + A^{-1}E_i + AE_{i+1}E_i + A^{-1}E_{i+1} + A^{-1}E_i + A^{-3}$
$=$
$A^3E_{i+1}E_iE_{i+1} + AE_{i+1}E_i + AE_{i+1}^2 + A^{-1}E_{i+1} + AE_iE_{i+1} + A^{-1}E_i + A^{-1}E_{i+1} + A^{-3}$ after delete similar elements and use some generators identities, we have to show that:

$$(A^{-1} + Ad + A^3)E_i = (A^{-1} + Ad + A^3)E_{i+1}$$

and this is correct since $(A^{-1} + Ad + A^3) = 0$

The Markov Trace

The Markov Trace is a function on a tempely-Lieb algebra, which defined as follows:

- given a tempely-lieb algebra object, we connect its buttom and top bars, in a similar way to a trace closure.
- when denote the number of loops created like this with a, the trace closure is d^{a-n}

The Markov trace Tr obeys the following:

- $\mathrm{Tr}[1] = 1$ (the identity tempely-algebra has n loops in its clousre, $d^{n-n} = 1$)
- $\forall X, Y \in TL[n, d]$, $\mathrm{Tr}[\mathrm{XY}] = \mathrm{Tr}[\mathrm{YX}]$
- $\forall X \in TL[n-1, d]$, $Tr[xE_{n-1}] = \frac{Tr[x]}{d}$ (add E_{n-1} add new peg but dont enlarge the number of loops).

The uniqueness of the Markov Trace

We will prove that the markov trace is the only function on tempely-lieb objects that obeys rules 1-3, By prove that rules 1-3 are enough to determine the trace value of any tempely-lieb object. We will regard each object as a word with letter from the the group $E_1 ... E_n$ We will define such word to be "reduced", if it wont be equal to any "shorter" word. The proof is done in induction on n, the maximal generator index in the reducible word:

- As for the base case, there is only one object with maximal generator 0 - the identity object, and its value is set to 1.
- assume that the value is well defined for any reducible object maximal generator $\leq n$ and lets prove it for n+1.

The uniqueness of the Markov Trace (2)

First, we will prove that in any such reducible word contains exactly one E_n generator. This will suffice, since then we can write $w = w_1 E_n w_2$, when w_1, w_2 are reducible words with maximal generator n-1, which thier value is well defined by the induction assumption.
Then, it holds that
$Tr[w] = Tr[w_1 E_n w_2] = Tr[w_1 w_2 E_n] = dTr[w_1 w_2]$, and since $w_1 w_2$ is another reducible word that its value is well defined, the proof will be complete.

The uniqueness of the Markov Trace (3)

Assume the contrary, that is there is a reducible word that contains two E_n. Then, we can write $w = w_1 E_n w_2 E_n w_3$, when w_i are reducible with maximal generator E_{n-1}. Now, if E_{n-1} isnt in w_2, we can change the order (from generators rules:) $w = w_1 w_2 E_n E_n w_3 \ = d w_1 w_2 E_n w_3$,so w isn't reducible. If there is E_{n-1} in w_2, we can write:

$$w = w_1 E_n v_1 E_{n-1} v_2 E_n w_3,$$

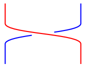
when v_i are reducible with maximal generator (E_{n-2}. Therefore, they can commute with E_{n-1}, creating

$$w = w_1 v_1 E_n E_{n-1} E_n v_2 w_3 = w_1 v_1 E_n v_2 w_3$$

, so w again is not reducible

Introduction

After half a century of searching knots invariants in a geometric way, in 1984 the Jones polynomial was discovered. It matches each knot a polynom , that stays invariant under the Reidemeister moves. We will first define the Kaufmann Bracket Polynomial, which is "almost" correct.
Consider a knot K. for each crossing in K, from the form



We decide at random to replace it with E_1, or with the tempely-lieb identity element. Each decision like this for all the crossings is a state σ

The Kaufmann Bracket Polynomial - definition

We denote by σ_+ the number of replaces we made with E_1, and by σ_- the number of replaces we made with the identity element. We denote byN_σ the number of loops created when all the changes of σ are applied. Then, the Kaufmann Bracket Polynomial is defined as: $L(A) = \sum_{all_s tates_\sigma} A^{\sigma_+} A^{-\sigma_-} d^{N_\sigma - 1}$

d is defined as: $d = -A^{-2} - A^2$

The Kaufmann Bracket Polynomial - properties

Note that the Kaufmann Bracket Polynomial holds the following rules:

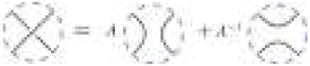
- $\forall A$, L(A)=1 in the unknot (one state with $\sigma_+ = 0, \sigma_- = 0, N_\sigma = 1$)
- 

  Figure: kaufmann bracket recursive nature

  This comes directly from the definition, if we consider only one cross, after choose a state sigma for all the others, or we choose to replace it with E_1 and multiply by A, or we choose to replace it by the identity and multiply by A^{-1}
- we can eliminate an isolated unknot and multiply the result with a factor of d. (examples will follow).

definition

Jones polynomial is different from Kaufmann bracket polynimial only with some normalization factor. We define by the w(k) for a knot k to be: $w(k) = \sum_{allcrossings} (-1)^{istheleftarrowabovetherightone}$ and Jones polynomial is defines as:
$V_k(t) = V_k(A^{-4}) = (-A)^{3w(k)} L_k(A)$.
Notice that w(k) for the unknot is 0, so the Jones polynomial of the unknot is still equals to 1 at any point.

example

Lets see an example to the calculation of Kaufmann and jons polynomial. Consider the Hopf link. The Kaufmann polynomial of this link is:

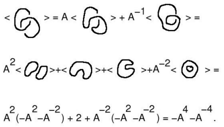

$< \ > = A < \ > + A^{-1} < \ > =$
$A^2 < \ > + < \ > + < \ > + A^{-2} < \ > =$
$A^2(-A^2-A^{-2}) + 2 + A^{-2}(-A^2-A^{-2}) = -A^4-A^{-4}.$

Figure: the kaufmann polynomial of the Hopf link

Moving to Jones Polynomial, we can see that w(HopfLink)=-2 (two cross with the same oreintation), so: $V_{hopfLink} = (-A)^{-6}(A^{-4}+A^4) = A^{-2}+A^{-10})$. Remember that $t = A^{-4}$ and we get $V_{hopfLink}(t) = \sqrt{t}(1+t^2)$

the connection to Reidiemster moves

For Completion, we will note that it is easy to see that Jones Polynomial remains the same under Reidiemster moves. We will show here only one of the three, the others can be proved in a similar technique:

the connection to markov trace and braid groups

We will now show a connection between the Jones Polynomial and the Markov trace. Proposition: $V_{B^{tr}}(A) = (-A)^{3w(B^{tr})} d^{n-1} Tr[\rho_A(B)]$ That is, the Jones polynomial of a trace-closure of some braid is connected to the Markov trace value of its corresponding templely-lieb object. Proof: We only have to proof that $L(B^{tr}) = d^{n-1} Tr[\rho_A(B)]$. Recall that for each crossing in the braid B, the kaufmann polynomial hold

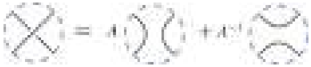


Figure: kaufmann bracket recursive nature

. The homomorphism also has the form of $\rho_A(\sigma_i) = AE_i + A^{-1}I$ - exactly the same!.

the connection to markov trace and braid groups(2)

After choosing a state σ, its value in the Kaufmann polynimial is $d^{N_\sigma - 1)}$, and its value i the Markov trace is $d^{N_\sigma} - n$), therefore we need the d^{n-1} factor.

The fibonacci representation of a braid group

We would like to obtain some matrix representation of a braid. Given an n-strand braid, we can write a string of n+1 elements on the buttom of the braid between every two strands, where each element is either * or p. The only restriction is that there will be no two adjacent * elements. The number of possibilities to do so is of course f_{n+3}, where f_n denotes the n-th fibonacci number.
Next, for each crossing and labelling of it (the 3 elements from the right, left, and in the crossing) , we would like to give a linear function that will "open up" the crossing, and that may change the center label. (we wouldn't want to change the right or left label, in order to preserve the string of elements correctness under all the operations).

The fibonacci representation of a braid group (2)

That is, in the most general form, a cross σ_i with labeling of P*P, can move to a times * Identity with labelling P*P + b times the identity with labelling PPP. we will denote it by
$(p\hat{*}p) = a(p * p) + b(ppp)$
Such matrix representation must have some properties that will make it usefull to us:

- for every braid, its matrix must be unitary, so we can link it to some quantom circuit.
- for every braid, there has to be some connection between the Jones Polynomial of the braid, and the relevant matrix.

The fibonacci representation of a braid group (3)

The explicit representation is: $(*\hat{p}p) = a(*pp)$
$(*\hat{p}*) = b(*p*)$
$(p\hat{*}p) = c(p*p) + d(ppp)$
$(p\hat{p}*) = a(pp*)$
$(p\hat{p}p) = d(p*p) + e(ppp)$
, with: $a = -A^4$
$b = A^8$
$c = A^8\tau^2 - A^4\tau$
$d = A^8\tau^{\frac{3}{2}} + A^4\tau^{\frac{3}{2}}$
$e = A^8\tau - A^4\tau^2$
$A = e^{\frac{-3\pi i}{5}}$
$\tau = \frac{2}{1+\sqrt{5}}$

The Fibonacci representation - example

We will show the Fibonacci representation of σ_1 with 3 strands:

$$\begin{pmatrix} b & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & a & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & a & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & e & 0 & d & 0 & 0 \\ 0 & 0 & 0 & 0 & a & 0 & 0 & 0 \\ 0 & 0 & 0 & d & 0 & e & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & e & d \\ 0 & 0 & 0 & 0 & 0 & 0 & d & c \end{pmatrix} \begin{pmatrix} *p*p \\ *ppp \\ *pp* \\ pppp \\ pp*p \\ pp*p \\ p*pp \\ ppp* \\ p*p* \end{pmatrix}$$

The Unitary of the Fibonacci representation

Its enough to prove that all the braid group generators, since multiplication of unitary matrices is again unitary matrix. Each generator like this include only one cross. As we saw in the definition and examples, each cross leaves only 1-2 non-zero elements in each matrix row, in the corresponding labelling entry. we will move onto the 5 options to label the 3 elements near the cross (since the other elements doesnt matter here).

- with all the labelling with the form ...*pp... , the unitary row will include only one "a" in the matrix diagonal, and no other row have non-zero entries in that column. That is, with multiplied by its dagger, the entry on the diagonal will be $a^{\dagger}a = e^{\frac{-12pii}{5}} e^{\frac{12pii}{5}} = 1$, and all the other entries will be zero.
- the same reasoning apply for labeling with the form ...pp*...

The Unitary of the Fibonacci representation (2)

- with all the labelling with the form ...*p*... , the unitary row will include only one "b" in the matrix diagonal, and no other row have non-zero entries in that column. That is, with multiplied by its dagger, the entry on the diagonal will be $b^{\dagger}b = e^{\frac{-24pii}{5}}e^{\frac{24pii}{5}} = 1$, and all the other entries will be zero.
- with all the labelling with the form ...p*p.., the unitary row will include one "d" in the ...ppp.. entry, and one "c" in the ...p*p.. entry. that is, when multiplied by its dagger, the diagonal element will be $c^{\dagger}c + d^{\dagger}d =$ $(A^{-8}\tau^2 - A^{-4}\tau)(A^8\tau^2 - A^4\tau) + (A^{-8}\tau^{\frac{3}{2}} + A^{-4}\tau^{\frac{3}{2}})(A^8\tau^{\frac{3}{2}} + A^4\tau^{\frac{3}{2}})$ $= \tau^4 - A^{-4}\tau^3 - A^4\tau^3 + \tau^2 + \tau^3 + A^{-4}\tau^3 + A^4\tau^3 + \tau^3$ $= \tau^4 + 2\tau^3 + \tau^2 = \tau^2(\tau+1)^2 = 1^2 = 1$

The Unitary of the Fibonacci representation (3)

- now, lets check another entries in this row, after multiply this matrix by its dagger. All the other rows has zero entries in the relevant columns, except the ...ppp.. row. when multiply, we will get $e^\dagger d + d^\dagger c =$ $(A^{-8}\tau - A^{-4}\tau^2)(A^8\tau^{\frac{3}{2}} + A^4\tau^{\frac{3}{2}}) + (A^{-8}\tau^{\frac{3}{2}} + A^{-4}\tau^{\frac{3}{2}})(A^8\tau^2 - A^4\tau)$ $= \tau^{2.5} + A^{-4}\tau^{2.5} - A^4\tau^{3.5} - \tau^{3.5} + \tau^{3.5} - A^{-4}\tau^{2.5} + A^4\tau^{3.5} - \tau^{2.5}$ $= 0$
- similar calculation will yield the result on the ...ppp... rows

Fibonacci representation and Jones polynimial

We will want to connect somehow between the Fibonacci representation and the Jones Polynomial. For this, we will use the Tempely Lieb algebra, and the Markov Trace. We saw that the Markov trace is a uniquely defined function over the Tempely-Lieb algebra, such that it is strongly connected to the Jones Polynomial. If we will be able to define a function over the Fibonacci representation that "behaves the same", we will be able to show such correlation between Jones Polynomial and the Fibonacci representation. We will donate this function as $\tilde{Tr}$

Fibonacci representation and Jones polynimial (2)

The requirements from $\tilde{Tr}$ are:

- $\tilde{Tr}(1) = 1$
- $\tilde{Tr}(XY) = \tilde{Tr}(YX)$
- $\tilde{Tr}[xE_{n-1}] = \frac{\tilde{(}Tr)[x]}{d}$

The last requirement, however, regards to a specific Templerly-Lieb element. Thus, we have to show that the Fibonacci representation and the Temperly-Lieb algebra "live in the same world", in order to translate E_{n-1} to some matrix in the Fibonacci representation, and prove the requirement on the matrix after the translation. Therefore, in order to even start talking on $\tilde{Tr}$, we have to show a representation of the Temperly-Lieb algebra inside the Fibonacci representation.

Fibonacci representation and Jones polynimial (3)

The requirements from the representation, are, as always, to preserve the original generators properties. Here, the representation is: $\rho_b(E_i) = A^{-1}\rho_F(\sigma_i) - A^{-2}1$, with 1 symbols the identity matrix.
it should hold that:

- $\rho_b(E_i)\rho_b(E_j) = \rho_b(E_j)\rho_b(E_i), when2/leq|i-j|$.This hold, since the Fibonacci representation is a representation, so $rho_F(\sigma_i), rho_F(\sigma_j)$ commutes, and therefore $\rho_b(E_i), \rho_b(E_j)$ commutes.
- $\rho_b(E_i{}^2 = d\rho_b(E_i$: in a similar way to what we did in the unitary proof, we can divide each $rho_F(\sigma_i$ to blocks of

$$(a)\ (b)\begin{pmatrix} c & d \\ d & e \end{pmatrix}$$

, and prove that this relation hold on each block separately.

Fibonacci representation and Jones polynimial (4)

The last requirement we should prove is that $\rho_b(E_i)\rho_b(E_{i+1})\rho_b(E_i) = \rho_b(E_i))$. Direct calculation shows that $rho_b(E_1), (rho_b(E_2)$ with 3 strands stisfy this requirment. This suffice, since the relation between $rho_b(E_i), rho_b(E_{i+1})$ remains the same when i is change (up to re-index of the matrix, the matrix doesn't change), or when the number of strands get bigger with the same index (we will have more "squares" of the form mentioned in the previous slide for more equivalent labellings).

Fibonacci representation and Jones polynimial (5)

Its time to consider the specific function $\tilde{Tr}$, and prove its desired properties. $\tilde{Tr} = \frac{1}{\phi f_n + f_{n-1}} \sum\limits_{s \in Q_{n+1}} W_s \rho_f(b)_{s,s}$, when $\rho_f(b)_{s,s}$ denote the s-th diagonal entry in the Fibonacci representation of b, and W_s is ϕif s ends with p, and 1 if s ends with *. Q_{n+1} is the set of all strings with length n+1 that obeys the "no two adjective *" rule.

Its easy to see that $\tilde{Tr}(1) = 1$. In the identity matrix, all the diagonal entries equals to one, there are f_n entries that end with p, and f_{n-1}entries that end with *, so

$\tilde{Tr} = \frac{1}{\phi f_n + f_{n-1}} \sum\limits_{s \in Q_{n+1}} W_s \rho_f(b)_{s,s} = \frac{1}{\phi f_n + f_{n-1}} (\phi f_n + f_{n-1}) = 1$

Fibonacci representation and Jones polynimial (6)

Its easy to see that $\tilde{Tr}(XY) = \tilde{Tr}(YX)$, since at theend we talking about traces of matrices, and traceis a commutative property.

We remain with the last requirement of $\tilde{Tr}[xE_{n-1}] = \frac{\tilde{(Tr)}[x]}{d}$. First, we will instantiate E_{n-1} according to its representation:
$E_{n-1} = A^{-1}\rho_F(\sigma_{n-1}) - A^{-2}1.$ Therefore,
$\tilde{Tr}[xE_{n-1}] = \tilde{Tr}[A^{-1}x\rho_F(\sigma_{n-1}) - A^{-2}x]$
Therefore, from the linearity of the trace we get
$\tilde{Tr}[xE_{n-1}] = A\tilde{Tr}[\rho_f({\rho_f}^{-1}(x) * \sigma_{n-1})] - A^{-2}\tilde{Tr}[x]$
Therefore, it remain to check the value of $\tilde{Tr}[\rho_f({\rho_f}^{-1}(x) * \sigma_{n-1})]$
A careful examination of all the different labelling options of the new strand, combined with the Fibonacci representation rules for the new crossing, yield that this requirement indeed hold

The numbers representation

We saw that the Fibonacci representation connects between braids and elements of p and *, such that the number of possible p,* sequences is f_{n+2}, when f_i is the i-th Fibinacci number.
If we where representing numbers in the "regular" form (binary basis - 1 for * and 0 for p, for example), then it would require n+1 qubits.
However, the ratio $\frac{f_i}{2^i}$ becomes exponentially small (since $\frac{f_{i+1}}{f_i} < 2$), and therefore computing the trace of the Fibonacci matrix over exponentially small number of elements, might not be in DQC1.
To resolve this issue, we will represent the elements $0 \ldots f_{n+2}$ as $z(s) = \sum_{i \in 1 \ldots n} s_i f_{i+1}$. It can be proven (by Induction), that this method indeed represents the numbers in the range $0 \ldots f_{n+2}$.