Title: Efficient quantum algorithms for an additive approximation of the Tutte polynomial and the q-state Potts model.

Date: May 21, 2007 04:00 PM

URL: http://pirsa.org/07050024

Abstract: I will present an efficient quantum algorithm for an additive approximation of the famous Tutte polynomial of any planar graph at any point. The Tutte polynomial captures an extremely wide range of interesting combinatorial properties of graphs, including the partition function of the q-state Potts model. This provides a new class of quantum complete problems.

Our methods generalize the recent AJL algorithm for the approximation of the Jones polynomial; instead of using unitary representations, we allow non-unitarity, which seems counter intuitive in the quantum world. Significant contribution of this is a proof that non-unitary operators can be used for universal quantum computation.

An efficient quantum algorithm for approximating the Tutte polynomial and the q-state Potts model

Dorit Aharonov[1], Itai Arad[1], Elad Eban[1] and Zeph Landau[2]

[1]School of Computer Science and Engineering,
The Hebrew University,
Jerusalem, Israel

[2]Department of Mathematics,
City College of New York, New York, USA

May, 2007

Previous quantum algorithms

"Shor type algorithms"

- Factoring (Shor, 1994)
- Group Order (Watrous, 2001)
- Legendre Symbol (van Dam, Hallgren, 2000)
- Pellś Equation (Hallgren, 2002)
- Dihedral HSP (Kuperberg, 2003)
- HSP in various groups
- Hidden quadratic structures (Childs, Schulman, Vazirani, 2007)

Random-walk based algorithms

Quantum random walks (Childs, Cleve, Deotto, Farhi, Gutmann, Spielman, 2002). Provides an exponential speed-up over the classical case

Previous quantum algorithms

"Shor type algorithms"

- Factoring (Shor, 1994)
- Group Order (Watrous, 2001)
- Legendre Symbol (van Dam, Hallgren, 2000)
- Pellś Equation (Hallgren, 2002)
- Dihedral HSP (Kuperberg, 2003)
- HSP in various groups
- Hidden quadratic structures (Childs, Schulman, Vazirani, 2007)

Random-walk based algorithms

Quantum random walks (Childs, Cleve, Deotto, Farhi, Gutmann, Spielman, 2002). Provides an exponential speed-up over the classical case

The AJL algorithm - approximating the Jones polynomial

The Jones polynomial

A knot-invariant polynomial, discovered by Jones in the mid 80's. Appears in topological field theory, statistical physics, biology, and many more

The AJL algorithm

An efficient quantum algorithm to approximate the Jones polynomial of a knot by Aharonov, Jones & Landau (2005). Based on work of Kitaev, Freedman, Larsen & Wang on Topological Quantum Computation (2002).

- Not based on quantum Fourier transform;
- Uses a mapping of the combinatorial problem into an algebra, and then uses a unitary representation of the algebra that can be implemented on a quantum computer.
- Provides an additive approximation of the Jones Polynomial.
- Is quantum-complete.

The AJL algorithm - approximating the Jones polynomial

The Jones polynomial

A knot-invariant polynomial, discovered by Jones in the mid 80's. Appears in topological field theory, statistical physics, biology, and many more

The AJL algorithm

An efficient quantum algorithm to approximate the Jones polynomial of a knot by Aharonov, Jones & Landau (2005). Based on work of Kitaev, Freedman, Larsen & Wang on Topological Quantum Computation (2002).

- Not based on quantum Fourier transform;
- Uses a mapping of the combinatorial problem into an algebra, and then uses a unitary representation of the algebra that can be implemented on a quantum computer.
- Provides an additive approximation of the Jones Polynomial.
- Is quantum-complete.

This talk: extending the AJL approach for the Tutte polynomial

Main result

- An efficient approximation of the Tutte polynomial (a generalization of the Jones polynomial) at any point.
- For many points the approximation is quantum-hard.
- Many new quantum-complete problems.

Contributions

- **Breaking the unitarity barrier**
  - Quantum algorithm that uses non-unitary representations.
  - Quantum universality with non-unitary operators.
- **The Potts model:** possible implications to the long-standing problem of approximating the partition function of the q-state Potts model.

Outline

Outline

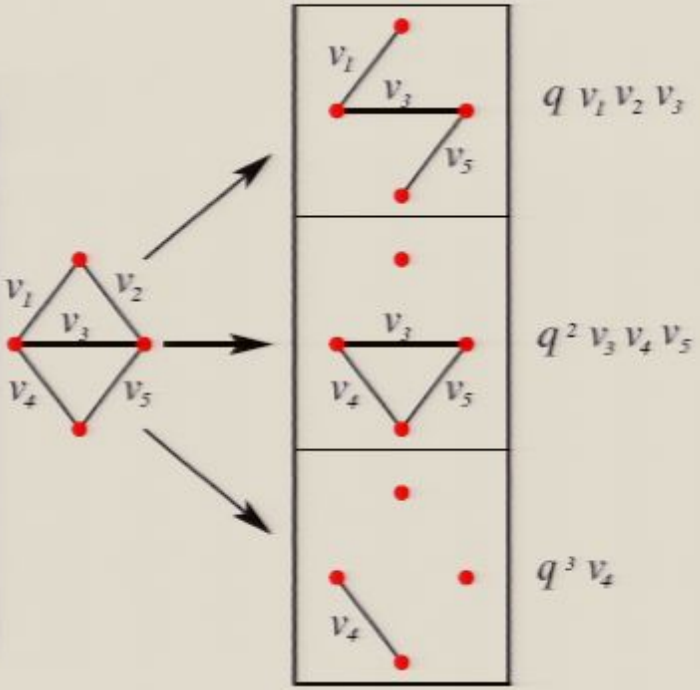
The (multivariate) Tutte polynomial

Definition

A defined for any graph $G = (V, E)$ with edge weights $\mathbf{v} \stackrel{\text{def}}{=} \{v_e | e \in E\}$

$$Z_G(q, \mathbf{v}) \stackrel{\text{def}}{=} \sum_{A \subseteq E} q^{k(A)} \prod_{e \in E} v_e$$

$k(A)$ - number of connected components in the subgraph $G = (V, A)$.



Some properties

- Encodes the reliability of a network, the number of spanning trees, chromatic polynomial and much more...
- Important special case: **the Jones polynomial**.
- For integer q and weights $v_e > -1$: the partition function of the q-state Potts model.
- Generalizes the standard Tutte polynomial $T_G(x, y)$ (when all weights are the same)

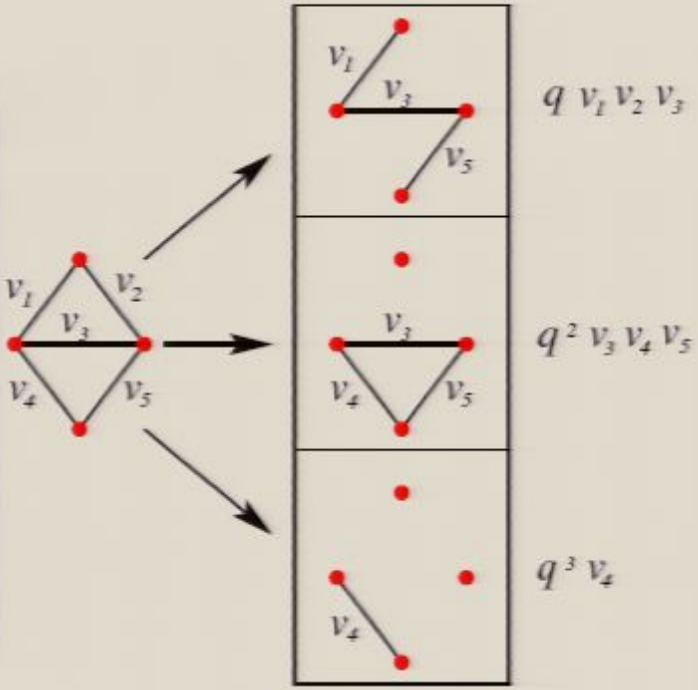
The (multivariate) Tutte polynomial

Definition

A defined for any graph $G = (V, E)$ with edge weights $\mathbf{v} \stackrel{\text{def}}{=} \{v_e | e \in E\}$

$$Z_G(q, \mathbf{v}) \stackrel{\text{def}}{=} \sum_{A \subseteq E} q^{k(A)} \prod_{e \in E} v_e$$

$k(A)$ - number of connected components in the subgraph $G = (V, A)$.



Some properties

- Encodes the reliability of a network, the number of spanning trees, chromatic polynomial and much more...
- Important special case: **the Jones polynomial**.
- For integer q and weights $v_e > -1$: the partition function of the q-state Potts model.
- Generalizes the standard Tutte polynomial $T_G(x, y)$ (when all weights are the same)

Known complexity results (standard Tutte polynomial)

Exact evaluation

$\#P$-hard for all (x, y) points but few trivial ones (Jaeger et al, 1990).

Multiplicative approximations (FPRAS)

- Exists for $q = 2$ with weights $v_e \geq 0$ (the Ferromagnetic Ising model) (Jerrum & Sinclair, 1993).
- NP-hard for $\sim 3/4$ of the (x, y) plane (Goldberg & Jerrum, 2006).
- For many points - including the ferromagnetic Potts model - complexity is unknown.

Can quantum computation do better?

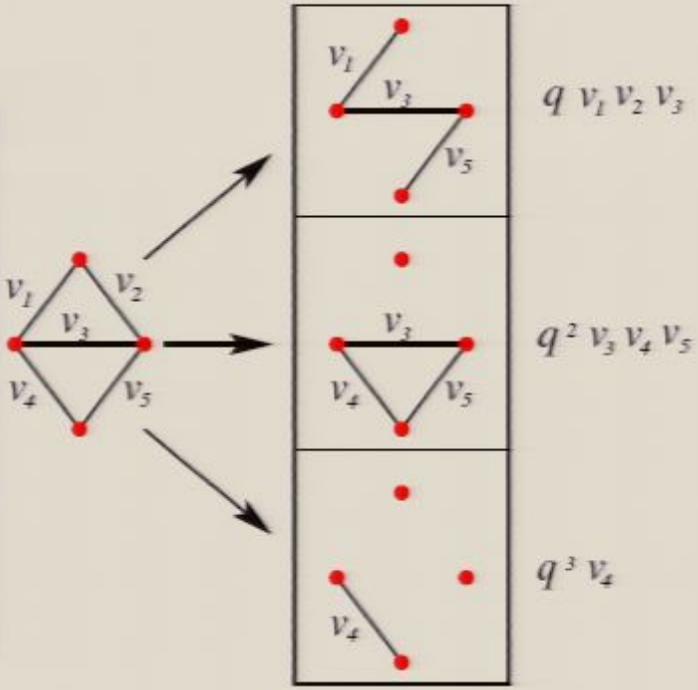
The (multivariate) Tutte polynomial

Definition

A defined for any graph $G = (V, E)$ with edge weights $\mathbf{v} \stackrel{\text{def}}{=} \{v_e | e \in E\}$

$$Z_G(q, \mathbf{v}) \stackrel{\text{def}}{=} \sum_{A \subseteq E} q^{k(A)} \prod_{e \in E} v_e$$

$k(A)$ - number of connected components in the subgraph $G = (V, A)$.



Some properties

- Encodes the reliability of a network, the number of spanning trees, chromatic polynomial and much more...
- Important special case: **the Jones polynomial**.
- For integer q and weights $v_e > -1$: the partition function of the q-state Potts model.
- Generalizes the standard Tutte polynomial $T_G(x, y)$ (when all weights are the same)

Known complexity results (standard Tutte polynomial)

Exact evaluation

$\#P$-hard for all (x, y) points but few trivial ones (Jaeger et al, 1990).

Multiplicative approximations (FPRAS)

- Exists for $q = 2$ with weights $v_e \geq 0$ (the Ferromagnetic Ising model) (Jerrum & Sinclair, 1993).
- NP-hard for $\sim 3/4$ of the (x, y) plane (Goldberg & Jerrum, 2006).
- For many points - including the ferromagnetic Potts model - complexity is unknown.

Can quantum computation do better?

Additive approximations - a relaxed notion of approximation

Suppose we wish to approximate the function $f(x)$ for a given x.

Multiplicative approximation

Find a y such that

$$f(x)(1-\epsilon) \leq y \leq f(x)(1+\epsilon)\,, \quad \epsilon = O(\frac{1}{poly(n)})$$

Additive approximation

Find a y such that

$$f(x) - \epsilon\Delta_x \leq y \leq f(x) + \epsilon\Delta_x\,, \quad \epsilon = O(\frac{1}{poly(n)})$$

Δ_x - The approximation scale of the problem (easy to calculate).

Δ_x is crucial: the problem might become trivial if $\Delta_x \gg f(x)$

The algorithm: rough statement of results

Theorem I (rough version)

There exists an efficient quantum algorithm that provides an additive approximation for the multivariate Tutte polynomial $Z_G(q, \mathbf{v})$, i.e., outputs a number Y

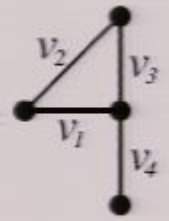
$$Z_G(q, \mathbf{v}) - \frac{\Delta}{poly(n)} \leq Y \leq Z_G(q, \mathbf{v}) + \frac{\Delta}{poly(n)}$$

for any planar graph $G = (V, E)$ with any set of weights $\mathbf{v} = \{v_e\}$ and any complex q.

The approximation scale Δ depends on $(G, q, \mathbf{v})$ and on the way in which G is embedded in the plane.

Outline of the quantum algorithm

Reminder: our goal

$G =$ 

$$Z_G(q, \mathbf{v}) \stackrel{\text{def}}{=} \sum_{A \subseteq E} q^{k(A)} \prod_{e \in E} v_e$$

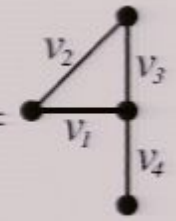
Find an additive approximation for $Z_G(q, \mathbf{v})$.

Outline of the quantum algorithm

$G = (V, E, \mathbf{v})$

Outline of the quantum algorithm

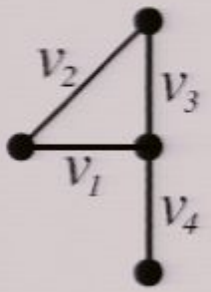
Reminder: our goal

$G =$ 

$$Z_G(q, \mathbf{v}) \stackrel{\text{def}}{=} \sum_{A \subseteq E} q^{k(A)} \prod_{e \in E} v_e$$

Find an additive approximation for $Z_G(q, \mathbf{v})$.

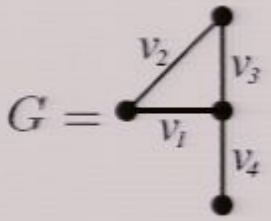
Outline of the quantum algorithm



$G = (V, E, \mathbf{v})$

Outline of the quantum algorithm

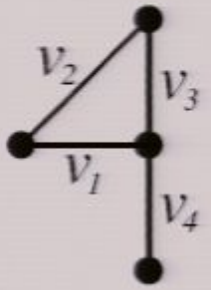
Reminder: our goal



$$Z_G(q, \mathbf{v}) \stackrel{\text{def}}{=} \sum_{A \subseteq E} q^{k(A)} \prod_{e \in E} v_e$$

Find an additive approximation for $Z_G(q, \mathbf{v})$.

Outline of the quantum algorithm

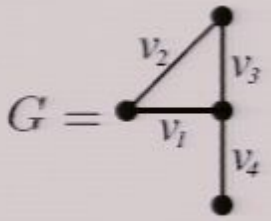


$G = (V, E, \mathbf{v})$

Weighted medial graph

Outline of the quantum algorithm

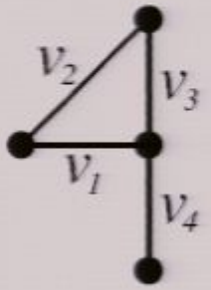
Reminder: our goal



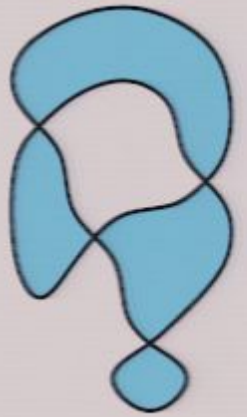
$$Z_G(q, \mathbf{v}) \stackrel{\text{def}}{=} \sum_{A \subseteq E} q^{k(A)} \prod_{e \in E} v_e$$

Find an additive approximation for $Z_G(q, \mathbf{v})$.

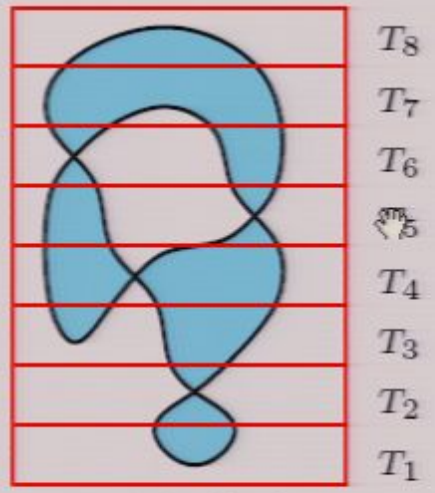
Outline of the quantum algorithm



$G = (V, E, \mathbf{v})$



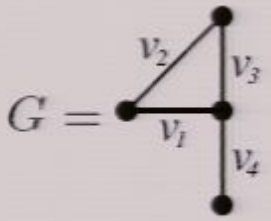
Weighted medial graph



$GTL(d)$ algebra
$T_8 \cdot \ldots \cdot T_1$

Outline of the quantum algorithm

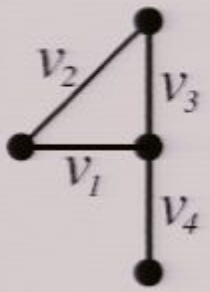
Reminder: our goal



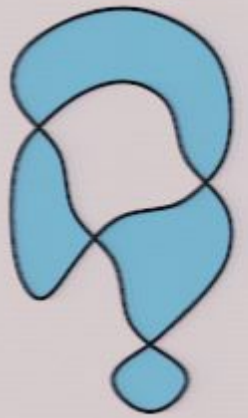
$$Z_G(q, \mathbf{v}) \stackrel{\text{def}}{=} \sum_{A \subseteq E} q^{k(A)} \prod_{e \in E} v_e$$

Find an additive approximation for $Z_G(q, \mathbf{v})$.

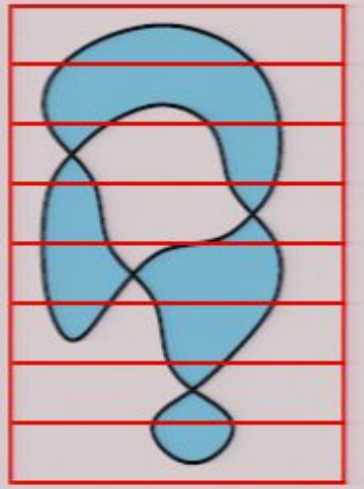
Outline of the quantum algorithm



$G = (V, E, \mathbf{v})$



Weighted medial graph



$GTL(d)$ algebra
$T_8 \cdot \ldots \cdot T_1$

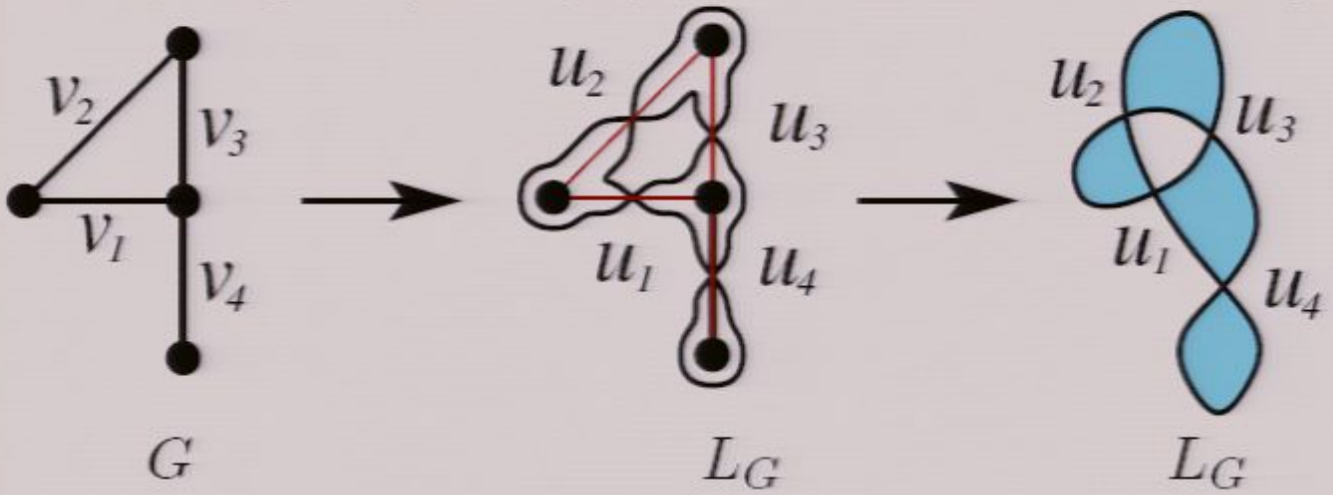
$$\begin{aligned} T_8 &\longrightarrow \rho(T_8) \\ T_7 &\longrightarrow \rho(T_7) \\ T_6 &\longrightarrow \rho(T_6) \\ T_5 &\longrightarrow \rho(T_5) \\ T_4 &\longrightarrow \rho(T_4) \\ T_3 &\longrightarrow \rho(T_3) \\ T_2 &\longrightarrow \rho(T_2) \\ T_1 &\longrightarrow \rho(T_1) \end{aligned}$$

Matrix Representation

From graphs to medial graphs

The Medial graph L_G

For any weighted planar graph G we can define a medial graph L_G.

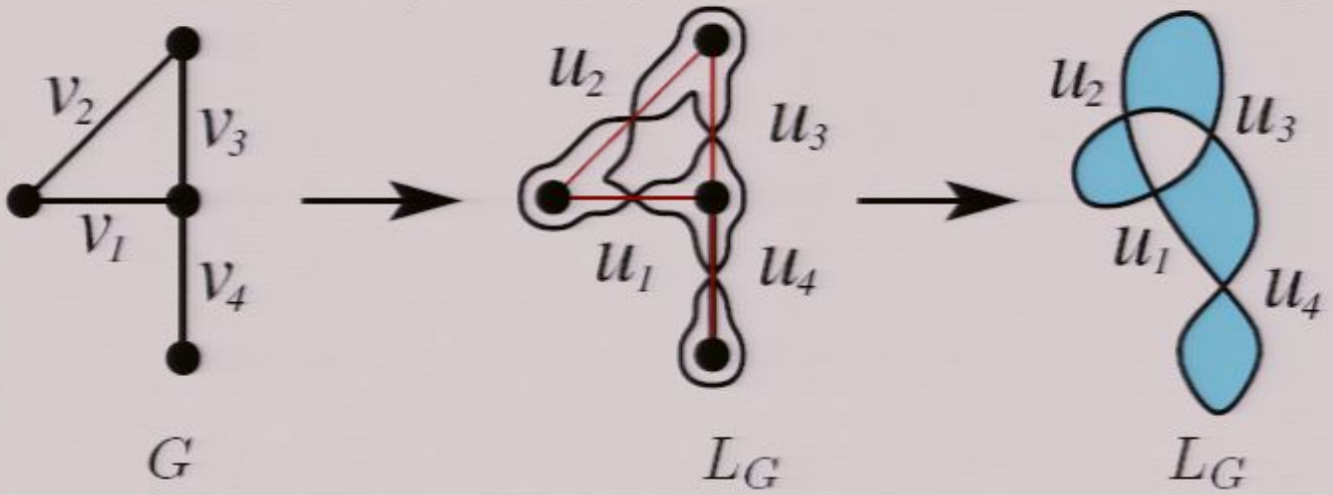


- L_G is a 4-regular planar graph.
- Edges in G are transformed into vertices of L_G, also called crossings.
- Edges weights v_i are transformed to crossing weights u_i.
- **Blue-and-White coloring:**
  1. Paint the external facet in white.
  2. Proceed by painting adjacent facets in opposite colors.
  3. $\Longrightarrow$ Blue regions correspond to vertices of G.

From graphs to medial graphs

The Medial graph L_G

For any weighted planar graph G we can define a medial graph L_G.



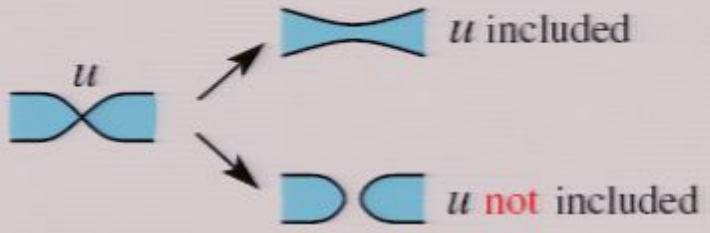
- L_G is a 4-regular planar graph.
- Edges in G are transformed into vertices of L_G, also called crossings.
- Edges weights v_i are transformed to crossing weights u_i.
- **Blue-and-White coloring:**
  1. Paint the external facet in white.
  2. Proceed by painting adjacent facets in opposite colors.
  3. $\Longrightarrow$ Blue regions correspond to vertices of G.

The weighted Kauffman bracket (equivalent to the Tutte polynomial)

$L_G \longrightarrow$ Kauffman bracket $\langle L_G \rangle (d, \mathbf{u})$.

The Kauffman Bracket

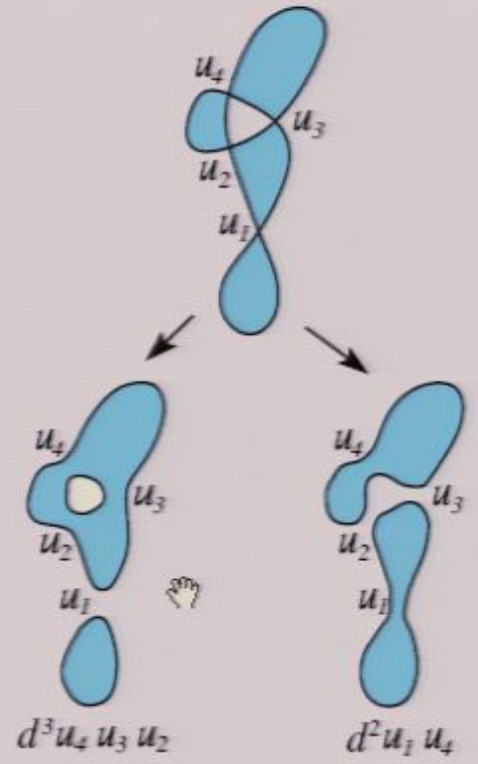
A polynomial that can be defined for any weighted medial graph L_G by opening the crossings in two ways:



$$\langle L_G \rangle (d, \mathbf{u}) \stackrel{\text{def}}{=} \sum_{\sigma} d^{\#\ell} \prod_{e \in \sigma} u_e$$

- d - a complex number
- σ An opening of the crossings. $e \in \sigma \Leftrightarrow$ shaded area connected.
- $\#\ell$ - Number of resulting loops

Opening a crossing



Relation to the Tutte polynomial

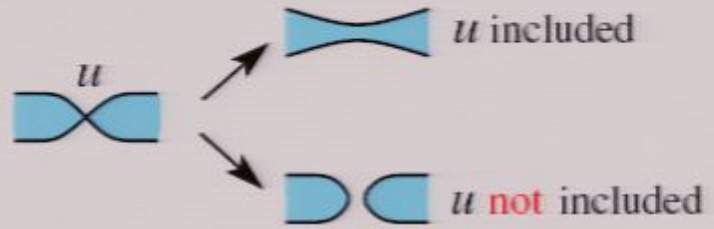
$$\langle L_G \rangle (d, \mathbf{u}) = d^{-|V|} Z_G(d^2, d\mathbf{u})$$

The weighted Kauffman bracket (equivalent to the Tutte polynomial)

$L_G \longrightarrow$ Kauffman bracket $\langle L_G \rangle (d, \mathbf{u})$.

The Kauffman Bracket

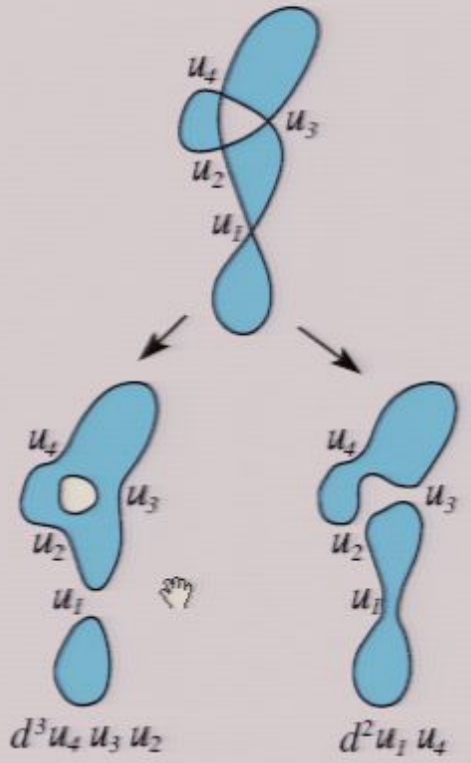
A polynomial that can be defined for any weighted medial graph L_G by opening the crossings in two ways:



$$\langle L_G \rangle (d, \mathbf{u}) \stackrel{\text{def}}{=} \sum_{\sigma} d^{\#\ell} \prod_{e \in \sigma} u_e$$

- d - a complex number
- σ An opening of the crossings. $e \in \sigma \Leftrightarrow$ shaded area connected.
- $\#\ell$ - Number of resulting loops

Opening a crossing



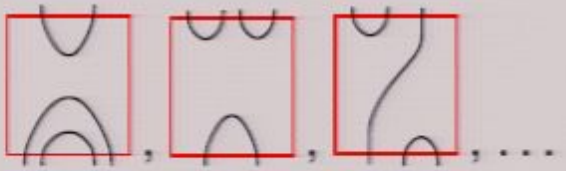
Relation to the Tutte polynomial

$$\langle L_G \rangle (d, \mathbf{u}) = d^{-|V|} Z_G(d^2, d\mathbf{u})$$

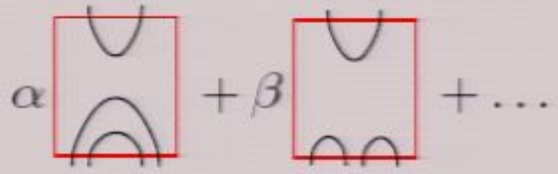
Moving to the language of algebras: the (generalized) Temperely-Lieb algebra

Definition: the $GTL(d)$ algebra

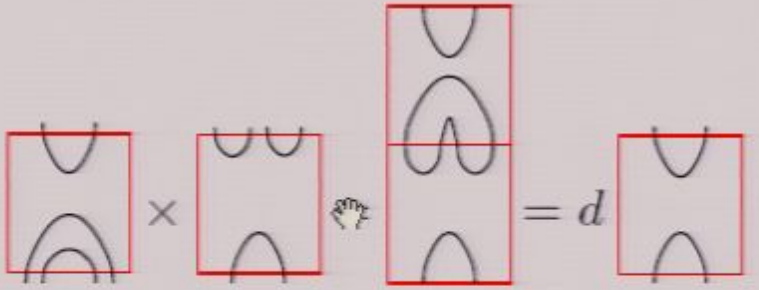
The $GTL(d)$ is generated by tangle diagrams, with no crossings and no loops. Number of in-pegs may be different from number of out-pegs.



- A general element is the weighted sum of such elements:



- **Product Rule:** simply place one on top of the other (when no. of strands match).



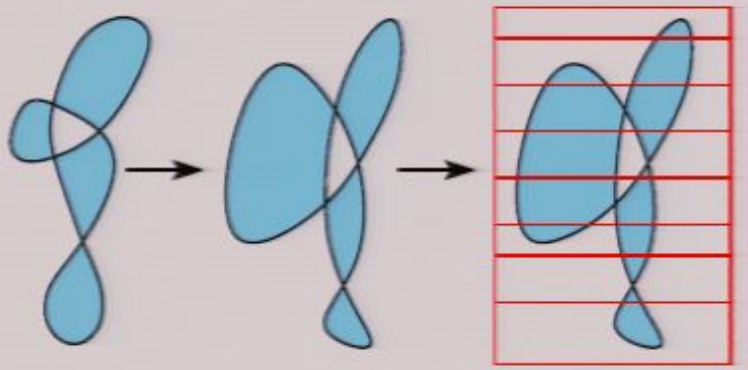
- Loops are taken out as d factors ($d =$ loop value).
- The identity element: $\mathcal{I} =$

From the medial graph to the $GTL(d)$ algebra

$\Psi : L_G \to GTL(d)$

We present L_G as a product of elementary diagrams:

- A Minima
- A Maxima
- A Crossing



All the operators are local!

Embedding an elementary diagram in $GTL(d)$

- A Minima:
- A Maxima:
- A Crossing: $- u +$

Corollary

$$\Psi(L_G) = (L_G)\, I$$

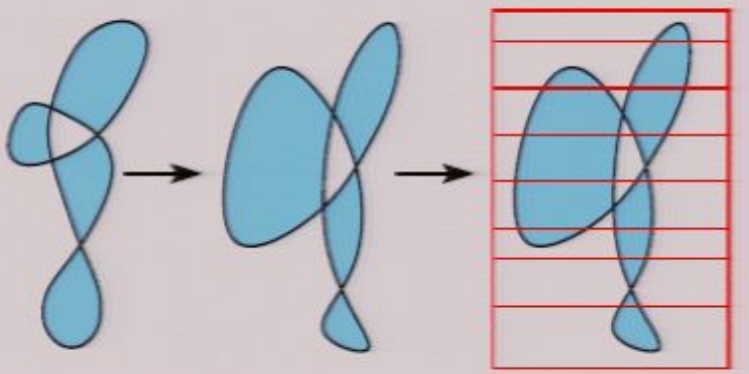
We want to find the prefactor before the identity!

From the medial graph to the $GTL(d)$ algebra

$\Psi : L_G \to GTL(d)$

We present L_G as a product of elementary diagrams:

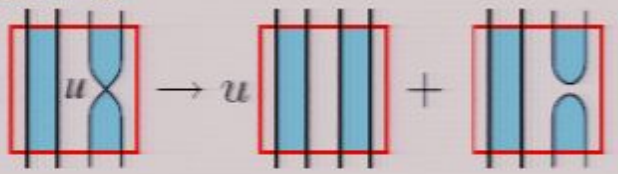
- A Minima
- A Maxima
- A Crossing



All the operators are local!

Embedding an elementary diagram in $GTL(d)$

- A Minima:
- A Maxima:
- A Crossing:



Corollary

$$\Psi(L_G) = \langle L_G \rangle I$$

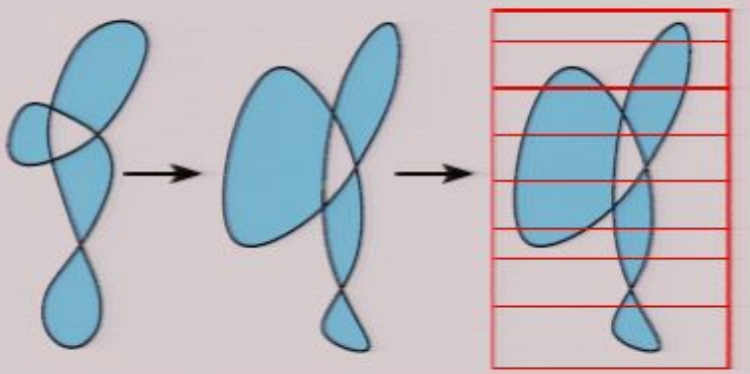
We want to find the prefactor before the identity!

From the medial graph to the $GTL(d)$ algebra

$\Psi : L_G \to GTL(d)$

We present L_G as a product of elementary diagrams:

- A Minima
- A Maxima
- A Crossing



All the operators are local!

Embedding an elementary diagram in $GTL(d)$

- A Minima:
- A Maxima:
- A Crossing: $u \to u +$

Corollary

$$\Psi(L_G) = \langle L_G \rangle \mathcal{I}$$

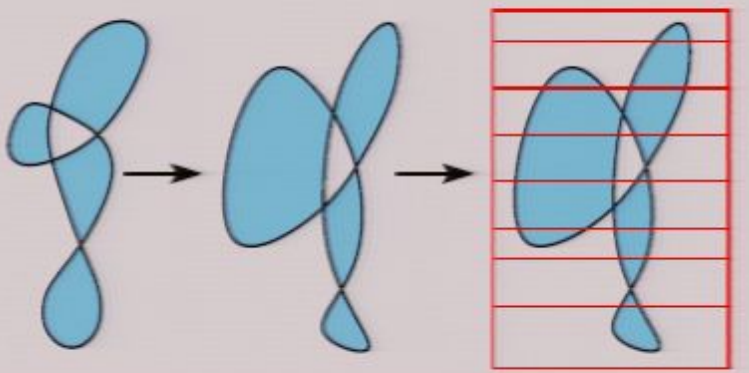
We want to find the prefactor before the identity!

From the medial graph to the $GTL(d)$ algebra

$\Psi : L_G \to GTL(d)$

We present L_G as a product of elementary diagrams:

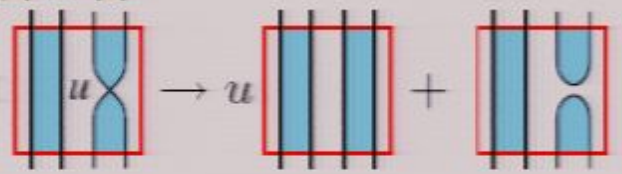
- A Minima
- A Maxima
- A Crossing



All the operators are local!

Embedding an elementary diagram in $GTL(d)$

- A Minima:
- A Maxima:
- A Crossing:



Corollary

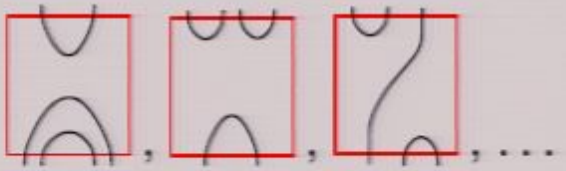
$$\Psi(L_G) = \langle L_G \rangle I$$

We want to find the prefactor before the identity!

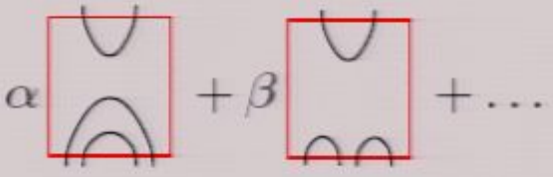
Moving to the language of algebras: the (generalized) Temperely-Lieb algebra

Definition: the $GTL(d)$ algebra

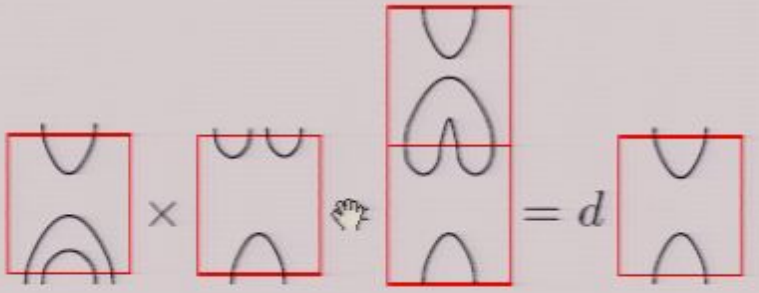
The $GTL(d)$ is generated by tangle diagrams, with no crossings and no loops. Number of in-pegs may be different from number of out-pegs.



- A general element is the weighted sum of such elements:



- **Product Rule:** simply place one on top of the other (when no. of strands match).



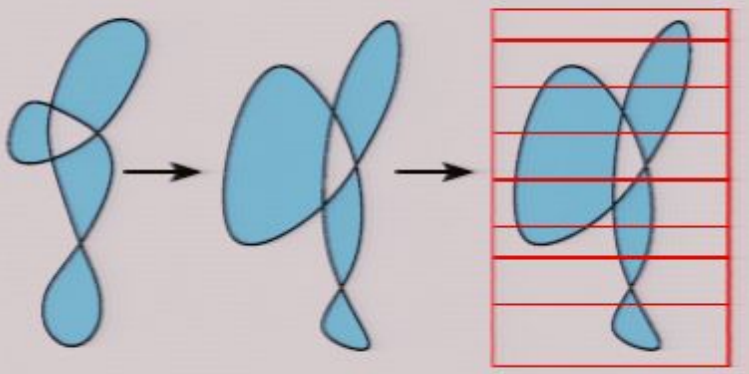
- Loops are taken out as d factors ($d =$ loop value).
- The identity element: $\mathcal{I} =$

From the medial graph to the $GTL(d)$ algebra

$\Psi : L_G \to GTL(d)$

We present L_G as a product of elementary diagrams:

- A Minima
- A Maxima
- A Crossing



All the operators are local!

Embedding an elementary diagram in $GTL(d)$

- A Minima:
- A Maxima:
- A Crossing: $-u+$

Corollary

$$\Psi(L_G) = (L_G)\,I$$

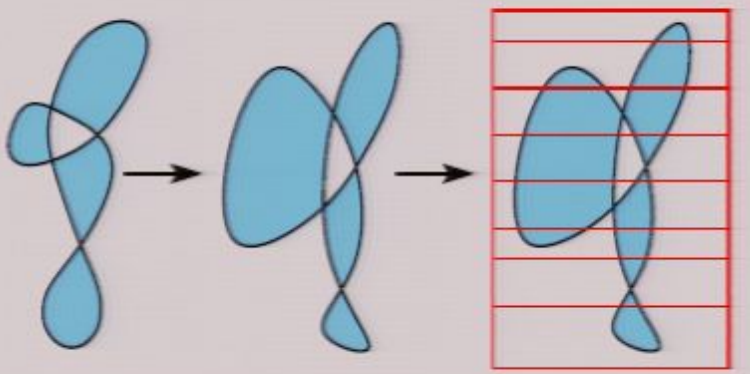
We want to find the prefactor before the identity!

From the medial graph to the $GTL(d)$ algebra

$\Psi : L_G \to GTL(d)$

We present L_G as a product of elementary diagrams:

- A Minima
- A Maxima
- A Crossing



All the operators are local!

Embedding an elementary diagram in $GTL(d)$

- A Minima:
- A Maxima:
- A Crossing: $u \to u +$

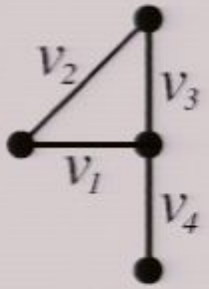
Corollary

$$\Psi(L_G) = \langle L_G \rangle \mathcal{I}$$

We want to find the prefactor before the identity!

Intermediate summary

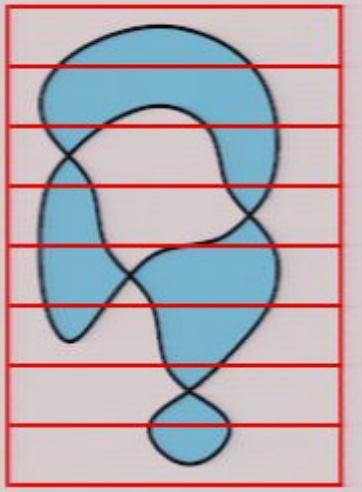
Outline of the quantum algorithm so far



Weighted planar graph $G = (V, E)$

Weighted medial graph

$\Psi \longrightarrow$



$GTL(d)$ algebra $\Psi(L_G) = \langle L_G \rangle \mathcal{I}$.

$T_8 \longrightarrow \rho(T_8)$
$T_7 \longrightarrow \rho(T_7)$
$T_6 \longrightarrow \rho(T_6)$
$T_5 \longrightarrow \rho(T_5)$
$T_4 \longrightarrow \rho(T_4)$
$T_3 \longrightarrow \rho(T_3)$
$T_2 \longrightarrow \rho(T_2)$
$T_1 \longrightarrow \rho(T_1)$

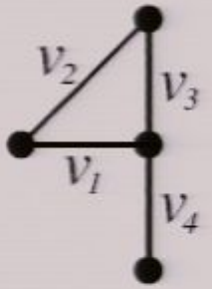
Move to matrix representation of $GTL(d)$

We need a representation that can be applied efficiently. We then apply

$$\rho(T_8) \cdot \ldots \cdot \rho(T_1) = \rho(T_8 \cdot \ldots \cdot T_1) = \rho((L_G)\mathcal{I}) = \langle L_G \rangle \mathbb{1}$$

Intermediate summary

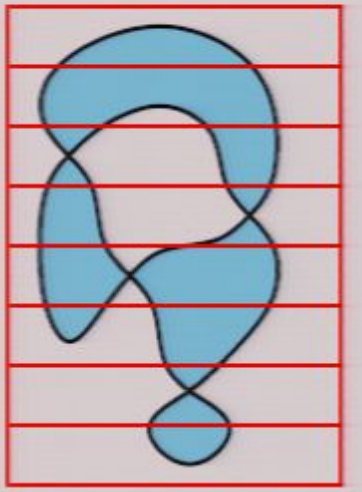
Outline of the quantum algorithm so far



Weighted planar graph $G = (V, E)$

Weighted medial graph

$\Psi \longrightarrow$



$GTL(d)$ algebra $\Psi(L_G) = \langle L_G \rangle \mathcal{I}$.

$$
\begin{aligned}
T_8 &\longrightarrow \rho(T_8)\\
T_7 &\longrightarrow \rho(T_7)\\
T_6 &\longrightarrow \rho(T_6)\\
T_5 &\longrightarrow \rho(T_5)\\
T_4 &\longrightarrow \rho(T_4)\\
T_3 &\longrightarrow \rho(T_3)\\
T_2 &\longrightarrow \rho(T_2)\\
T_1 &\longrightarrow \rho(T_1)
\end{aligned}
$$

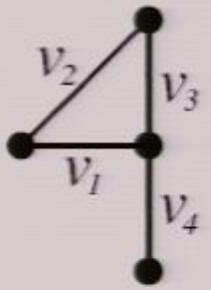
Move to matrix representation of $GTL(d)$

We need a representation that can be applied efficiently. We then apply

$$\rho(T_8) \cdot \ldots \cdot \rho(T_1) = \rho(T_8 \cdot \ldots \cdot T_1) = \rho((L_G)\mathcal{I}) = \langle L_G \rangle \mathbb{1}$$

Intermediate summary

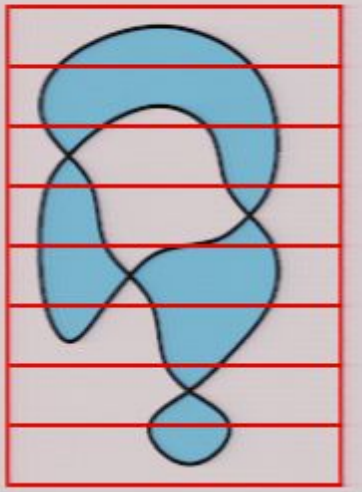
Outline of the quantum algorithm so far



Weighted planar graph $G = (V, E)$

Weighted medial graph

$\Psi \longrightarrow$



$GTL(d)$ algebra $\Psi(L_G) = \langle L_G \rangle \mathcal{I}$.

$$
\begin{aligned}
T_8 &\longrightarrow \rho(T_8)\\
T_7 &\longrightarrow \rho(T_7)\\
T_6 &\longrightarrow \rho(T_6)\\
T_5 &\longrightarrow \rho(T_5)\\
T_4 &\longrightarrow \rho(T_4)\\
T_3 &\longrightarrow \rho(T_3)\\
T_2 &\longrightarrow \rho(T_2)\\
T_1 &\longrightarrow \rho(T_1)
\end{aligned}
$$

Move to matrix representation of $GTL(d)$

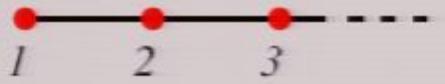
We need a representation that can be applied efficiently. We then apply

$$\rho(T_8) \cdot \ldots \cdot \rho(T_1) = \rho(T_8 \cdot \ldots \cdot T_1) = \rho\big(\langle L_G \rangle \mathcal{I}\big) = \langle L_G \rangle \mathbb{1}$$

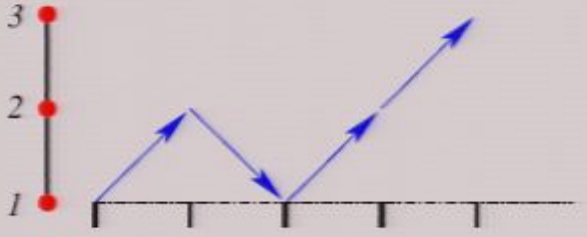
The Path-Model representation

The $\{H_n\}$ spaces

Spanned by n-steps paths on the line, starting from 1.



$|12123\rangle \in H_4$ is:



A diagram with 0 lower strands and 0 upper strands is simply a linear transformation from $H_0 = span\{|1\rangle\}$ into itself - a simple scalar multiplication.

How does a diagram operate on paths ?

- Paint the regions of the diagram.
- Diagrams with n lower strands and m upper strands define a transformation $H_n \to H_m$.
- Example:

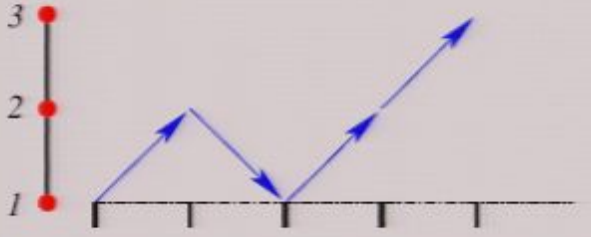
$$|12323\rangle \to a\,|1212323\rangle + \beta\,|1212343\rangle$$

The Path-Model representation

The $\{H_n\}$ spaces

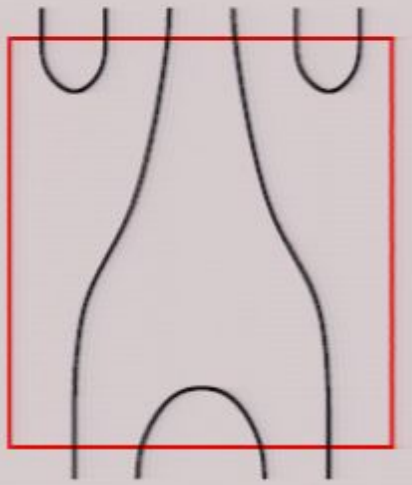
Spanned by n-steps paths on the line, starting from 1.

$|12123\rangle \in H_4$ is:



A diagram with 0 lower strands and 0 upper strands is simply a linear transformation from $H_0 = span\{|1\rangle\}$ into itself - a simple scalar multiplication.

How does a diagram operate on paths ?

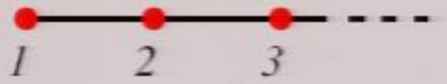


- Paint the regions of the diagram.
- Diagrams with n lower strands and m upper strands define a transformation $H_n \to H_m$.
- Example:

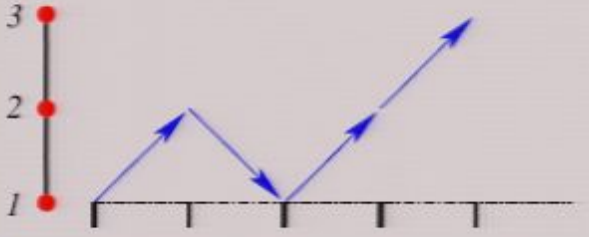
The Path-Model representation

The $\{H_n\}$ spaces

Spanned by n-steps paths on the line, starting from 1.

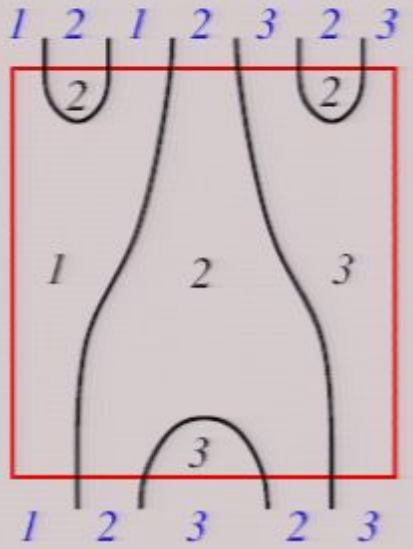


$|12123\rangle \in H_4$ is:



A diagram with 0 lower strands and 0 upper strands is simply a linear transformation from $H_0 = span\{|1\rangle\}$ into itself - a simple scalar multiplication.

How does a diagram operate on paths ?



H_6
$\uparrow$
H_4

- Paint the regions of the diagram.
- Diagrams with n lower strands and m upper strands define a transformation $H_n \to H_m$.
- Example:

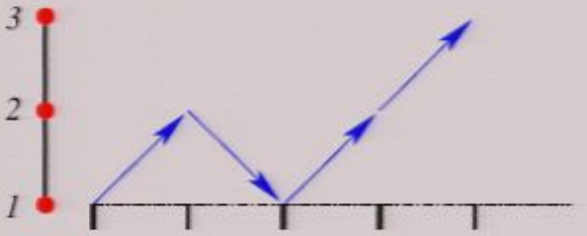
$$|12323\rangle \to \alpha\,|1212323\rangle + \beta\,|1212343\rangle$$

The Path-Model representation

The $\{H_n\}$ spaces

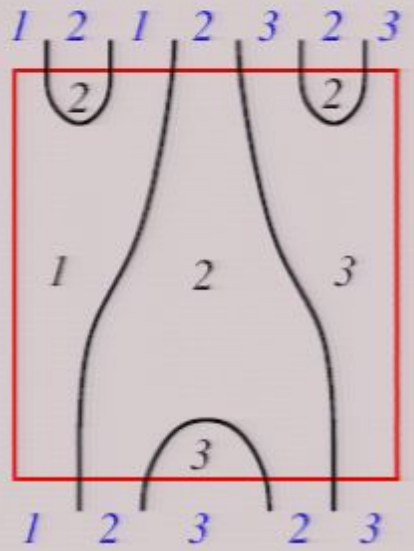
Spanned by n-steps paths on the line, starting from 1.

$|12123\rangle \in H_4$ is:



A diagram with 0 lower strands and 0 upper strands is simply a linear transformation from $H_0 = span\{|1\rangle\}$ into itself - a simple scalar multiplication.

How does a diagram operate on paths ?



H_6
$\uparrow$
H_4

- Paint the regions of the diagram.
- Diagrams with n lower strands and m upper strands define a transformation $H_n \to H_m$.
- Example:

$$|12323\rangle \to \alpha\,|1212323\rangle + \beta\,|1212343\rangle$$

Calculating $\langle L_G \rangle$ with the path-model representation

Recall

$$\rho(T_N) \cdot \ldots \cdot \rho(T_1) = \rho(T_N \cdot \ldots \cdot T_1) = \langle L_G \rangle \mathbb{1}$$

Calculating $\langle L_G \rangle$ with matrices

$|1\rangle \in H_0$

Calculating $\langle L_G \rangle$ with the path-model representation

Recall

$$\rho(T_N) \cdot \ldots \cdot \rho(T_1) = \rho(T_N \cdot \ldots \cdot T_1) = \langle L_G \rangle \mathbb{1}$$

Calculating $\langle L_G \rangle$ with matrices

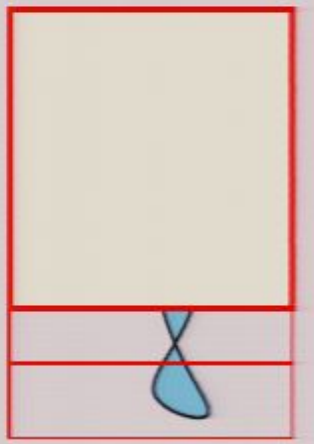
$|\psi_1\rangle = \rho(T_1)\,|1\rangle \in H_2$

Calculating $\langle L_G \rangle$ with the path-model representation

Recall

$$\rho(T_N) \cdot \ldots \cdot \rho(T_1) = \rho(T_N \cdot \ldots \cdot T_1) = \langle L_G \rangle \mathbb{1}$$

Calculating $\langle L_G \rangle$ with matrices



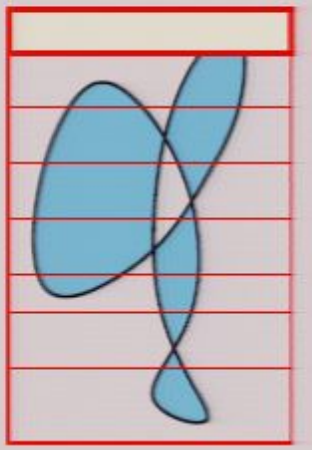
$$|\psi_2\rangle = \rho(T_2)\rho(T_1)\,|1\rangle \in H_2$$

Calculating $\langle L_G \rangle$ with the path-model representation

Recall

$$\rho(T_N) \cdot \ldots \cdot \rho(T_1) = \rho(T_N \cdot \ldots \cdot T_1) = \langle L_G \rangle \mathbb{1}$$

Calculating $\langle L_G \rangle$ with matrices



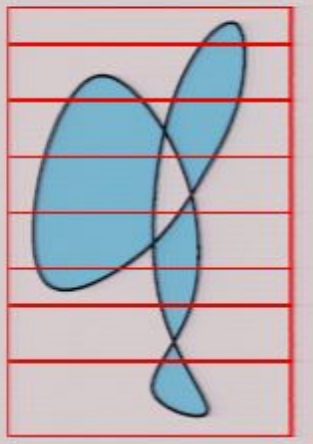
$$|\psi_{N-1}\rangle = \rho(T_{N-1}) \cdots \rho(T_1) \, |1\rangle$$

Calculating $\langle L_G \rangle$ with the path-model representation

Recall

$$\rho(T_N) \cdot \ldots \cdot \rho(T_1) = \rho(T_N \cdot \ldots \cdot T_1) = \langle L_G \rangle \mathbb{1}$$

Calculating $\langle L_G \rangle$ with matrices



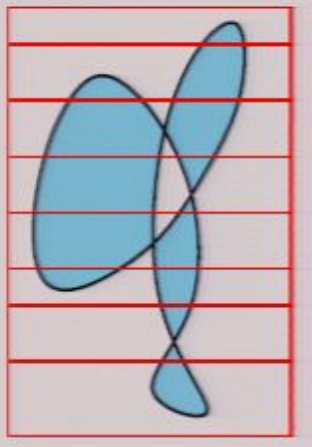
$$|\psi_N\rangle = \rho(T_N) \cdots \rho(T_1) |1\rangle$$

Calculating $\langle L_G \rangle$ with the path-model representation

Recall

$$\rho(T_N) \cdot \ldots \cdot \rho(T_1) = \rho(T_N \cdot \ldots \cdot T_1) = \langle L_G \rangle \mathbb{1}$$

Calculating $\langle L_G \rangle$ with matrices



$$|\psi_N\rangle = \rho(T_N) \cdots \rho(T_1) |1\rangle = \langle L_G \rangle |1\rangle$$

To calculate $\langle L_G \rangle(d, \mathbf{u})$, we take the inner product:

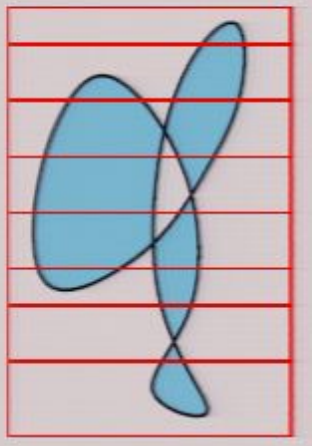
$$\langle L_G \rangle(d, \mathbf{u}) = \langle 1| \overbrace{\rho(T_N) \cdots \rho(T_1)}^{\langle L_G \rangle \mathbb{I}} |1\rangle$$

Calculating $\langle L_G \rangle$ with the path-model representation

Recall

$$\rho(T_N) \cdot \ldots \cdot \rho(T_1) = \rho(T_N \cdot \ldots \cdot T_1) = \langle L_G \rangle \mathbb{1}$$

Calculating $\langle L_G \rangle$ with matrices



Problem: it looks like we can only use unitary representations $\rho(T_i)$, which severely limits the set of parameters $(d, \mathbf{u})$.

$$\langle L_G \rangle(d, \mathbf{u}) = \langle 1| \overbrace{\rho(T_N) \cdots \rho(T_1)}^{\langle L_G \rangle \mathbb{I}} |1\rangle$$

Calculating $\langle L_G \rangle$ with the path-model representation

Recall

$$\rho(T_N) \cdot \ldots \cdot \rho(T_1) = \rho(T_N \cdot \ldots \cdot T_1) = \langle L_G \rangle \mathbb{1}$$

Calculating $\langle L_G \rangle$ with matrices

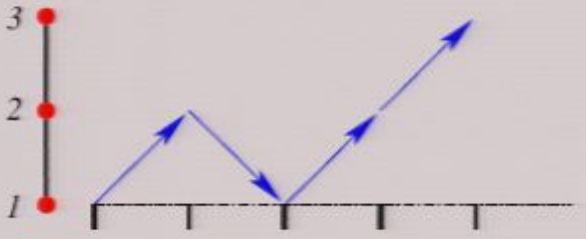
$|\psi_1\rangle = \rho(T_1)\,|1\rangle \in H_2$

The Path-Model representation

The $\{H_n\}$ spaces

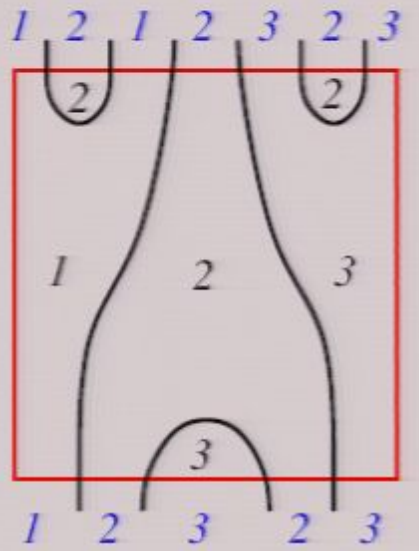
Spanned by n-steps paths on the line, starting from 1.

$|12123\rangle \in H_4$ is:



A diagram with 0 lower strands and 0 upper strands is simply a linear transformation from $H_0 = span\{|1\rangle\}$ into itself - a simple scalar multiplication.

How does a diagram operate on paths ?



H_6
$\uparrow$
H_4

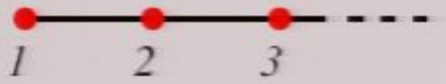
- Paint the regions of the diagram.
- Diagrams with n lower strands and m upper strands define a transformation $H_n \to H_m$.
- Example:

$$|12323\rangle \to \alpha\,|1212323\rangle + \beta\,|1212343\rangle$$

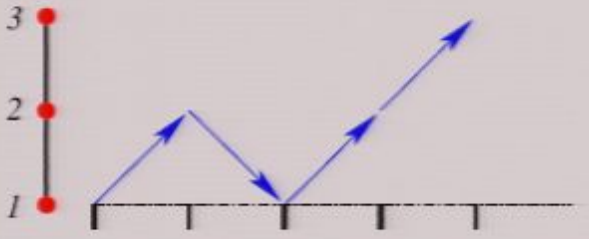
The Path-Model representation

The $\{H_n\}$ spaces

Spanned by n-steps paths on the line, starting from 1.

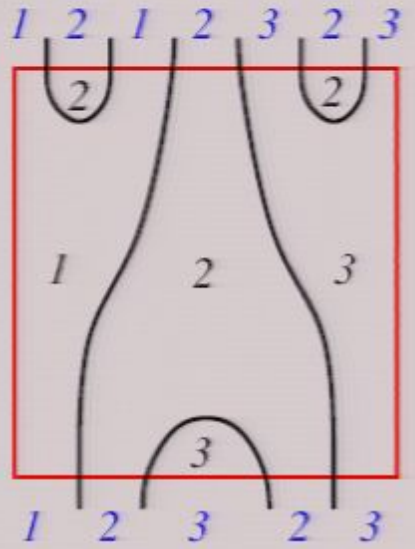


$|12123\rangle \in H_4$ is:



A diagram with 0 lower strands and 0 upper strands is simply a linear transformation from $H_0 = span\{|1\rangle\}$ into itself - a simple scalar multiplication.

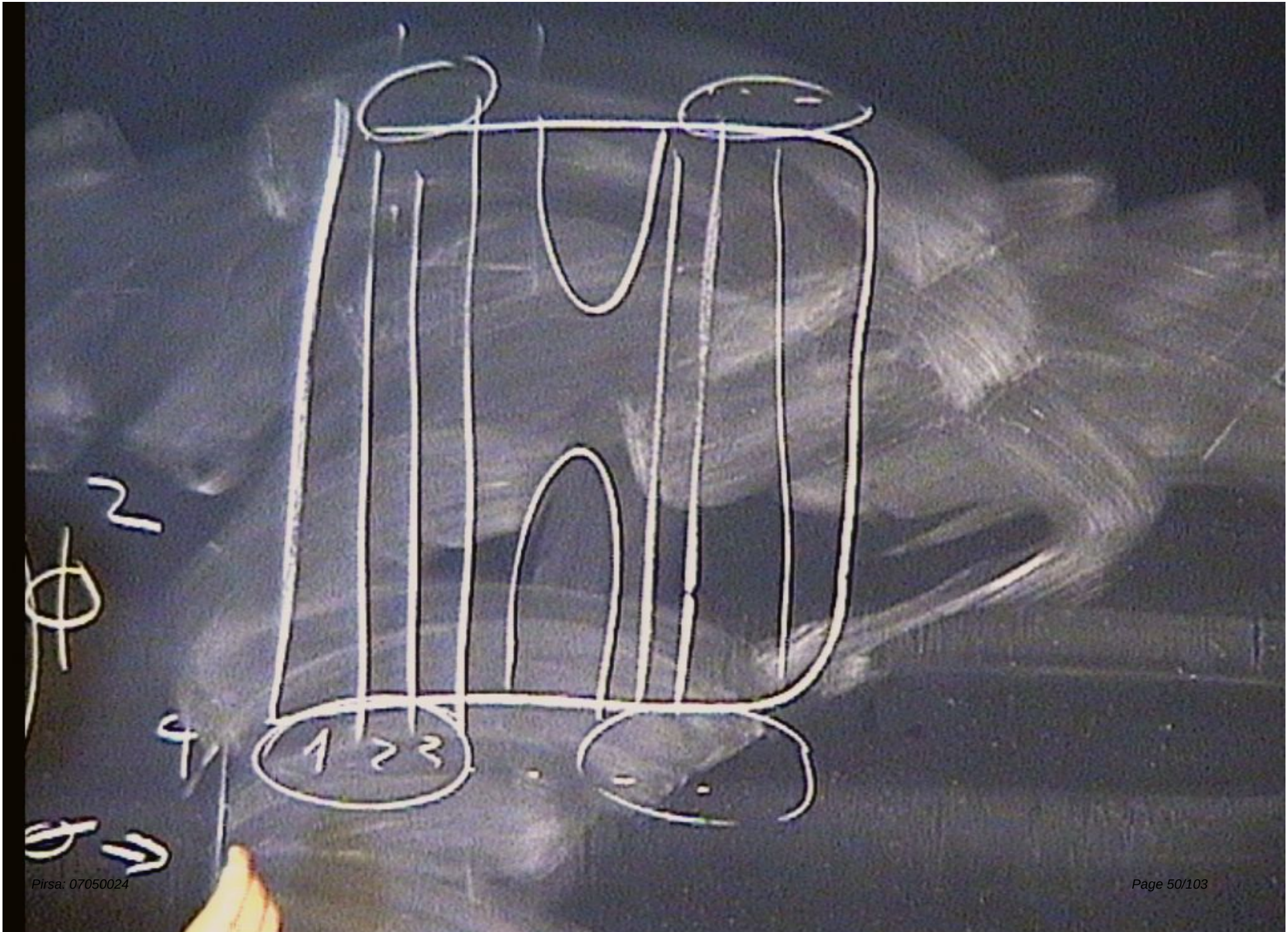
How does a diagram operate on paths ?

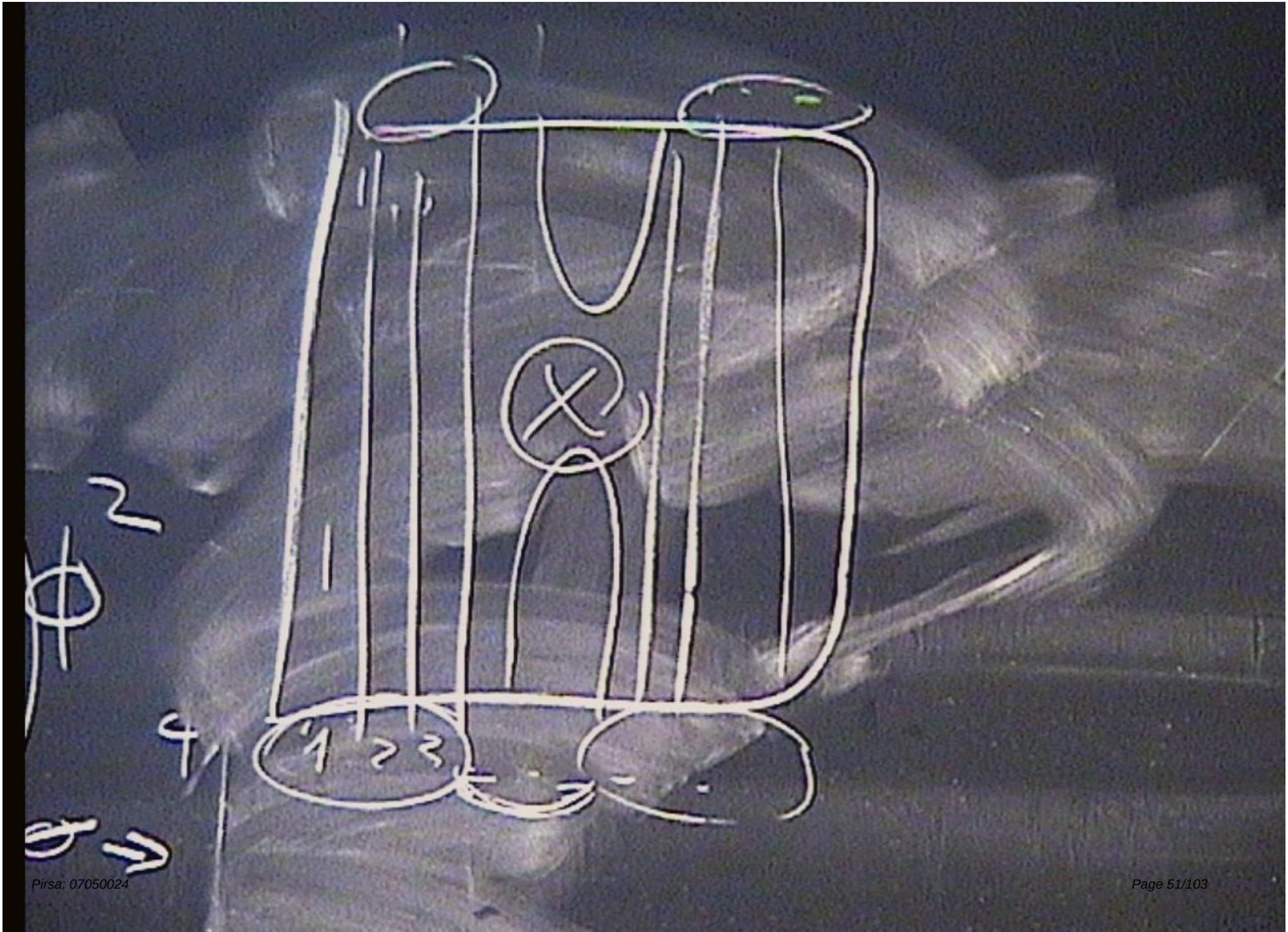


H_6
$\uparrow$
H_4

- Paint the regions of the diagram.
- Diagrams with n lower strands and m upper strands define a transformation $H_n \to H_m$.
- Example:

$$|12323\rangle \to \alpha\,|1212323\rangle + \beta\,|1212343\rangle$$


2
1 2 3

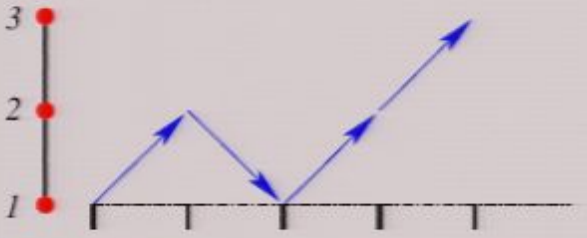

1 2 3

The Path-Model representation

The $\{H_n\}$ spaces

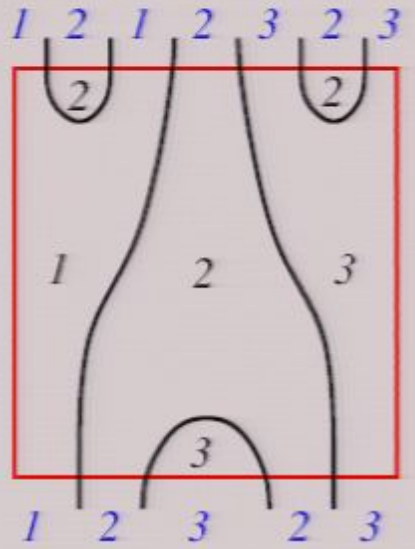
Spanned by n-steps paths on the line, starting from 1.

$|12123\rangle \in H_4$ is:



A diagram with 0 lower strands and 0 upper strands is simply a linear transformation from $H_0 = span\{|1\rangle\}$ into itself - a simple scalar multiplication.

How does a diagram operate on paths ?



H_6
$\uparrow$
H_4

- Paint the regions of the diagram.
- Diagrams with n lower strands and m upper strands define a transformation $H_n \to H_m$.
- Example:

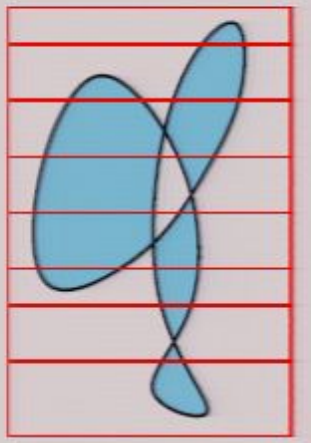
$$|12323\rangle \to \alpha\,|1212323\rangle + \beta\,|1212343\rangle$$

Calculating $\langle L_G \rangle$ with the path-model representation

Recall

$$\rho(T_N) \cdot \ldots \cdot \rho(T_1) = \rho(T_N \cdot \ldots \cdot T_1) = \langle L_G \rangle \mathbb{1}$$

Calculating $\langle L_G \rangle$ with matrices



$$|\psi_N\rangle = \rho(T_N) \cdots \rho(T_1) |1\rangle = \langle L_G \rangle |1\rangle$$

To calculate $\langle L_G \rangle(d, \mathbf{u})$, we take the inner product:

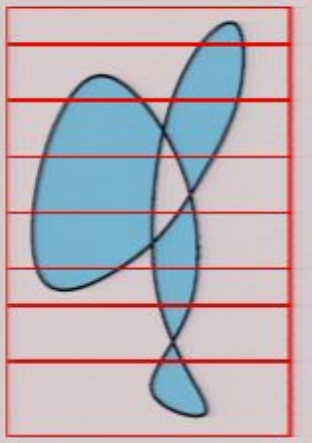
$$\langle L_G \rangle(d, \mathbf{u}) = \langle 1| \overbrace{\rho(T_N) \cdots \rho(T_1)}^{\langle L_G \rangle \mathbb{I}} |1\rangle$$

Calculating $\langle L_G \rangle$ with the path-model representation

Recall

$$\rho(T_N) \cdot \ldots \cdot \rho(T_1) = \rho(T_N \cdot \ldots \cdot T_1) = \langle L_G \rangle \mathbb{1}$$

Calculating $\langle L_G \rangle$ with matrices



Problem: it looks like we can only use unitary representations $\rho(T_i)$, which severely limits the set of parameters $(d, \mathbf{u})$.

$$\langle L_G \rangle(d, \mathbf{u}) = \langle 1| \overbrace{\rho(T_N) \cdots \rho(T_1)}^{\langle L_G \rangle \mathbb{I}} |1\rangle$$

Implementing non-unitary elementary operators

Crossing operator

- Reminder: The **Crossing** operator is a (not necessarly unitary) square matrix $M : H_n \multimap H_n$.

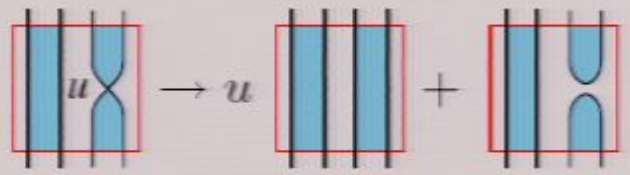
Minima & Maxima operators

- A **Minima** ($H_n \multimap H_{n+2}$) is implemented by adding qubits.
- A **Maxima** ($H_n \multimap H_{n-2}$) is implemented by discarding qubits.

Implementing non-unitary elementary operators

Crossing operator

- Reminder: The **Crossing** operator is a (not necessarly unitary) square matrix $M : H_n \to H_n$.



We use the Polar Decomposition $M = PU$

Minima & Maxima operators

- A **Minima** ($H_n \multimap H_{n+2}$) is implemented by adding qubits.
- A **Maxima** ($H_n \multimap H_{n-2}$) is implemented by discarding qubits.

Implementing non-unitary elementary operators

Crossing operator

- Reminder: The **Crossing** operator is a (not necessarly unitary) square matrix $M : H_n \to H_n$.

We use the Polar Decomposition $M = PU$

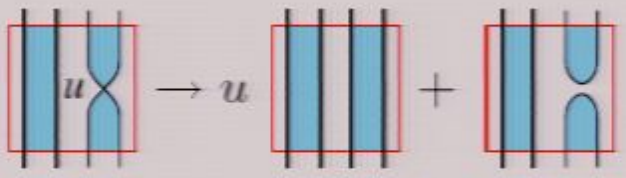
Minima & Maxima operators

- A **Minima** ($H_n \to H_{n+2}$) is implemented by adding qubits.
- A **Maxima** ($H_n \to H_{n-2}$) is implemented by discarding qubits.

Implementing non-unitary elementary operators

Crossing operator

- Reminder: The **Crossing** operator is a (not necessarly unitary) square matrix $M : H_n \to H_n$.



We use the Polar Decomposition $M = PU$

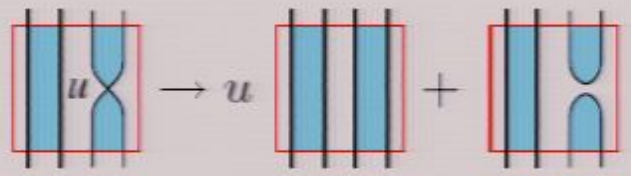
Minima & Maxima operators

- A **Minima** ($H_n \to H_{n+2}$) is implemented by adding qubits.
- A **Maxima** ($H_n \to H_{n-2}$) is implemented by discarding qubits.

Implementing non-unitary elementary operators

Crossing operator

- Reminder: The **Crossing** operator is a (not necessarily unitary) square matrix $M : H_n \to H_n$.



We use the Polar Decomposition $M = PU$

$\Rightarrow U$ is unitary - easy to implement (because of locality).

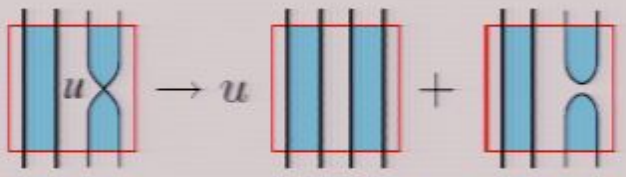
Minima & Maxima operators

- A **Minima** ($H_n \to H_{n+2}$) is implemented by adding qubits.
- A **Maxima** ($H_n \to H_{n-2}$) is implemented by discarding qubits.

Implementing non-unitary elementary operators

Crossing operator

- Reminder: The **Crossing** operator is a (not necessarily unitary) square matrix $M : H_n \to H_n$.



We use the Polar Decomposition $M = PU$

$\Rightarrow P = diag(r_1, r_2, \ldots,)$

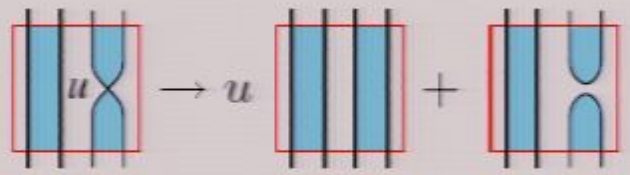
Minima & Maxima operators

- A **Minima** ($H_n \to H_{n+2}$) is implemented by adding qubits.
- A **Maxima** ($H_n \to H_{n-2}$) is implemented by discarding qubits.

Implementing non-unitary elementary operators

Crossing operator

- Reminder: The **Crossing** operator is a (not necessarly unitary) square matrix $M : H_n \to H_n$.



We use the Polar Decomposition $M = PU$

$\Rightarrow P = diag(r_1, r_2, \ldots,)$

We implement $\frac{P}{||P||} = diag(1, r_2/r_1, r_3/r_1, \ldots,)$ by pushing the weights to an ancilla qubit

$$|i\rangle \otimes |0\rangle \to \frac{r_i}{r_1} |i\rangle \otimes |0\rangle + \sqrt{1 - (r_i/r_1)^2}\, |r_i\rangle \otimes |1\rangle$$

and book-keeping the norm $||P|| = ||M||$.

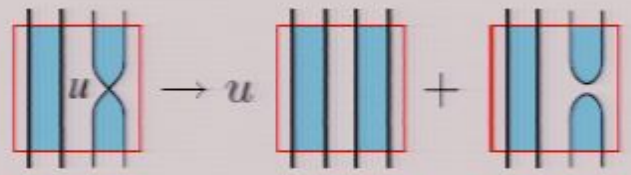
Minima & Maxima operators

- A **Minima** ($H_n \to H_{n+2}$) is implemented by adding qubits.
- A **Maxima** ($H_n \to H_{n-2}$) is implemented by discarding qubits.

Implementing non-unitary elementary operators

Crossing operator

- Reminder: The **Crossing** operator is a (not necessarly unitary) square matrix $M : H_n \to H_n$.



We use the Polar Decomposition $M = PU$

$\Rightarrow P = diag(r_1, r_2, \dots,)$

We implement $\frac{P}{||P||} = diag(1, r_2/r_1, r_3/r_1, \dots,)$ by pushing the weights to an ancilla qubit

$$|i\rangle \otimes |0\rangle \to \frac{r_i}{r_1} |i\rangle \otimes |0\rangle + \sqrt{1 - (r_i/r_1)^2}\, |r_i\rangle \otimes |1\rangle$$

and book-keeping the norm $||P|| = ||M||$.

Minima & Maxima operators

- A **Minima** $(H_n \to H_{n+2})$ is implemented by adding qubits.
- A **Maxima** $(H_n \to H_{n-2})$ is implemented by discarding qubits.

Summary of quantum algorithm

- The computation starts with the state $|1\rangle$ (a path with 0 steps).
- Apply the elementary operators: minimas, maximas, crossings.
- In the end, we get

$$|\psi_N\rangle = \frac{\langle L_G \rangle}{||\rho(T_1)|| \cdot \ldots \cdot ||\rho(T_N)||} |1\rangle \otimes \overbrace{|0\rangle \otimes \ldots \otimes |0\rangle}^{\text{ancilla qubits}} + \ldots$$

- Estimate the inner product $\langle 1 \otimes 0 \otimes \ldots \otimes 0 | \psi_N \rangle$ using the Hadamard test, Obtain:

$$\frac{\langle L_G \rangle}{||\rho(T_1)|| \cdot \ldots \cdot ||\rho(T_N)||} \pm 1/poly(n)$$

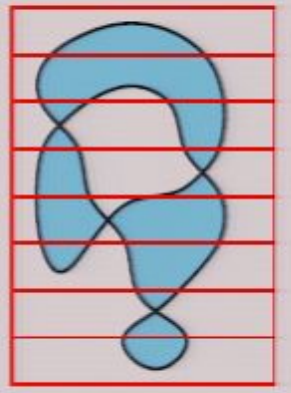
The approximation scale

$$\Delta_{alg} = ||\rho(T_1)|| \cdot \ldots \cdot ||\rho(T_N)||$$

Summary of quantum algorithm

- The computation starts with the state $|1\rangle$ (a path with 0 steps).
- Apply the elementary operators: minimas, maximas, crossings.
- In the end, we get

$$|\psi_N\rangle = \frac{\langle L_G \rangle}{||\rho(T_1)|| \cdot \ldots \cdot ||\rho(T_N)||} |1\rangle \otimes \overbrace{|0\rangle \otimes \ldots \otimes |0\rangle}^{\text{ancilla qubits}} + \ldots$$



- Estimate the inner product $\langle 1 \otimes 0 \otimes \ldots \otimes 0 | \psi_N \rangle$ using the Hadamard test, Obtain:

$$\frac{\langle L_G \rangle}{||\rho(T_1)|| \cdot \ldots \cdot ||\rho(T_N)||} \pm 1/poly(n)$$

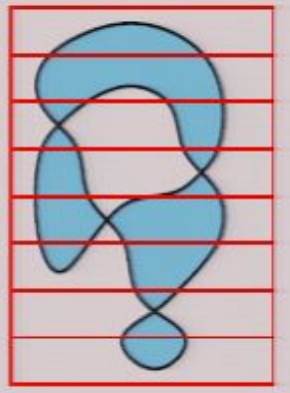
The approximation scale

$$\Delta_{alg} = ||\rho(T_1)|| \cdot \ldots \cdot ||\rho(T_N)||$$

Summary of quantum algorithm

- The computation starts with the state $|1\rangle$ (a path with 0 steps).
- Apply the elementary operators: minimas, maximas, crossings.
- In the end, we get

$$|\psi_N\rangle = \frac{\langle L_G \rangle}{||\rho(T_1)|| \cdot \ldots \cdot ||\rho(T_N)||} |1\rangle \otimes \overbrace{|0\rangle \otimes \ldots \otimes |0\rangle}^{\text{ancilla qubits}} + \ldots$$



- Estimate the inner product $\langle 1 \otimes 0 \otimes \ldots \otimes 0 | \psi_N \rangle$ using the Hadamard test, Obtain:

$$\frac{\langle L_G \rangle}{||\rho(T_1)|| \cdot \ldots \cdot ||\rho(T_N)||} \pm 1/poly(n)$$

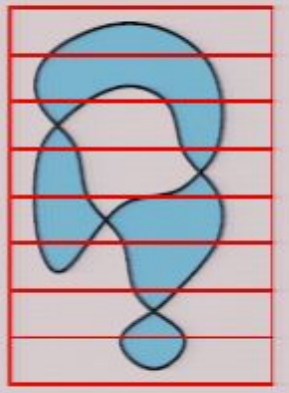
The approximation scale

$$\Delta_{alg} = ||\rho(T_1)|| \cdot \ldots \cdot ||\rho(T_N)||$$

Summary of quantum algorithm

- The computation starts with the state $|1\rangle$ (a path with 0 steps).
- Apply the elementary operators: minimas, maximas, crossings.
- In the end, we get

$$|\psi_N\rangle = \frac{\langle L_G \rangle}{||\rho(T_1)|| \cdot \ldots \cdot ||\rho(T_N)||} |1\rangle \otimes \overbrace{|0\rangle \otimes \ldots \otimes |0\rangle}^{\text{ancilla qubits}} + \ldots$$



- Estimate the inner product $\langle 1 \otimes 0 \otimes \ldots \otimes 0 | \psi_N \rangle$ using the Hadamard test, Obtain:

$$\frac{\langle L_G \rangle}{||\rho(T_1)|| \cdot \ldots \cdot ||\rho(T_N)||} \pm 1/poly(n)$$

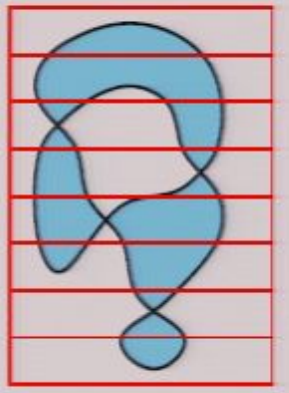
The approximation scale

$$\Delta_{alg} = ||\rho(T_1)|| \cdot \ldots \cdot ||\rho(T_N)||$$

Summary of quantum algorithm

- The computation starts with the state $|1\rangle$ (a path with 0 steps).
- Apply the elementary operators: minimas, maximas, crossings.
- In the end, we get

$$|\psi_N\rangle = \frac{\langle L_G \rangle}{||\rho(T_1)|| \cdot \ldots \cdot ||\rho(T_N)||} |1\rangle \otimes \overbrace{|0\rangle \otimes \ldots \otimes |0\rangle}^{\text{ancilla qubits}} + \ldots$$



- Estimate the inner product $\langle 1 \otimes 0 \otimes \ldots \otimes 0 | \psi_N \rangle$ using the Hadamard test, Obtain:

$$\frac{\langle L_G \rangle}{||\rho(T_1)|| \cdot \ldots \cdot ||\rho(T_N)||} \pm 1/poly(n)$$

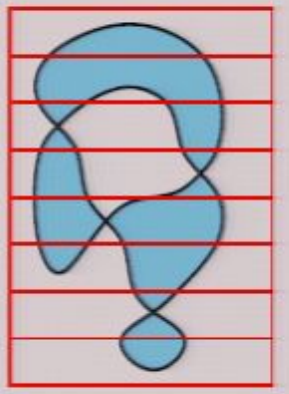
The approximation scale

$$\Delta_{alg} = ||\rho(T_1)|| \cdot \ldots \cdot ||\rho(T_N)||$$

Summary of quantum algorithm

- The computation starts with the state $|1\rangle$ (a path with 0 steps).
- Apply the elementary operators: minimas, maximas, crossings.
- In the end, we get

$$|\psi_N\rangle = \frac{\langle L_G \rangle}{||\rho(T_1)|| \cdot \ldots \cdot ||\rho(T_N)||} |1\rangle \otimes \overbrace{|0\rangle \otimes \ldots \otimes |0\rangle}^{\text{ancilla qubits}} + \ldots$$



- Estimate the inner product $\langle 1 \otimes 0 \otimes \ldots \otimes 0 | \psi_N \rangle$ using the Hadamard test, Obtain:

$$\frac{\langle L_G \rangle}{||\rho(T_1)|| \cdot \ldots \cdot ||\rho(T_N)||} \pm 1/poly(n)$$

The approximation scale

$$\Delta_{alg} = ||\rho(T_1)|| \cdot \ldots \cdot ||\rho(T_N)||$$

Conclusion of algorithmic part

Theorem I

There exists an efficient quantum algorithm that provides an additive approximation for the multivariate Tutte polynomial $Z_G(q, \mathbf{v})$ of any planar graph $G = (V, E)$ with any set of weights $\mathbf{v} = \{v_e\}$ and any complex q.

The approximation scale Δ_{alg} is given by

$$\Delta_{alg} = ||\rho(T_1)|| \cdot \ldots \cdot ||\rho(T_N)|| \, .$$

But is it non-trivial ?

- We prove non-triviality by proving BQP-hardness.
- This is already known for the special case of the Jones polynomial $\sim$ the unitary case.
- **Main difficulty:** how to prove universality for the non-unirary case?
- We generalize the universality proof of the Jones polynomial of Aharonov & Arad (2006), following the work of Freedman, Kitaev, Larsen & Wang.
- First time universialty is proved with non-unitary operators.

Conclusion of algorithmic part

Theorem I

There exists an efficient quantum algorithm that provides an additive approximation for the multivariate Tutte polynomial $Z_G(q, \mathbf{v})$ of any planar graph $G = (V, E)$ with any set of weights $\mathbf{v} = \{v_e\}$ and any complex q.

The approximation scale Δ_{alg} is given by

$$\Delta_{alg} = ||\rho(T_1)|| \cdot \ldots \cdot ||\rho(T_N)|| \, .$$

But is it non-trivial ?

- We prove non-triviality by proving BQP-hardness.
- This is already known for the special case of the Jones polynomial $\sim$ the unitary case.
- **Main difficulty:** how to prove universality for the non-unirary case?
- We generalize the universality proof of the Jones polynomial of Aharonov & Arad (2006), following the work of Freedman, Kitaev, Larsen & Wang.
- First time universialty is proved with non-unitary operators.

Welcome to the dark side of the talk. . .

Quantum Univerality

Universality

Given a quantum circuit $U = U_L \cdot \ldots \cdot U_1$ over n qubits, we need to decide whether $|\langle 0^{\otimes n}| U |0^{\otimes n}\rangle|^2 \leq 1/3$ or $|\langle 0^{\otimes n}| U |0^{\otimes n}\rangle|^2 \geq 2/3$.

We will construct a weighted graph $G = (V, E)$ such that

$$Z_G(q, \mathbf{v}) \simeq \Delta_{hard} \langle 0^{\otimes n}| U |0^{\otimes n}\rangle$$

Quantum Univerality

Universality

Given a quantum circuit $U = U_L \cdot \ldots \cdot U_1$ over n qubits, we need to decide whether $|\langle 0^{\otimes n}| U |0^{\otimes n}\rangle|^2 \leq 1/3$ or $|\langle 0^{\otimes n}| U |0^{\otimes n}\rangle|^2 \geq 2/3$.

We will construct a weighted graph $G = (V, E)$ such that

$$Z_G(q, \mathbf{v}) \simeq \Delta_{hard} \langle 0^{\otimes n}| U |0^{\otimes n}\rangle$$

Outline of the universality proof

Universality

$$Z_G(q, \mathbf{v}) \simeq \Delta_{hard} \left\langle 0^{\otimes n} \right| U \left| 0^{\otimes n} \right\rangle$$

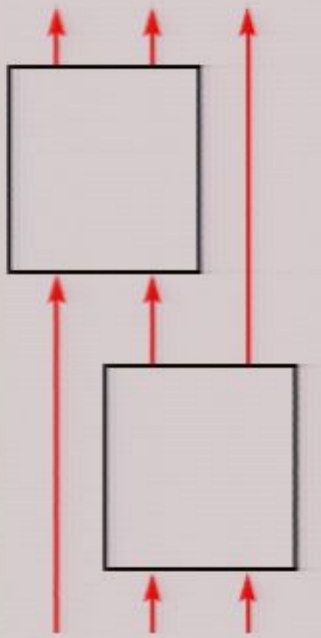
Finding G

Outline of the universality proof

Universality

$$Z_G(q, \mathbf{v}) \simeq \Delta_{hard} \left\langle 0^{\otimes n} \right| U \left| 0^{\otimes n} \right\rangle$$

Finding G



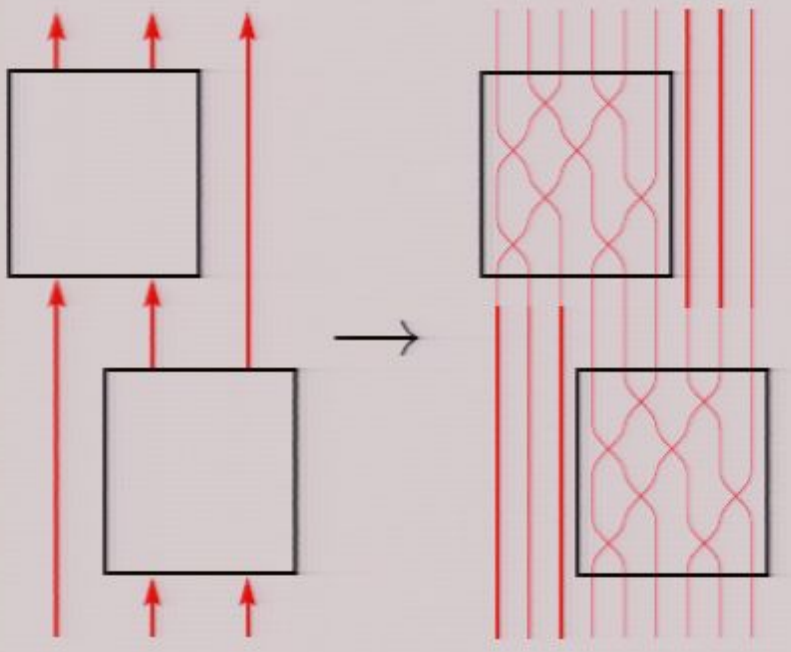
Quantum circuit
$U_N \cdot \ldots \cdot U_1$

Outline of the universality proof

Universality

$$Z_G(q, \mathbf{v}) \simeq \Delta_{hard} \left\langle 0^{\otimes n} \right| U \left| 0^{\otimes n} \right\rangle$$

Finding G



Quantum circuit
$U_N \cdot \ldots \cdot U_1$

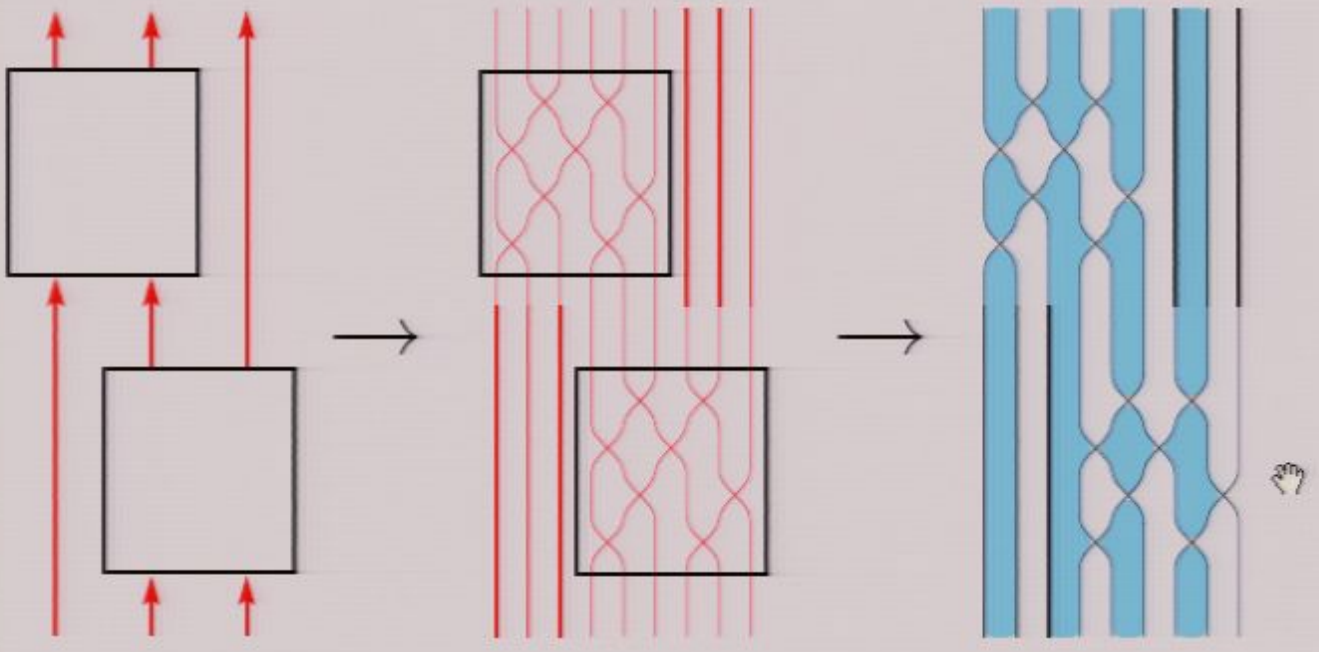
Encode as paths &
Approximate with
crossings

Outline of the universality proof

Universality

$$Z_G(q, \mathbf{v}) \simeq \Delta_{hard} \left\langle 0^{\otimes n} \right| U \left| 0^{\otimes n} \right\rangle$$

Finding G



Quantum circuit $U_N \cdot \ldots \cdot U_1$

Encode as paths & Approximate with crossings

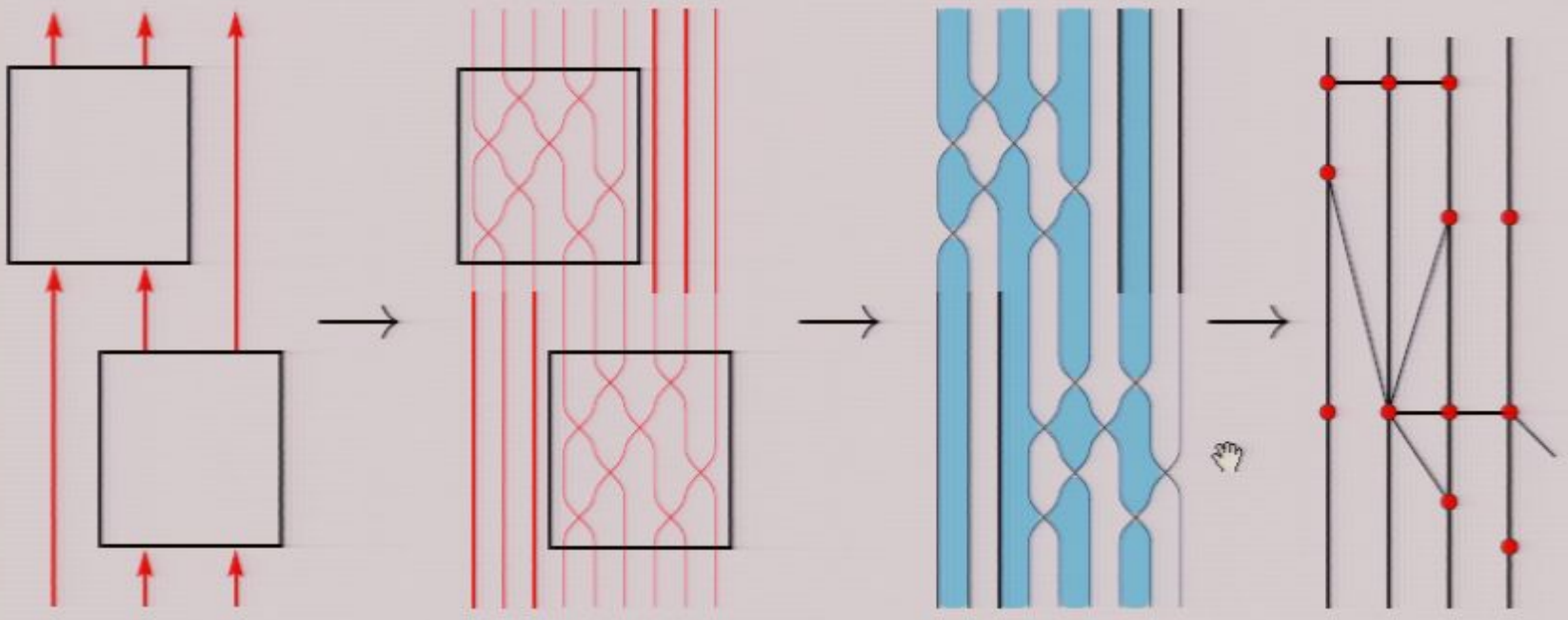
Medial graph L_G

Outline of the universality proof

Universality

$$Z_G(q, \mathbf{v}) \simeq \Delta_{hard} \left\langle 0^{\otimes n} \right| U \left| 0^{\otimes n} \right\rangle$$

Finding G



Quantum circuit $U_N \cdot \ldots \cdot U_1$

Encode as paths & Approximate with crossings

Medial graph L_G

Weighted graph G

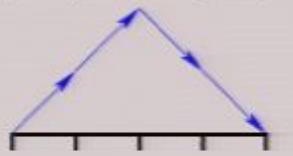
The 4-steps encoding (Kitaev, Wocjan & Yard)

Encoding qubits

Every qubit is mapped into a 4-steps path that starts at 1 and ends at 1:

$$|\underline{0}\rangle \stackrel{\text{def}}{=} |12121\rangle \ ,$$

$$|\underline{1}\rangle \stackrel{\text{def}}{=} |12321\rangle \ ,$$



String of n qubits is mapped to a $4n$-path in the subspace $L_{4n} \subseteq H_{4n}$.

The path space of 2 qubits - L_s

$|\underline{00}\rangle$ $|\underline{01}\rangle$ $|\underline{10}\rangle$ $|\underline{11}\rangle$

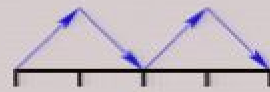
A 2-qubit gate is translated to a unitary operator over L_s

The 4-steps encoding (Kitaev, Wocjan & Yard)

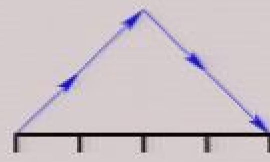
Encoding qubits

Every qubit is mapped into a 4-steps path that starts at 1 and ends at 1:

$|\underline{0}\rangle \stackrel{\text{def}}{=} |12121\rangle$,

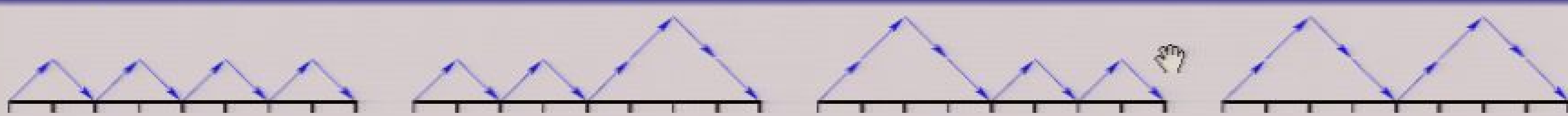


$|\underline{1}\rangle \stackrel{\text{def}}{=} |12321\rangle$,



String of n qubits is mapped to a $4n$-path in the subspace $L_{4n} \subseteq H_{4n}$.

The path space of 2 qubits - L_8



$|\underline{00}\rangle$ $|\underline{01}\rangle$ $|\underline{10}\rangle$ $|\underline{11}\rangle$

A 2-qubit gate is translated to a unitary operator over L_8

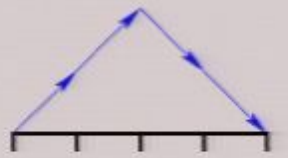
The 4-steps encoding (Kitaev, Wocjan & Yard)

Encoding qubits

Every qubit is mapped into a 4-steps path that starts at 1 and ends at 1:

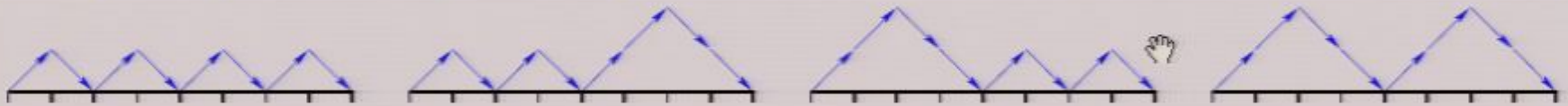
$$|\underline{0}\rangle \stackrel{\text{def}}{=} |12121\rangle \ ,$$

$$|\underline{1}\rangle \stackrel{\text{def}}{=} |12321\rangle \ ,$$



String of n qubits is mapped to a $4n$-path in the subspace $L_{4n} \subseteq H_{4n}$.

The path space of 2 qubits - L_8



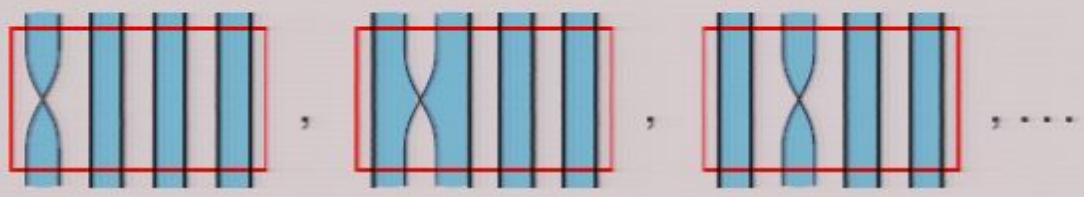
$|\underline{00}\rangle$ $|\underline{01}\rangle$ $|\underline{10}\rangle$ $|\underline{11}\rangle$

A 2-qubit gate is translated to a unitary operator over L_8

The 8-steps subspace

The crossing operators on L_8

7 crossing operators $\{\sigma_1, \sigma_2, \ldots, \sigma_7\}$ on 8 strands



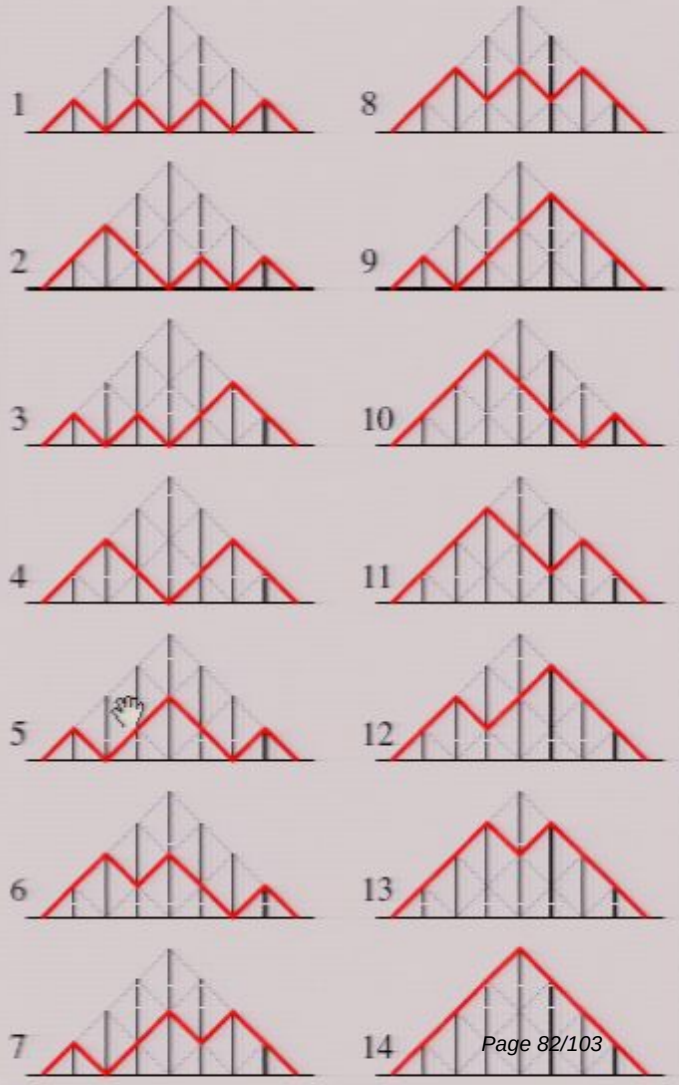
σ_1 σ_2 σ_3

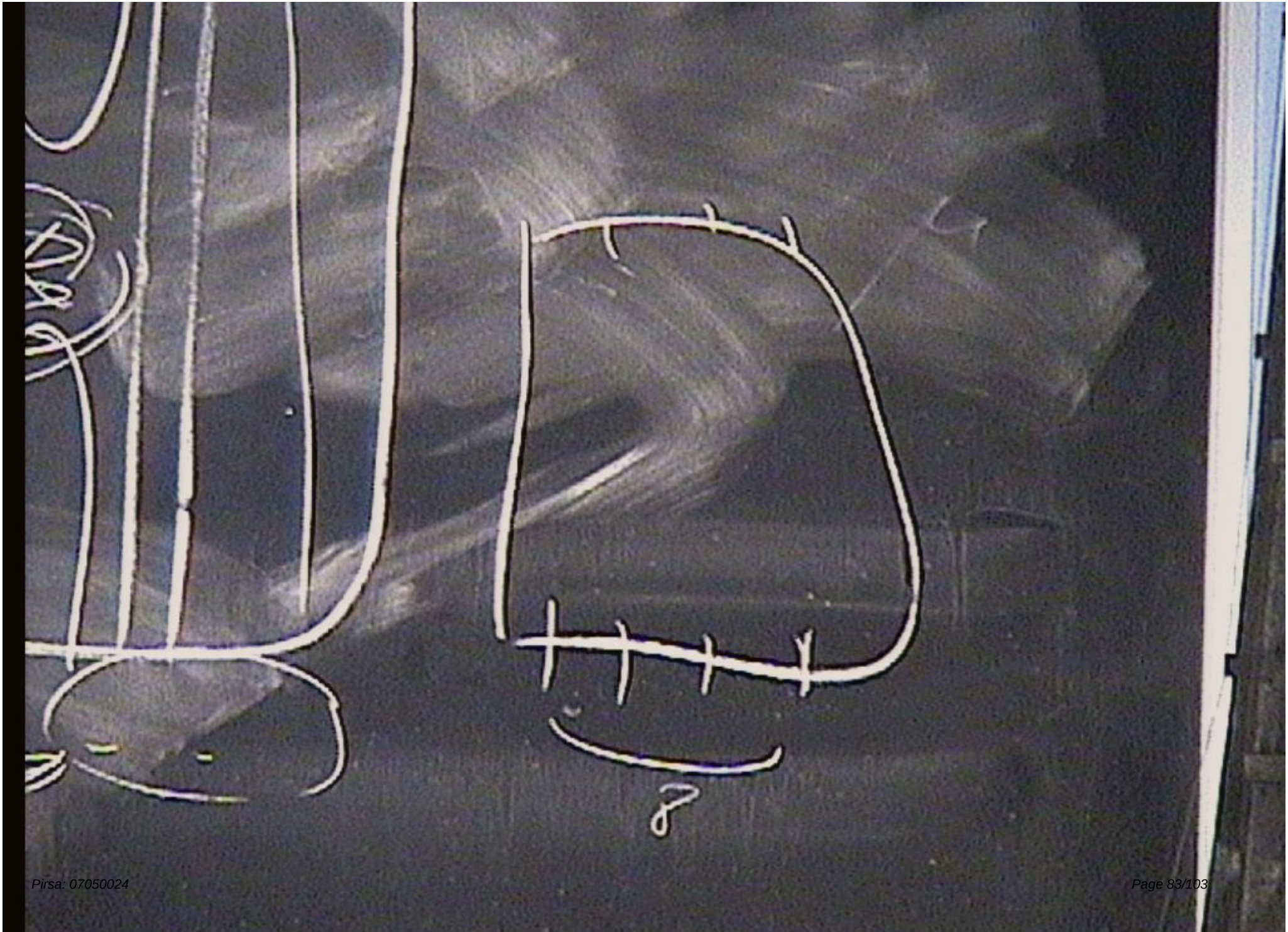
The smallest subspace that includes L_8 and is invariant to $\sigma_1, \ldots, \sigma_7$ is

$H_{8,1\to 1}$ = all 8-steps paths from $1 \to 1$

We need to show density in $SU(14)$ or $SL(14)$

$\dim(H_{8,1\to 1}) = 14$

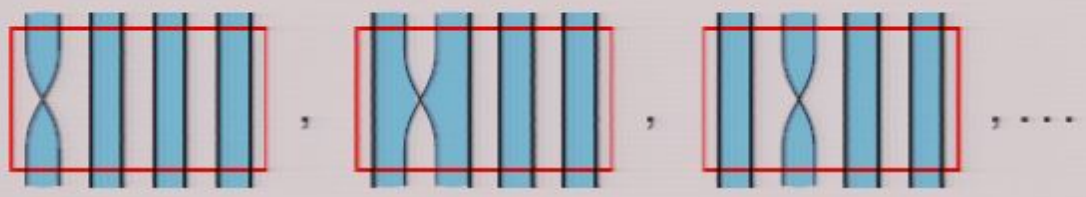




The 8-steps subspace

The crossing operators on L_8

7 crossing operators $\{\sigma_1, \sigma_2, \ldots, \sigma_7\}$ on 8 strands



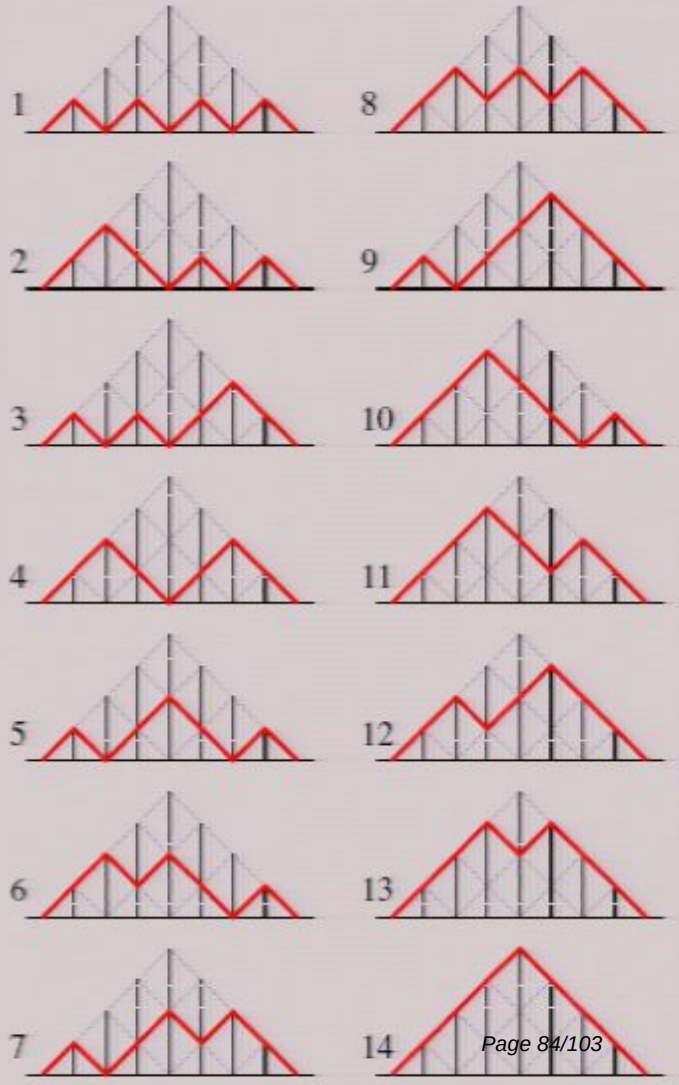
σ_1 σ_2 σ_3

The smallest subspace that includes L_8 and is invariant to $\sigma_1, \ldots, \sigma_7$ is

$H_{8,1\to 1}$ = all 8-steps paths from $1 \to 1$

We need to show density in $SU(14)$ or $SL(14)$

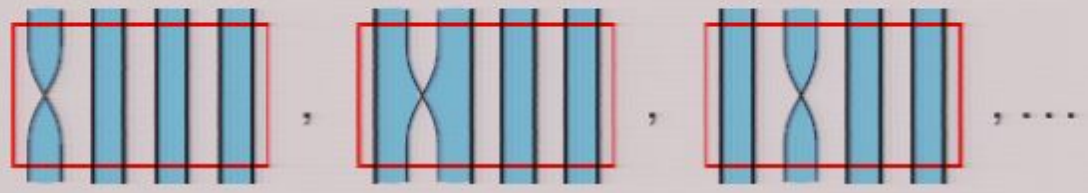
$\dim(H_{8,1\to 1}) = 14$



The 8-steps subspace

The crossing operators on L_8

7 crossing operators $\{\sigma_1, \sigma_2, \ldots, \sigma_7\}$ on 8 strands



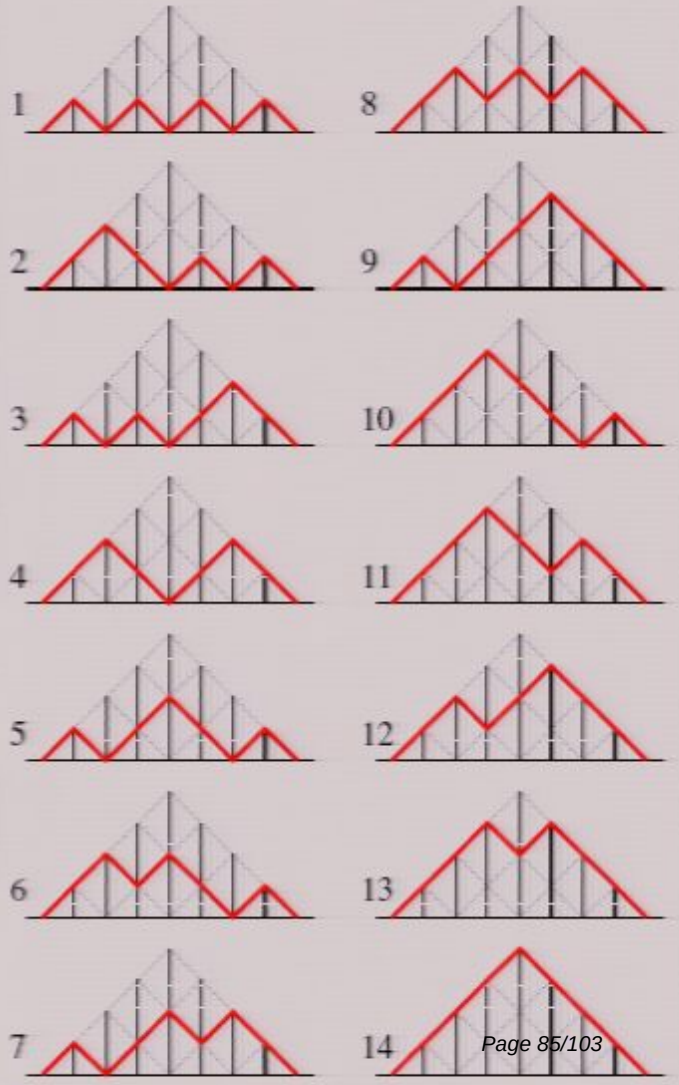
σ_1 σ_2 σ_3

The smallest subspace that includes L_8 and is invariant to $\sigma_1, \ldots, \sigma_7$ is

$H_{8,1\to 1}$ = all 8-steps paths from $1 \to 1$

We need to show density in $SU(14)$ or $SL(14)$

$\dim(H_{8,1\to 1}) = 14$



Building up the density - a tale of 3 lemmas

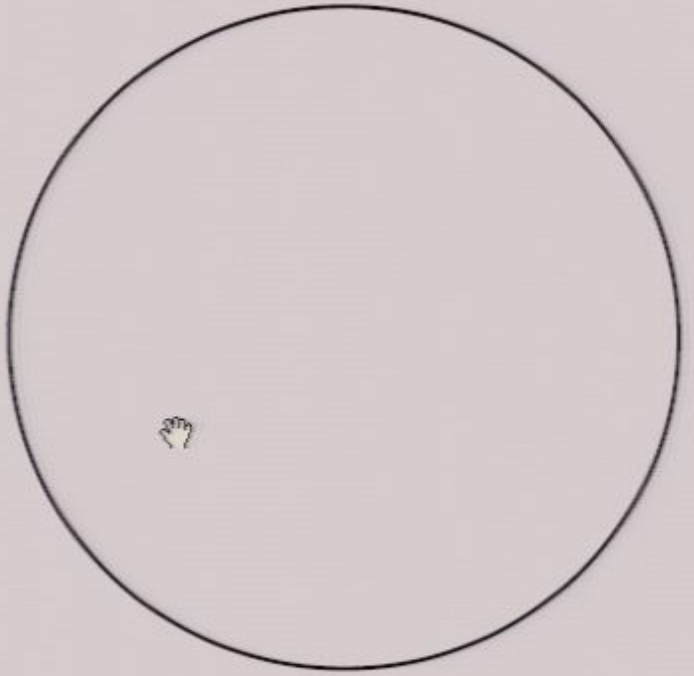
The Seeding Lemma

Generate a subgroup dense in $SU(2)$ or $SL(2)$ using σ_1, σ_2.

Operators

σ_1, σ_2

$SU(14)$ or $SL(14)$



Building up the density - a tale of 3 lemmas

The Seeding Lemma

Generate a subgroup dense in $SU(2)$ or $SL(2)$ using σ_1, σ_2.

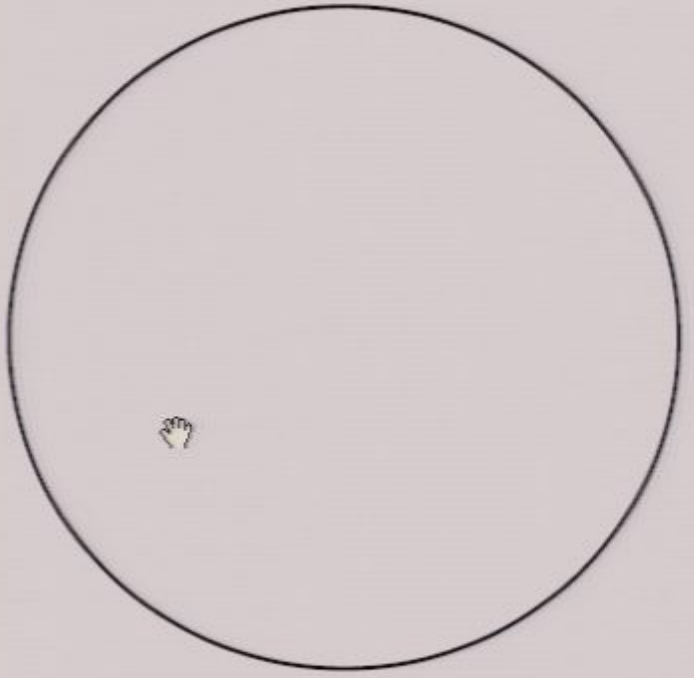
- **Unitary case:** $SU(2)$ is a compact group; density happens most of the times.
  Proof: Classification of finite groups in $SO(3)$
- **Non-unitary case:** $SL(2)$ is non-compact; density does not happen most of the times.
  Proof: Möbius transformations + Jørgensen inequality

Restricts $\mathbf{v}$ and q

Operators

σ_1, σ_2

$SU(14)$ or $SL(14)$



Building up the density - a tale of 3 lemmas

The Seeding Lemma

Generate a subgroup dense in $SU(2)$ or $SL(2)$ using σ_1, σ_2.

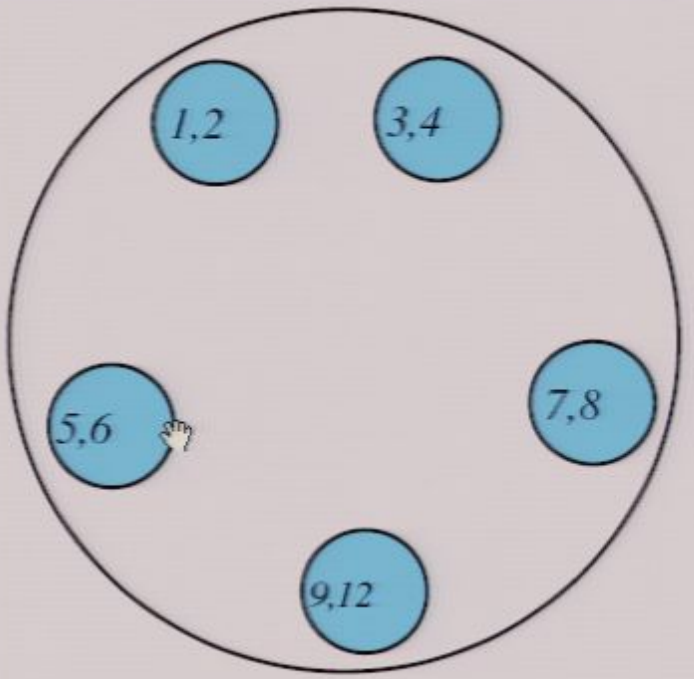
- **Unitary case:** $SU(2)$ is a compact group; density happens most of the times.
  Proof: Classification of finite groups in $SO(3)$
- **Non-unitary case:** $SL(2)$ is non-compact; density does not happen most of the times.
  Proof: Möbius transformations + Jørgensen inequality

Restricts $\mathbf{v}$ and q

Operators

σ_1, σ_2

$SU(14)$ or $SL(14)$



Building up the density - a tale of 3 lemmas

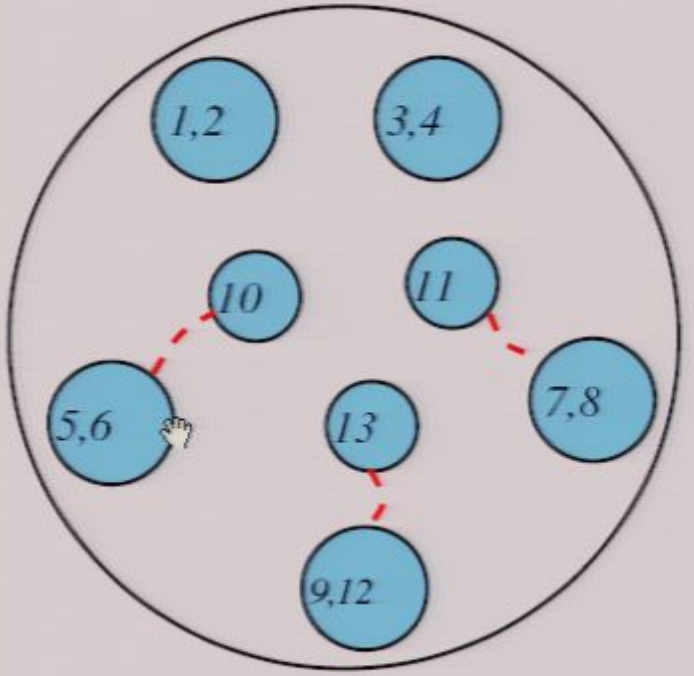
The Seeding Lemma

Generate a subgroup dense in $SU(2)$ or $SL(2)$ using σ_1, σ_2.

Operators

$\sigma_1, \sigma_2, \sigma_3$

$SU(14)$ or $SL(14)$



Building up the density - a tale of 3 lemmas

The Seeding Lemma

Generate a subgroup dense in $SU(2)$ or $SL(2)$ using σ_1, σ_2.

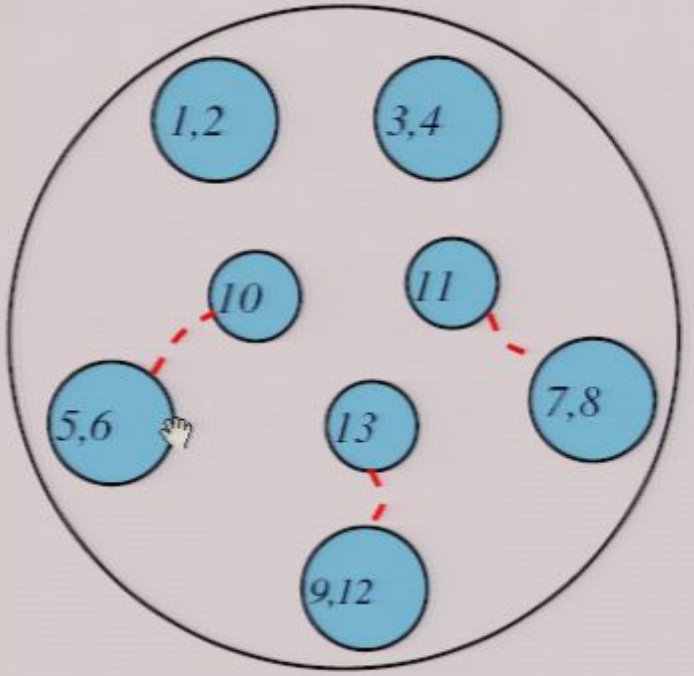
The Bridge Lemma

If we have density on $SU(A)$ and $SU(B)$ and a bridge $W : A \leftrightarrow B$, then we have density on $SU(A \oplus B)$.

Operators

$\sigma_1, \sigma_2, \sigma_3$

$SU(14)$ or $SL(14)$



Building up the density - a tale of 3 lemmas

The Seeding Lemma

Generate a subgroup dense in $SU(2)$ or $SL(2)$ using σ_1, σ_2.

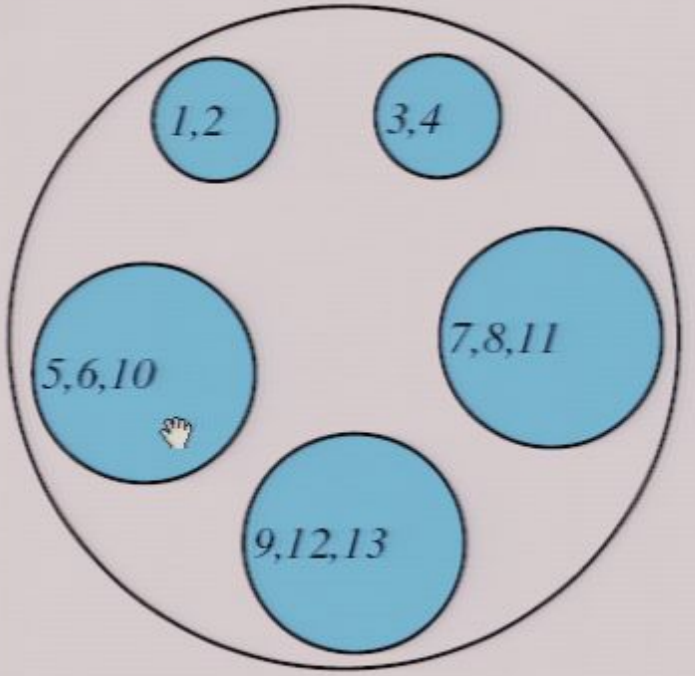
The Bridge Lemma

If we have density on $SU(A)$ and $SU(B)$ and a bridge $W : A \leftrightarrow B$, then we have density on $SU(A \oplus B)$.

Operators

$\sigma_1, \sigma_2, \sigma_3$

$SU(14)$ or $SL(14)$



Building up the density - a tale of 3 lemmas

The Seeding Lemma

Generate a subgroup dense in $SU(2)$ or $SL(2)$ using σ_1, σ_2.

The Bridge Lemma

If we have density on $SU(A)$ and $SU(B)$ and a bridge $W : A \leftrightarrow B$, then we have density on $SU(A \oplus B)$.

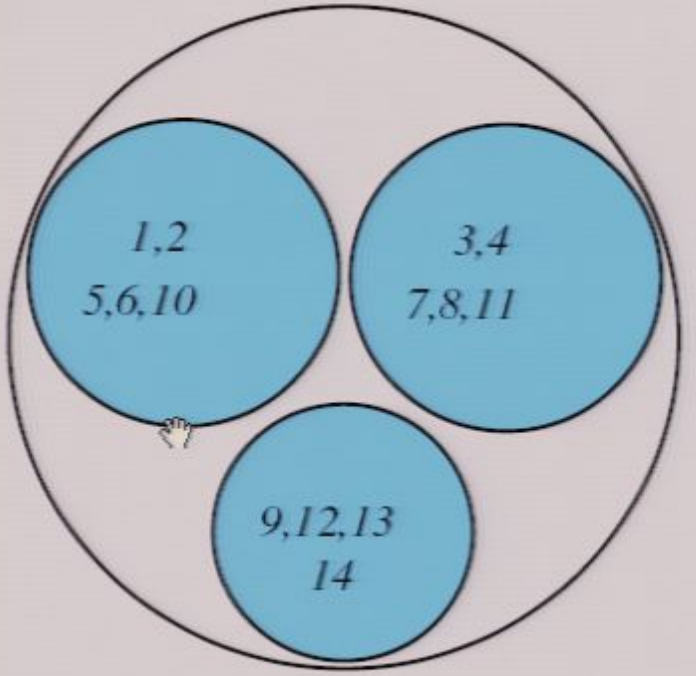
The Decoupling Lemma

If the same operators generate density on $SU(A)$ and $SU(B)$ and if $\dim A \neq \dim B$ then $SU(A)$ is decoupled from $SU(B)$.

Operators

$\sigma_1, \sigma_2, \sigma_3, \sigma_4$

$SU(14)$ or $SL(14)$



Building up the density - a tale of 3 lemmas

The Seeding Lemma

Generate a subgroup dense in $SU(2)$ or $SL(2)$ using σ_1, σ_2.

The Bridge Lemma

If we have density on $SU(A)$ and $SU(B)$ and a bridge $W : A \leftrightarrow B$, then we have density on $SU(A \oplus B)$.

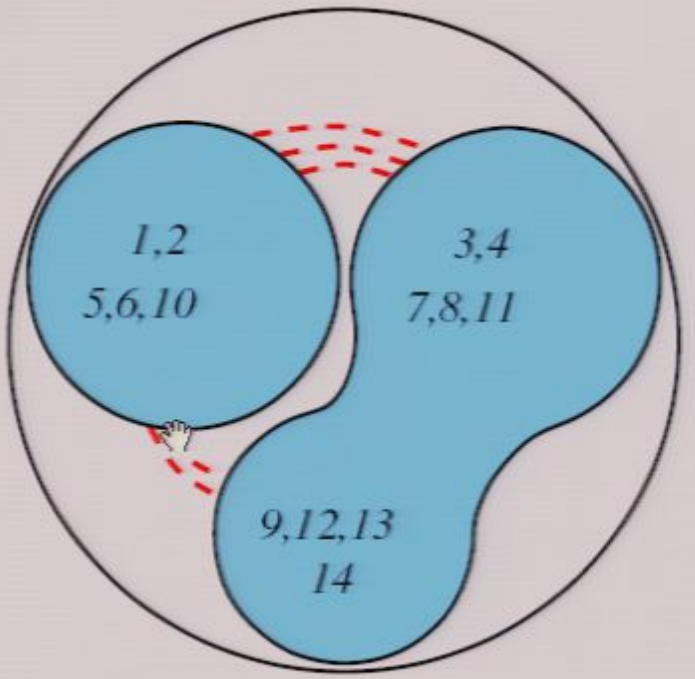
The Decoupling Lemma

If the same operators generate density on $SU(A)$ and $SU(B)$ and if $\dim A \neq \dim B$ then $SU(A)$ is decoupled from $SU(B)$.

Operators

$\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6$

$SU(14)$ or $SL(14)$



Building up the density - a tale of 3 lemmas

The Seeding Lemma

Generate a subgroup dense in $SU(2)$ or $SL(2)$ using σ_1, σ_2.

The Bridge Lemma

If we have density on $SU(A)$ and $SU(B)$ and a bridge $W : A \leftrightarrow B$, then we have density on $SU(A \oplus B)$.

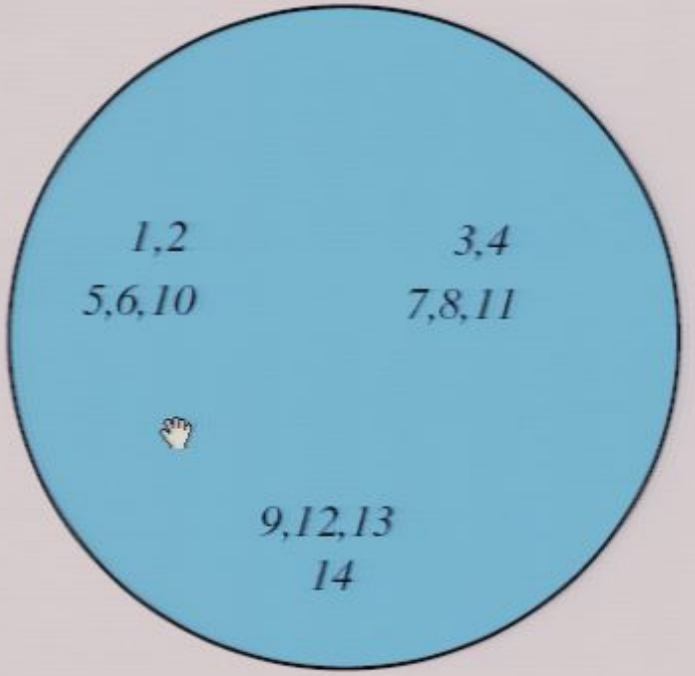
The Decoupling Lemma

If the same operators generate density on $SU(A)$ and $SU(B)$ and if $\dim A \neq \dim B$ then $SU(A)$ is decoupled from $SU(B)$.

Operators

$\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6$

$SU(14)$ or $SL(14)$



Building up the density - a tale of 3 lemmas

The Seeding Lemma

Generate a subgroup dense in $SU(2)$ or $SL(2)$ using σ_1, σ_2.

The Bridge Lemma

If we have density on $SU(A)$ and $SU(B)$ and a bridge $W : A \leftrightarrow B$, then we have density on $SU(A \oplus B)$.

The Decoupling Lemma

If the same operators generate density on $SU(A)$ and $SU(B)$ and if $\dim A \neq \dim B$ then $SU(A)$ is decoupled from $SU(B)$.

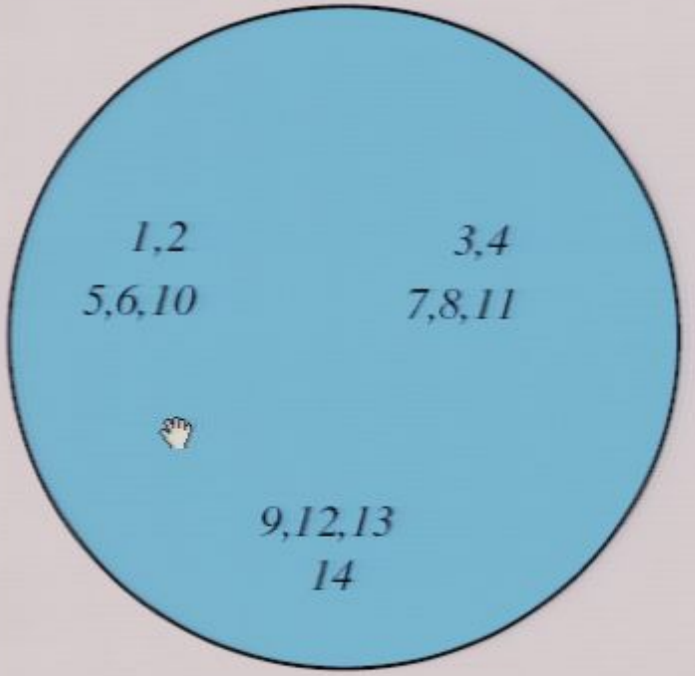
Density $\Longrightarrow$ efficiency

To finish up use the (non-unitary) Solovey-Kitaev theorem

Operators

$\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6$

$SU(14)$ or $SL(14)$



Summary of universality proof

Universality

$$Z_G(q, \mathbf{v}) \simeq \Delta_{hard} \left\langle 0^{\otimes n} \right| U \left| 0^{\otimes n} \right\rangle$$

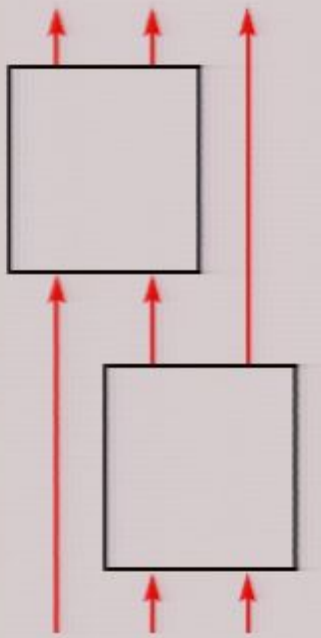
Finding G

Summary of universality proof

Universality

$$Z_G(q, \mathbf{v}) \simeq \Delta_{hard} \left\langle 0^{\otimes n} \right| U \left| 0^{\otimes n} \right\rangle$$

Finding G



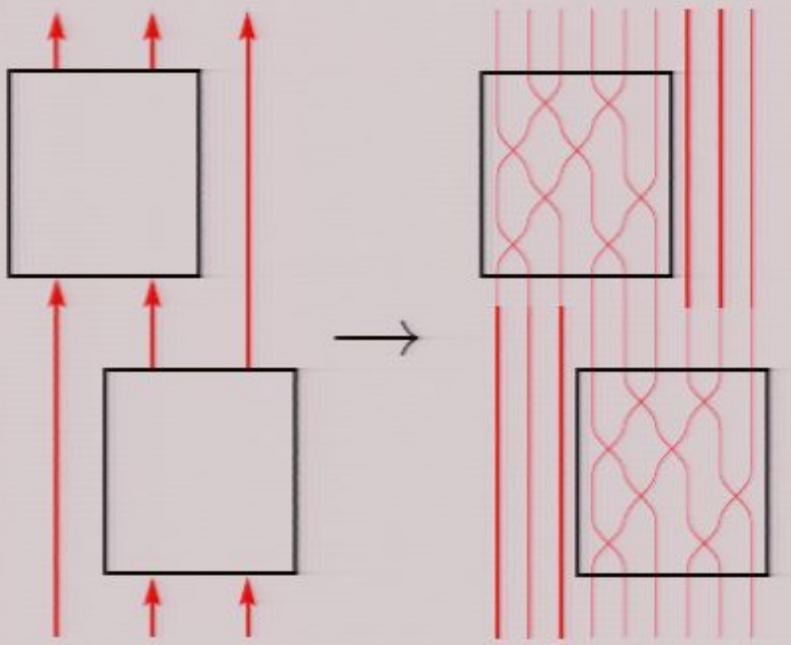
Quantum circuit
$U_N \cdot \ldots \cdot U_1$

Summary of universality proof

Universality

$$Z_G(q, \mathbf{v}) \simeq \Delta_{hard} \left\langle 0^{\otimes n} \right| U \left| 0^{\otimes n} \right\rangle$$

Finding G



Quantum circuit
$U_N \cdot \ldots \cdot U_1$

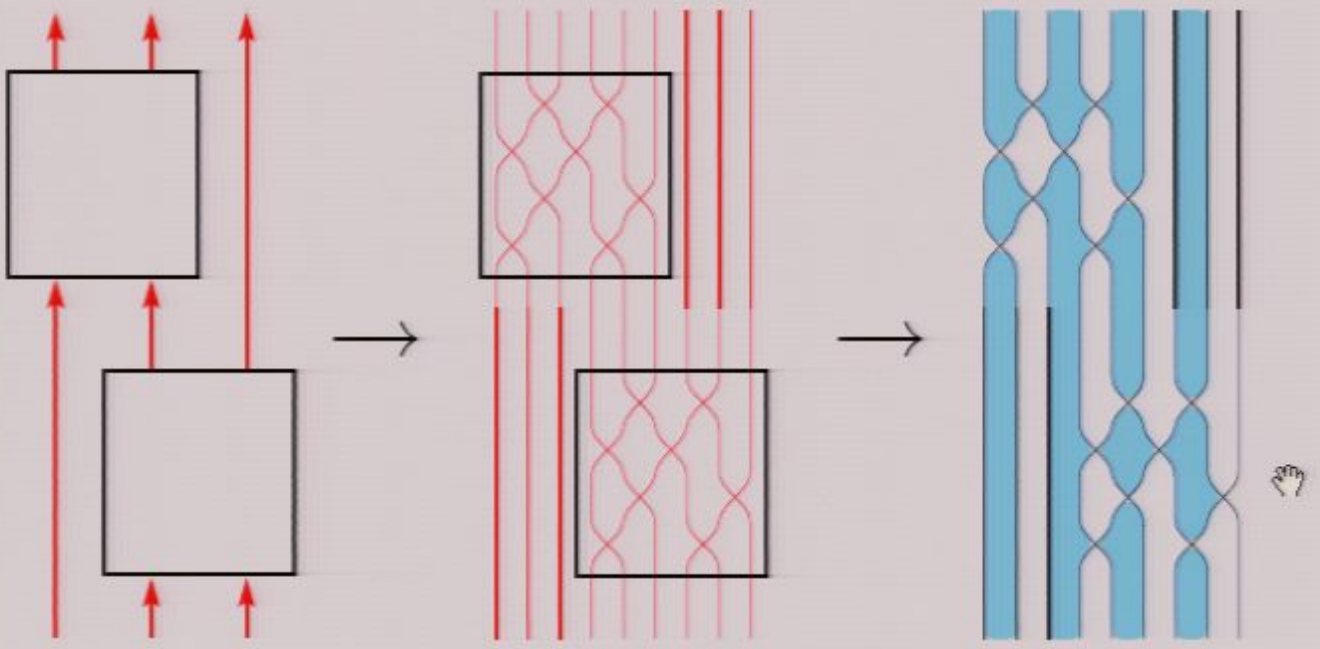
Encode as paths &
Approximate with
crossings

Summary of universality proof

Universality

$$Z_G(q, \mathbf{v}) \simeq \Delta_{hard} \left\langle 0^{\otimes n} \right| U \left| 0^{\otimes n} \right\rangle$$

Finding G



Quantum circuit $U_N \cdot \ldots \cdot U_1$

Encode as paths & Approximate with crossings

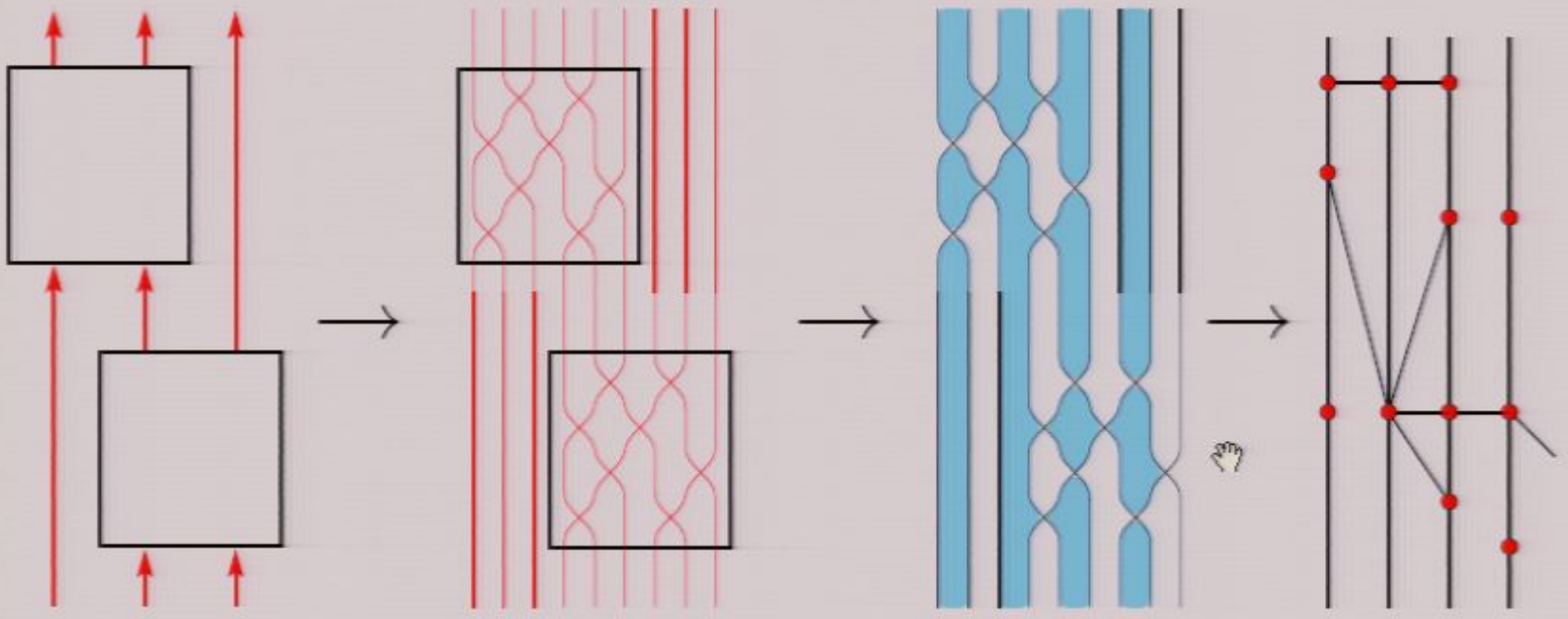
Medial graph L_G

Summary of universality proof

Universality

$$Z_G(q, \mathbf{v}) \simeq \Delta_{hard} \left\langle 0^{\otimes n} \right| U \left| 0^{\otimes n} \right\rangle$$

Finding G



Quantum circuit $U_N \cdot \ldots \cdot U_1$

Encode as paths & Approximate with crossings

Medial graph L_G

Weighted graph G

Summary and open questions

Summary

- An efficient quantum algorithm to approximate the Tutte polynomial for any planar graph and any set of weights. Uses a local mapping of the combinatorial problem into an algebra and implementing on a quantum computer using a representation. Many of the cases quantum-complete.
- **Breaking the unitarity barrier**
  - Quantum algorithm that uses non-unitary representations.
  - Quantum universality with non-unitary operators.
- **The Potts model:** possible implications to the long-standing problem of approximating the partition function of the q-state Potts model.

Open questions

- Universality does not apply for the Potts model; what is the complexity of the problem?
- Can it be generalized to non-planar graphs?
- Other combinatorial problems encoded by other algebras?
- Non-unitary universality for quantum: does it have any interesting implications?

$$|Y - f(x)| \leq \epsilon \cdot \Delta_x$$

$$\epsilon \neq \frac{1}{\text{Poly}(n)}$$