

# Classifying Complexity with the ZX Calculus

Alex Townsend-Teague<sup>0,1</sup> and Konstantinos Meichanetzidis<sup>1,2</sup>

<sup>0</sup> Mathematical Institute, University of Oxford

<sup>1</sup> Department of Computer Science, University of Oxford

<sup>2</sup> Cambridge Quantum Computing Ltd.

The ZX calculus is a graphical language which allows for reasoning about suitably represented tensor networks, aka ZX diagrams, in terms of rewrite rules. Here, we focus on problems which amount to exactly computing a scalar encoded as a closed tensor network. In general, such problems are #P-hard. However, there are families of such problems which are known to be in P when the dimension is below a certain value. By expressing problem-instances from these families as ZX diagrams, we see that the easy instances belong to the stabiliser fragment of the ZX calculus. Building on previous work on efficient simplification of qubit stabiliser diagrams, we present simplifying rewrites for the case of qutrits. Finally, we look at the specific examples of evaluating the Jones polynomial and of counting graph-colourings. Our exposition further constitutes the ZX calculus as a suitable and unifying language for studying the complexity of a broad range of classical or quantum problems.

## 1 Introduction

A plethora of hard problems of interest to physics and computer science regard interacting many-body multi-state systems, classical or quantum, and reduce to *exactly* computing a single scalar. These range from probability amplitudes in quantum mechanics, partition functions in classical statistical mechanics, counting problems, probabilistic inference, and many more. Problem instances can be encoded as a *closed* tensor-networks over an appropriate (semi)ring. The scalar can be evaluated by *full tensor contraction*, which in general is #P-hard [7]. In fact, finding the optimal contraction path for a general tensor network is NP-hard.

The ZX calculus is a graphical language whose origin lies in the field of quantum foundations [6]. It allows for reasoning about ZX diagrams, i.e. tensor networks which are expressed in terms of primitive generators [13]. The generators have a concrete representation as tensors over a (semi)ring and they obey a set of rewrite rules which respect semantics [12]. Importantly, the rewrite rules *depend* on the (semi)ring over which the diagram is interpreted as a tensor network. If a scalar of interest is expressed as a ZX diagram, one can rewrite the diagram by applying rewrite rules. One's goal is then to apply a sequence of rewrites and perform *full diagram simplification* in order to compute the desired scalar. Note that given an arbitrary tensor network, one can attempt to invent rewrite rules by inspecting the contents of the tensors and performing linear-algebra operations [10].

In the context of quantum computing, the Gottesman-Knill theorem states that stabiliser (or Clifford) circuits can be simulated *efficiently* on classical computers [1]. By simulation here we mean the *exact* computation of *amplitudes*. These circuits can be expressed by the *stabiliser* fragment of the ZX calculus. An alternative, and in fact more general, proof of the Gottesman-Knill theorem for qubits has been obtained graphically in terms of the qubit ZX calculus. Interestingly, even if the motivation for studying stabiliser diagrams stems from quantum computation, the result that stabiliser ZX diagrams can be simplified efficiently has broader applicability. This is because ZX diagrams can express arbitrary

linear maps; not every ZX diagram is a quantum circuit, but every quantum circuit can be cast as a ZX diagram.

Beyond quantum computing, ZX diagrams have been leveraged to study the complexity of boolean satisfiability (SAT) and model counting (#SAT) [5]. Model counting entails computing the number of satisfying assignments to a boolean formula (which is given in conjunctive normal form) and this number can be expressed a closed diagram. Specifically, one finds that in the case of XORSAT and #XORSAT, which are known to be tractable, the corresponding ZX diagrams representing the problems exactly correspond to qubit stabilizer diagrams, which are efficient to simplify. Going further, the ZH calculus [3], an extension of the ZX calculus, can be imported to make graphically obvious why 2SAT is in P but #2SAT is #P-complete while 3SAT is NP-complete and #3SAT is #P-hard. That is, the counting version of 2SAT is hard while deciding is easy, but this is not the case for 3SAT where both deciding and counting are hard. The *key realisation* is in that the decision problem is a counting problem but where the diagram is interpreted over the Boolean semiring. Then the *rewrite rules change accordingly* and obviously imply the above result by allowing efficient simplification strategies.

Such observations make apparent the *unifying power of diagrammatic reasoning for studying the complexity* of problem families of universal interest, where the degrees of freedom are two-dimensional. Building on the spirit of [5], we treat ZX as a library comprising out-of-the-box and readily-available diagrammatic rewrite-rules from which we import the appropriate tools for the job depending on the occasion. The key rewrite rules that enable the efficient diagram simplification of qubit stabiliser ZX diagrams are called *local complementation* and *pivot*, the latter being composition of three of the former [8]. In this work, we recall the qutrit version of local complementation and derive the corresponding pivot rule which implies that qutrit stabiliser diagrams can be simplified efficiently. We then present two case studies: evaluating the Jones polynomial at lattice roots of unity, and graph colouring. Both of these problem families reduce to evaluating closed tensor networks and show a transition in complexity at a particular dimension, below which the tensor network corresponds to a stabiliser ZX diagram.

We underline that throughout this work, all rewrites are valid *up to scalar* since we are only concerned about making statements about complexity.

## 2 Simplifying Qubit ZX-Diagrams

In this section we briefly review the ZX calculus and recalling the definitions of a graph-like diagrams and stabiliser diagrams. Crucially, we also recall the simplifying rewrites that enable the efficient simplification of stabiliser diagrams.

### 2.1 The Qubit ZX-Calculus

Here we give a quick introduction to the qubit ZX-calculus. The qubit ZX-calculus is a diagrammatic language for quantum mechanics generated by *spiders*. The spider legs, or *wires*, represent vector spaces of dimension  $d = 2$ . Diagrams are read bottom-to-top; bottom open wires (not connected to anything) are *input wires* and top ones are *output*. Diagrams with only outputs are called *states* and with only inputs are called *effects*. A *closed* diagram is one with no input nor output wires and it represents a scalar.

Spiders can be *composed*; output wires can be connected to input wires to form spider networks which are again valid ZX diagrams. Placing diagrams side by side represents parallel composition ( $\otimes$ ), with concrete interpretation as the tensor product. Vertically stacking diagrams corresponds to sequential composition ( $\circ$ ) and concretely it means matrix multiplication. Specifically, it means tensor contraction

of two spider tensors along the wire connecting them, i.e. the common tensor index represented by this wire is summed over. The concrete representation of these operations of course depends on the (semi)ring over which the spiders are interpreted as tensors. For spiders  $S$  and  $S'$ , we denote these as

$$\llbracket S \otimes S' \rrbracket = \llbracket S \rrbracket \otimes \llbracket S' \rrbracket, \quad \llbracket S \circ S' \rrbracket = \llbracket S \rrbracket \cdot \llbracket S' \rrbracket \quad (1)$$

Spiders come in two species: green  $Z$ -spiders and red  $X$ -spiders, decorated by a *phase*  $\alpha \in [0, 2\pi)$ . When  $\alpha = 0$ , we will omit it. The concrete representation of the spider generators as tensors over  $\mathbb{C}$  is:

$$\left[ \begin{array}{c} m \\ \vdots \\ \text{green spider } \alpha \\ \vdots \\ n \end{array} \right] = |0\rangle^{\otimes m} \langle 0|^{\otimes n} + e^{i\alpha} |1\rangle^{\otimes m} \langle 1|^{\otimes n}, \quad \left[ \begin{array}{c} m \\ \vdots \\ \text{red spider } \alpha \\ \vdots \\ n \end{array} \right] = |+\rangle^{\otimes m} \langle +|^{\otimes n} + e^{i\alpha} |-\rangle^{\otimes m} \langle -|^{\otimes n},$$

where  $\{|0\rangle, |1\rangle\}$  is the  $Z$ -basis and  $|\pm\rangle = |0\rangle \pm |1\rangle$  the  $X$ -basis in  $\mathbb{C}^2$ , in Dirac notation. The Hadamard gate  $H$ , whose function is to switch between the  $Z$  and  $X$  bases, is denoted as a yellow box. Often we will instead draw a dashed blue line to represent a *Hadamard edge*:

$$\begin{array}{c} | \\ \text{yellow box} \\ | \end{array} = \begin{array}{c} | \\ \text{dashed blue line} \\ | \end{array} = \begin{array}{c} \text{green spider } \frac{\pi}{2} \\ \text{red spider } \frac{\pi}{2} \\ \text{green spider } \frac{\pi}{2} \end{array}, \quad \left[ \begin{array}{c} | \\ \text{yellow box} \\ | \end{array} \right] \simeq |0\rangle \langle 0| + |0\rangle \langle 1| + |1\rangle \langle 0| - |1\rangle \langle 1| \quad (2)$$

The ZX calculus is universal for multilinear maps; any tensor over a (semi)ring has a corresponding ZX diagram. The rewrite rules of the ZX calculus (see Fig.?? in ??) allow manipulation of the diagrams by *local tensor-rewrites*. Importantly, the rewrites *preserve the semantics*, i.e. the concrete tensor representation over  $\mathbb{C}$ , up to a scalar. The ZX calculus is complete in the sense that any true equation between tensors can be proven *only* in terms of rewrites; the diagram on the left hand side of the equation can be transformed to that on the right hand side but applying rewrite rules. This gives ZX the status of a ‘calculus’. What is important about qubit ZX diagrams is that *only topology matters*; the concrete tensor semantics of the diagram is invariant under deformations of the network as long as the inter-spider connectivity is respected.

## 2.2 Stabilizer Qubit ZX Diagrams

The *stabiliser fragment* of the calculus consists of all diagrams in which all phases are  $a = \kappa \frac{\pi}{2}$ ,  $\kappa \in \mathbb{Z}$ . In (Theorem 5.4 [8]) the authors give an efficient algorithm for simplifying any *qubit* ZX-diagram to an equivalent diagram with fewer spiders. The algorithm consists of consecutive applications of spider-eliminating rewrites.

First it is shown that every diagram is equivalent to a *graph-like* diagram: every spider is green, every edge is a Hadamard edge, there are no parallel edges or self-loops, every input and output wire is connected to a spider and every spider has at most one input or output wire. Then the following two rewrite rules, derived via *local complementation* and *pivoting*, can be used to eliminate spiders:

$$\begin{array}{c} \text{green spider } \alpha_1 \text{ --- } \text{green spider } \pm \frac{\pi}{2} \text{ --- } \text{green spider } \alpha_n \\ \vdots \quad \quad \quad \vdots \quad \quad \quad \vdots \\ \text{green spider } \alpha_2 \text{ --- } \text{green spider } \alpha_{n-1} \end{array} = \begin{array}{c} \text{green spider } \alpha_1 \mp \frac{\pi}{2} \text{ --- } \text{green spider } \alpha_n \mp \frac{\pi}{2} \\ \vdots \quad \quad \quad \vdots \\ \text{green spider } \alpha_2 \mp \frac{\pi}{2} \text{ --- } \text{green spider } \alpha_{n-1} \mp \frac{\pi}{2} \end{array} \quad (3)$$

$$(4)$$

For more details, see Section 4 [8]. Eq.3 says that we can remove any spider with phase  $\pm \frac{\pi}{2}$  at the cost of performing a local complementation at said spider. Furthermore, Eq.4 says we can remove any pair of spiders with phases in  $\{0, \pi\}$  connected by a Hadamard edge at the cost of performing a local pivot along said edge. After each application of (3) or (4), the following rewrite rule, aka the Hopf rule which can be derived from the qubit ZX rules, can be used to remove any parallel Hadamard edges and ensure the diagram remains graph-like: **what about self-loops? also, make the edges blue dashed.**

$$(5)$$

In particular, and importantly to this work, given a closed stabilizer diagram the algorithm *efficiently simplifies* it until it contains at most one spider. If it exists, this spider is green, legless and has phase in  $\{0, \pi\}$ . The point relevant to this work is that spiders can be *eliminated efficiently*, i.e. with a polynomial number of rewrites in the initial number of spiders. Note that every spider elimination introduces a polynomial number of edges in the diagram, which prevents an overwhelming memory cost of the simplification procedure.

### 3 Simplifying Qutrit ZX-Diagrams

We now turn to the qutrit ZX-calculus and examine the analogous story to that of the previous subsection but now for the case where the dimension of the vector space carried by the wires is  $d = 3$ .

#### 3.1 The Qutrit ZX-Calculus

As in the qubit case, the qutrit ZX calculus is about spiders connected by wires, but there are key differences, some subtler than others. Again, spiders come in two species, Z (green) and X (red), with the three-dimensional Z basis denoted as  $\{|0\rangle, |1\rangle, |2\rangle\}$ . Let  $\omega = e^{i\frac{2\pi}{3}}$  denote the third root of unity with  $\bar{\omega} = \omega^2$  its complex conjugate. The qutrit X-basis consists of the three vectors:

$$|+\rangle = \frac{1}{\sqrt{3}}(|0\rangle + |1\rangle + |2\rangle), \quad |\omega\rangle = \frac{1}{\sqrt{3}}(|0\rangle + \omega|1\rangle + \bar{\omega}|2\rangle), \quad |\bar{\omega}\rangle = \frac{1}{\sqrt{3}}(|0\rangle + \bar{\omega}|1\rangle + \omega|2\rangle) \quad (6)$$

$$\left[ \left[ \begin{array}{c} \overbrace{\quad\quad\quad}^m \\ \vdots \\ \textcircled{\alpha \atop \beta} \\ \vdots \\ \underbrace{\quad\quad\quad}_n \end{array} \right] \right] = |0\rangle^{\otimes m} \langle 0|^{\otimes n} + e^{i\alpha} |1\rangle^{\otimes m} \langle 1|^{\otimes n} + e^{i\beta} |2\rangle^{\otimes m} \langle 2|^{\otimes n} \quad (7)$$

$$\left[ \begin{array}{c} \overbrace{\quad\quad\quad}^m \\ \vdots \\ \textcircled{\alpha \atop \beta} \\ \vdots \\ \underbrace{\quad\quad\quad}_n \end{array} \right] = |+\rangle^{\otimes m} \langle +|^{\otimes n} + e^{i\alpha} |\omega\rangle^{\otimes m} \langle \omega|^{\otimes n} + e^{i\beta} |\bar{\omega}\rangle^{\otimes m} \langle \bar{\omega}|^{\otimes n} \quad (8)$$

Hadamard boxes are now neither self-conjugate nor self-adjoint, so we change our notation: we let a yellow parallelogram decorated with a 1 (mod 3) denote a Hadamard gate, while decorating with a 2 (mod 3) denotes its adjoint. We will shortly explain this choice. We also use a dashed blue line for the *Hadamard edge* ( $H$ -edge) and a purple dashed line for its adjoint ( $H^\dagger$ -edge):

$$\begin{array}{c} \text{triangle} \end{array} = \begin{array}{c} \text{triangle} \\ \text{triangle} \\ \text{triangle} \end{array}, \quad \begin{array}{c} \text{triangle} \end{array} = \begin{array}{c} \text{triangle} \\ \text{triangle} \\ \text{triangle} \end{array} \quad (9)$$

$$\begin{array}{c} \text{---} \text{---} \text{---} \end{array} \neq \begin{array}{c} \text{---} \text{---} \text{---} \end{array}, \quad \begin{array}{c} \text{---} \text{---} \text{---} \end{array} \neq \begin{array}{c} \text{---} \text{---} \text{---} \end{array} \quad (10)$$
$$\begin{array}{c} \vdots \\ \vdots \\ \alpha \\ \beta \\ \vdots \end{array} \begin{array}{c} \vdots \\ \vdots \\ \gamma \\ \delta \\ \vdots \end{array} \begin{array}{c} \vdots \\ \vdots \\ \vdots \\ \vdots \end{array} = \begin{array}{c} \vdots \\ \vdots \\ \alpha \\ \beta \\ \vdots \end{array} \begin{array}{c} \vdots \\ \vdots \\ \gamma \\ \delta \\ \vdots \end{array} \begin{array}{c} \vdots \\ \vdots \\ \vdots \\ \vdots \end{array}, \quad \begin{array}{c} \vdots \\ \vdots \\ \alpha \\ \beta \\ \vdots \end{array} \begin{array}{c} \vdots \\ \vdots \\ \gamma \\ \delta \\ \vdots \end{array} \begin{array}{c} \vdots \\ \vdots \\ \vdots \\ \vdots \end{array} = \begin{array}{c} \vdots \\ \vdots \\ \alpha \\ \beta \\ \vdots \end{array} \begin{array}{c} \vdots \\ \vdots \\ \gamma \\ \delta \\ \vdots \end{array} \begin{array}{c} \vdots \\ \vdots \\ \vdots \\ \vdots \end{array} \quad (11)$$

Secondly, the ‘snake equations’ hold for *same-colour* cups and caps. Snakes with differently coloured cups and caps can still be yanked into a straight wire at the cost of adding two Hadamard boxes, shown below for  $h \in \{1, 2\}$ :

$$\text{green cap} \text{---} \text{green cup} = |, \quad \text{green cap} \text{---} \text{red cup} = \text{triangle} \text{---} \text{triangle} \quad (12)$$

The above equations hold also if we flip the colours, green  $\leftrightarrow$  red.

### 3.2 Graph-Like Qutrit ZX Diagrams

A *graph-like qutrit ZX diagram* is one where every spider is green, spiders are only connected Hadamard edges (blue) or their adjoints (purple), every pair of spiders is connected by at most one  $H$ -edge or  $H^\dagger$ -edge, every input and output wire is connected to a spider, and every spider is connected to at most one input or output wire. A graph-like qutrit ZX diagram is a *graph state* when every spider has zero phase (top and bottom) and is connected to an output.

Note the difference compared to the qubit case: we need not worry about self-loops because the qutrit ZX calculus doesn’t define a ‘plain’ cap or cup. But this comes at a cost: spiders in the qutrit case fuse more fussily. Specifically, when two spiders of the same colour are connected by at least one plain edge and at least one  $H$ - or  $H^\dagger$ -edge, fusion is not possible. The following equation, for  $h \in \{1, 2\}$ , helps us get around this:

$$\text{green spider } \alpha/\beta \text{ --- blue edge --- green spider } \gamma/\delta \text{ --- yellow triangle} \stackrel{(H)}{=} \text{green spider } \alpha/\beta \text{ --- yellow triangle --- green spider } \gamma/\delta \text{ --- yellow triangle} \stackrel{(id)}{=} \text{green spider } \alpha/\beta \text{ --- yellow triangle --- green spider } \gamma/\delta \text{ --- yellow triangle} \quad (13)$$

We will shortly show that every qutrit ZX-diagram is equivalent to a graph-like one, making use of the following lemmas:

**Lemma 3.1.** The following equation holds in the qutrit ZX-calculus:

$$\text{green spider } \alpha/\beta \text{ --- blue edge --- green spider } \gamma/\delta \stackrel{=}{=} \text{green spider } \alpha/\beta \text{ --- green spider } \gamma/\delta \stackrel{=}{=} \text{green spider } \alpha/\beta \text{ --- green spider } \gamma/\delta \quad (14)$$

*Proof.* It is shown in Lemma 2.8 [9] that the qutrit ZX-calculus satisfies the following ‘Hopf law’:

$$\text{green spider } \alpha/\beta \text{ --- blue edge --- red spider } \gamma/\delta \stackrel{=}{=} \text{green spider } \alpha/\beta \text{ --- red spider } \gamma/\delta \quad (15)$$

Therefore we can argue as follows, for  $h \in \{1, 2\}$ :

$$(16)$$

□

**Lemma 3.2.** The following two equations hold in the qutrit ZX-calculus:

$$(17)$$

*Proof.* This is Lemma 3.4 in [9].

□

As we will formalise later, the lemmas above justify our notation for Hadamard gates: we can think of Hadamard edges as 1-weighted edges and their adjoints as 2-weighted edges, then work modulo 3, since every triple of parallel edges disappears. Where the previous lemmas relate single  $H$ - and  $H^\dagger$ -boxes across multiple edges, the next relates multiple  $H$ - and  $H^\dagger$ -boxes on single edges.

**Lemma 3.3.** The following three equations hold in the qutrit ZX calculus, for  $h \in \{1, 2\}$ :

$$(18)$$

*Proof.* For the first equation, negate the top two boxes via (s), then cancel twice via (H). Similarly, for the second equation, negate the top two boxes via (s), then cancel once via (H). The third equation is just a simple application of (id). □

Again, intuitively we can think of Hadamard boxes of having value 1 and their adjoints  $-1$  and then work modulo 4.

**Corollary 3.4.** Every qutrit ZX diagram is equivalent to one that is graph-like.

*Proof.* First use the colour change rule to turn all X-spiders into Z-spiders. Then use Lemma 3.3 to remove excess  $H$ - and  $H^\dagger$ -boxes, inserting a spider between any remaining consecutive pair of such boxes, so that all spiders are connected only by plain edges,  $H$ -edges or  $H^\dagger$ -edges. Fuse together as many as possible, and apply (13) where fusion is not possible, so that no plain edge connects two spiders. Apply Lemmas 3.1 and 3.2 to all connected pairs of spiders until at most one  $H$ - or  $H^\dagger$ -edge remains between them. Finally, to ensure every input and output is connected to a spider and every spider is

connected to at most one input or output, we can use **(H)** and **(id)** to add a few spiders,  $H$ - and  $H^\dagger$ -boxes as needed:

$$\begin{array}{c} | \\ \text{---} \end{array} = \begin{array}{c} \text{---} \\ \text{---} \end{array}, \quad \begin{array}{c} \text{---} \\ \text{---} \end{array} = \begin{array}{c} \text{---} \\ \text{---} \end{array}, \quad \begin{array}{c} \text{---} \\ \text{---} \end{array} = \begin{array}{c} \text{---} \\ \text{---} \end{array} \quad (19)$$

□

A graph state is described fully by its underlying multigraph, or equivalently by an adjacency matrix, where edges take weights in  $\mathbb{Z}_3$  [Lemma 4.2 [11]]. Nodes correspond to phaseless green spiders, edges of weight 1 correspond to Hadamard edges, and edges of weight 2 correspond to  $H^\dagger$  edges. As in the qubit case, graph states admit a local complementation operation Definition 2.6 [11], though the effect is now slightly more complicated. We'll give the intuition after the formal definition:

**Definition 3.5.** Given  $a \in \mathbb{Z}_3$  and a graph state  $G$  with adjacency matrix  $W = (w_{i,j})$ , the  $a$ -local complementation at node  $k$  is the new graph state  $G *_a k$ , whose adjacency matrix  $W' = (w'_{i,j})$  given by:

$$w'_{i,j} = w_{i,j} + aw_{i,k}w_{j,k} \quad (20)$$

So only those edges between neighbours of node  $k$  are affected, but rather than just having their weight increased by 1 (modulo 2) as in the qubit case, the increase in weight also depends on the weights of the edges from  $i$  and  $j$  to  $k$ . As always, this is best seen graphically. For two nodes  $i$  and  $j$  both connected to  $k$  by the same colour edge,  $a$ -local complementation at  $k$  increases weight  $w_{i,j}$  by  $a$ . If instead  $i$  and  $j$  are connected to  $k$  by edges of different colour,  $a$ -local complementation at  $k$  decreases  $w_{i,j}$  by  $a$ . We show a fragment of a ZX-diagram below under the effect of this operation:

$$\begin{array}{c} \text{---} \\ \text{---} \end{array} \xrightarrow{*a} \begin{array}{c} \text{---} \\ \text{---} \end{array} \quad (21)$$

But the fragment above doesn't give the full picture. As in the qubit case, local complementation gives an equality up to introducing some single qubit phase gates on the outputs.

**Theorem 3.6.** Given  $a \in \mathbb{Z}_3$  and a graph state  $(G, W)$  containing a node  $k$ , let  $N(k)$  denote the neighbours of  $k$  - that is, nodes  $i$  with weight  $w_{i,k} \in \{1, 2\}$ . Then the following equality holds:

$$\begin{array}{c} \text{---} \\ \text{---} \end{array} = \begin{array}{c} \text{---} \\ \text{---} \end{array} \quad G *_a k$$

*Proof.* This is Theorem 4.4 and Corollary 4.5 in [11].

□

Composing local complementations gives a local pivot operation.

**Definition 3.7.** Given  $a, b, c \in \mathbb{Z}_3$  and a graph state  $G$  containing nodes  $i$  and  $j$ , the  $(a, b, c)$ -local pivot along  $ij$  is the new graph state  $G \wedge_{(a,b,c)} ij := ((G *_a i) *_b j) *_c i$ .

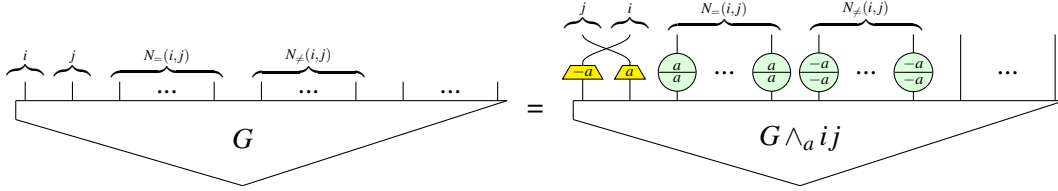


This again results in an equality, up to introducing some extra gates on outputs. Here we shall only consider an  $(a, -a, a)$ -local pivot along an edge  $ij$  of non-zero weight, for  $a \in \{1, 2\}$ . We will call this a *proper  $a$ -local pivot* along  $ij$ , and denote it  $G \wedge_a ij$ .

**Theorem 3.8.** Given  $a \in \mathbb{Z}_3$  and a graph state  $(G, W)$  containing connected nodes  $i$  and  $j$ , define the following:

- $N_=(i, j) := \{k \in N(i) \cap N(j) \mid w_{k,i} = w_{k,j}\}$
- $N_\neq(i, j) := \{k \in N(i) \cap N(j) \mid w_{k,i} \neq w_{k,j}\}$

Then the following equation relates  $G$  and its proper  $a$ -local pivot along  $ij$ :



*Proof.* See Appendix ??.

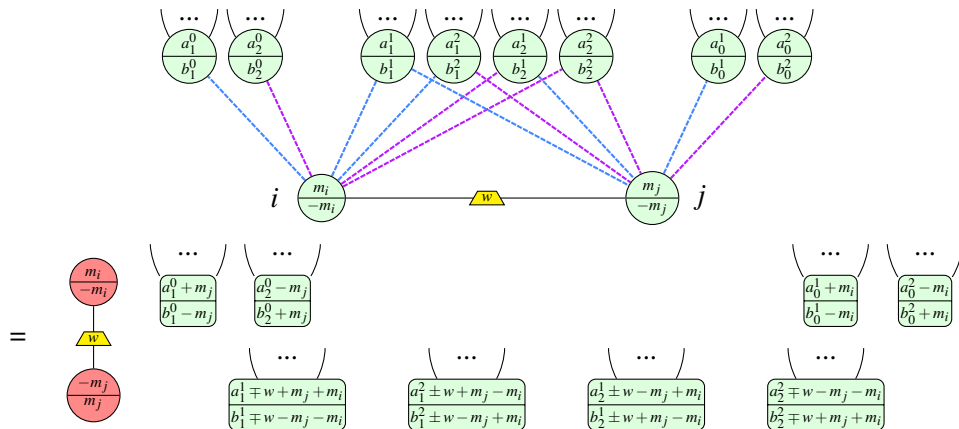
□

These two operations are again the drivers behind the simplification procedure. We classify spiders into three families:

$$\mathcal{M} = \left\{ \begin{pmatrix} \cdots \\ 0 \\ 0 \\ \cdots \end{pmatrix}, \begin{pmatrix} \cdots \\ 1 \\ 2 \\ \cdots \end{pmatrix}, \begin{pmatrix} \cdots \\ 2 \\ 1 \\ \cdots \end{pmatrix} \right\}, \quad \mathcal{N} = \left\{ \begin{pmatrix} \cdots \\ 0 \\ 1 \\ \cdots \end{pmatrix}, \begin{pmatrix} \cdots \\ 1 \\ 0 \\ \cdots \end{pmatrix}, \begin{pmatrix} \cdots \\ 0 \\ 2 \\ \cdots \end{pmatrix}, \begin{pmatrix} \cdots \\ 2 \\ 0 \\ \cdots \end{pmatrix} \right\}, \quad \mathcal{P} = \left\{ \begin{pmatrix} \cdots \\ 1 \\ 1 \\ \cdots \end{pmatrix}, \begin{pmatrix} \cdots \\ 2 \\ 2 \\ \cdots \end{pmatrix} \right\}.$$

exactly as in Theorem 3.1 [11]. We call a spider in a graph-like ZX-diagram *interior* if it isn't connected to an input or output (so for our use case all spiders are interior). Given any graph-like ZX-diagram, we will show that we can eliminate pairs of connected interior  $\mathcal{M}$ -spiders by local pivoting, and standalone interior  $\mathcal{N}$ - and  $\mathcal{P}$ -spiders by local complementation.

**Theorem 3.9.** Given any graph-like ZX-diagram containing two interior  $\mathcal{M}$ -spiders  $i$  and  $j$  connected by edge  $ij$  of weight  $w_{i,j} =: w \in \{1, 2\}$ , suppose we perform a proper  $\pm w$ -local pivot along  $ij$ . Then the new ZX-diagram is related to the old one by the equality:



where all changes to weights of edges where neither endpoint is  $i$  or  $j$  are omitted. Furthermore, in order to save space, each node with phase  $\begin{pmatrix} a_x^y \\ b_x^y \end{pmatrix}$  is representative of *all* nodes connected to  $i$  by an  $x$ -weighted edge and to  $j$  by a  $y$ -weighted edge.

*Proof.* See ?? in the Appendix.  $\square$

**Theorem 3.10.** Given any graph-like ZX-diagram containing an interior  $\mathcal{N}$ -spider  $k$  with phase  $\frac{0}{n}$  for  $n \in \{1, 2\}$ , suppose we perform a  $(-n)$ -local complementation at  $k$ . Then the new ZX-diagram is related to the old one by the equality:

where all changes to weights of edges where neither endpoint is  $k$  are omitted. If instead  $k$  has phase  $\frac{n}{0}$  for  $n \in \{1, 2\}$ , suppose we perform the same  $(-n)$ -local complementation at  $k$ . Then the equality relating the new and old diagrams becomes:

*Proof.* See Appendix ??.  $\square$

**Theorem 3.11.** Given any graph-like ZX-diagram containing an interior  $\mathcal{P}$ -spider  $k$  with phase  $\frac{p}{p}$  for  $p \in \{1, 2\}$ , suppose we perform a  $p$ -local complementation at  $k$ . Then the new ZX-diagram is related to the old one by the equality:

where all changes to weights of edges where neither endpoint is  $k$  are omitted.

*Proof.* See Appendix ??.  $\square$

We can now combine these three elimination theorems into an algorithm for efficiently simplifying a closed graph-like ZX-diagram. First note that after applying any one of the three elimination theorems to such a diagram we may end up with a state that is no longer graph-like. Fortunately the only way in which this can happen is if two spiders end up being connected by multiple  $H$ - or  $H^\dagger$ -edges, and we have shown via Lemmas 3.1 and 3.2 that these can always be reduced to just one edge.

**Theorem 3.12.** Given any closed graph-like ZX-diagram, the following algorithm will always terminate after a finite number of steps, returning an equivalent graph-like ZX-diagram with no  $\mathcal{N}$ -spiders,  $\mathcal{P}$ -spiders, or adjacent pairs of  $\mathcal{M}$ -spiders. Repeat the steps below until no rule matches. After each step, apply Lemmas 3.1 and 3.2 as needed until the resulting diagram is graph-like:

1. Eliminate an  $\mathcal{N}$ -spider via Theorem 3.10.
2. Eliminate a  $\mathcal{P}$ -spider via Theorem 3.11.
3. Eliminate two adjacent  $\mathcal{M}$ -spiders via Theorem 3.9.

*Proof.* At every step the total number of spiders decreases by at least one, so since we start with a finite diagram the algorithm terminates after a finite number of steps. By construction, when it does so we are left with an equivalent graph-like ZX-diagram with no  $\mathcal{N}$ -spiders,  $\mathcal{P}$ -spiders, or adjacent pairs of  $\mathcal{M}$ -spiders.  $\square$

**Corollary 3.13.** In particular, if we start with a stabilizer diagram, we can eliminate all but perhaps one  $\mathcal{M}$ -spider, depending on whether the initial number of  $\mathcal{M}$ -spiders was odd or even.

*Proof.* No step introduces any non-stabilizer phases.  $\square$

The algorithm above could be extended to graph-like diagrams with inputs or outputs as in Theorem 5.4 [8], but since for our purposes we don't need to do so, we have not gone to the trouble.

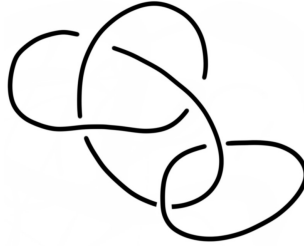
## 4 Case studies

### 4.1 Jones Polynomial at Lattice Roots of Unity

For  $d = 2, 3, 4$  it's easy. For  $d \geq 5$  it's hard.

Mention Potts and also anyon stuff but the important thing is the tensor net.

A knot  $K$  is a circle embedded in  $\mathbb{R}^3$ . A set of knots tangled together is a *link*  $L$ . A link  $L$  is represented as the projection of the link on  $\mathbb{R}^2$  but retaining the information of over or under crossings. For example, the trefoil knot linked with an unknot can be drawn as:



(22)

We say that  $L \simeq L'$  if there is a sequence of Reidemeister moves from  $L$  to  $L'$ , i.e. reversible local strand deformations that do not change the topology, for example by cutting or gluing strands (see Appendix).

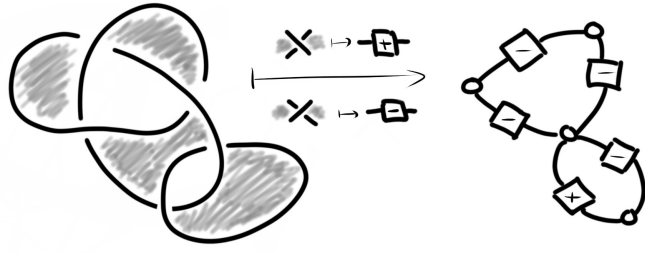
The Jones polynomial  $V_L(t)$  is a Laurent polynomial in  $t \in \mathbb{C}$  and is a link *invariant*. This means that  $V_L(t) \neq V_{L'}(t) \Rightarrow L \not\simeq L'$ , where  $L \simeq L'$ .

In general, computing  $V_L(t)$  is exponentially costly in the number of crossings  $|C|$ , something made explicit when one uses the Kauffman bracket method where a skein relation is used iteratively on every crossing [KauffmanBook]. Exactly evaluating the Jones polynomial at specific points  $t \in \mathbb{C}$  is #P-hard, *except* at the *lattice roots of unity*  $\pm 1, \pm i, \pm e^{i2\pi/3}, \pm e^{i4\pi/3}$  where it can be evaluated efficiently (in time  $O(\text{poly}(|C|))$ ) [Welsh].

In the context of quantum computation, additively approximating the Jones polynomial at non-lattice roots of unity is the paradigmatic BQP-complete problem [aharonov]. For the lattice roots of unity, the quantum amplitude to which the problem reduces can be computed efficiently. Specifically, the quantum circuits that need to be simulated are stabiliser circuits which are known to be classically tractable [ref? Kuperberg? Aharonov?]. From the more natural point of view of topological quantum computation, the Jones polynomial at roots of unity corresponds to a quantum amplitude in the fusion

space of non-abelian anyons. Computation is performed by initialising, braiding, and the anyons. Specifically, in the case of  $SU(2)_k$  anyons, the Jones polynomial is evaluated at  $e^{i\frac{2\pi}{2+k}}$  [Rowel-Wang]. Note the correspondence  $q \in \{1, 2, 3, 4\} \Leftrightarrow k \in 1, 2, 4$ . This is consistent with the fact that topological quantum computation with  $SU(2)_2$  anyons (Ising) or  $SU(2)_4$  anyons is not universal (unless the  $SU(2)_4$  anyons are augmented by fusion and measurements in which case they are capable of universal quantum computation [1504.02098]).

Remarkably, the Jones polynomial can be expressed in terms of the partition function of a  $q$ -state Potts model [Wu]. The Potts model is defined on a signed graph, called the Tait graph, which is obtained as follows. The link diagram is bicoloured (checkerboard style) and every coloured area is mapped to a vertex and every crossing is mapped to a signed edge according to its orientation relative to the surrounding colours. Every vertex is assigned a  $q$ -state classical spin and every signed edge indicates a spin-spin Tait-sign-dependent interaction  $J_{\pm} \in \mathbb{C}$  which relate to the Jones polynomial's variable as  $e^{J_{\pm}} = -t^{\mp}$ . The role of the link invariant is played by the partition function  $Z(q)$ ; in fact, the interactions  $J_{\pm}$  are such so that the graph operations corresponding to Reidemeister moves leave  $Z(q)$  invariant. Multiplying with an efficiently computable prefactor  $\mathcal{A}(t) = (-t^{\frac{1}{2}} - t^{-\frac{1}{2}})^{(-|V|-1)} (-t^{\frac{3}{4}})^w t^{\frac{1}{4}\tau}$  for bookkeeping of twist factors, the relation to the Jones polynomial is  $V_L(t) = \mathcal{A}(t)Z(q)$ . The link shown in Eq.22 returns the following tensor network.


(23)

For  $q = 2$  in ZX notation:

$$\begin{aligned} \left[ \begin{array}{|c|} \hline \oplus \\ \hline \end{array} \right] &= \left[ \begin{array}{c} \text{---} \bigcirc \text{---} \end{array} \right]_{\pi/2} \sqrt{2} e^{i\frac{\pi}{4}} \\ \left[ \begin{array}{|c|} \hline \ominus \\ \hline \end{array} \right] &= \left[ \begin{array}{c} \text{---} \bigcirc \text{---} \end{array} \right]_{3\pi/2} \sqrt{2} e^{-i\frac{\pi}{4}} \end{aligned}$$
(24)

We provide a graphical proof that evaluating the Jones polynomial of arbitrary links at the lattice roots of unity is in P. For these cases, the problem reduces to the evaluation of the partition function of a planar  $q$ -state Potts model. The partition is represented as a closed tensor network. We employ the ZX-calculus, which is a sound and complete graphical language for tensor networks and allows reasoning via graphical rewrites. We show that there exist polynomially long rewrite sequences that fully simplify the tensor network and return  $Z(q)$  for  $q \in \{1, 2, 3, 4\}$ , which correspond to evaluating the Jones polynomial at the lattice roots of unity.

**Remark 4.1.** A small technical remark is in order: the rule set given here is complete for qubit stabilizer quantum mechanics when equality is taken only up to a scalar factor. To achieve completeness under exact equality, a slightly modified rule set would be required, as in (for example) [2]. But for our purposes this would be overkill; in order to show that the Jones polynomial of knots at lattice roots of unity is

efficiently computable, it suffices to consider the ‘non-exact’ rules - i.e. it suffices to show such a Jones polynomial is efficiently computable up to a scalar factor. But of course to actually compute such a Jones polynomial, we will need to keep track of scalars.

We can now give ZX-diagrams whose standard interpretations are equal (up to a scalar) to the matrices  $T_{\pm}^{(q)}$  in (??) for  $q \in \{2, 4\}$ .

For spiders  $S$  and  $T$ , we then have:

$$\llbracket S \otimes T \rrbracket = \llbracket S \rrbracket \otimes \llbracket T \rrbracket \quad (25)$$

$$\llbracket S \circ T \rrbracket = \llbracket S \rrbracket \llbracket T \rrbracket \quad (26)$$

where on the right hand side of the first equation the  $\otimes$  symbol means the Kronecker product of two matrices. Thus a diagram with standard interpretation  $T_{2\pm}$  will have one input and one output, while a diagram for  $T_{4\pm}$  will have two inputs and two outputs.

**Proposition 4.2.** Under the standard interpretation as linear maps, the following diagrams give (up to a scalar) the required matrices:

$$\llbracket \text{red circle with } \pm \frac{\pi}{2} \rrbracket \simeq \llbracket \text{white square with } \pm \rrbracket_{q=2}, \quad \llbracket \text{red circle with } \pi \text{ connected by a yellow square to another red circle with } \pi \rrbracket \simeq \llbracket \text{white square with } \pm \rrbracket_{q=4}. \quad (27)$$

*Proof.* See Appendix ??.

□

Since any ZX-diagram derived from a knot in the manner described in Section ?? will be closed, this algorithm suffices to prove that the calculation of the Jones polynomial of any knot at the lattice roots of unity  $\pm 1$  and  $\pm i$  is efficient. This is because reading off the scalar at the end is trivial; either we have the empty diagram, which has standard interpretation 1, or a single legless Z-spider with phase  $k\pi$ :

$$\llbracket \text{green circle with } k\pi \rrbracket = \llbracket \text{green circle with } k\pi \text{ and a green dot above it} \rrbracket = (\langle 0| + \langle 1|)(|0\rangle + e^{ik\pi}|1\rangle) = 1 + e^{ik\pi} \quad (28)$$

Furthermore, keeping track of the scalar factors introduced with each application of Theorem ?? or Lemma ?? can be done efficiently. [ToDo: proof, if not explicitly doing all this in a scalar-exact fashion. Reference Backens]

Having now defined the qutrit ZX-calculus we turn our attention back to our tensor network for the Jones polynomial of a knot (ToDo: ref). We are seeking a diagram in the qutrit ZX-calculus that equals (up to a scalar) the matrix  $T_{\pm}^{(q)}$  from (??).

**Proposition 4.3.** Under the standard interpretation as a linear map, the following diagram gives (up to a scalar) the required matrix:

$$\llbracket \text{red circle with } \pm 1 \text{ and } \pm 1 \text{ on the sides} \rrbracket \simeq \llbracket \text{white square with } \pm \rrbracket_{q=3} = \begin{pmatrix} e^{\mp i \frac{\pi}{3}} & 1 & 1 \\ 1 & e^{\mp i \frac{\pi}{3}} & 1 \\ 1 & 1 & e^{\mp i \frac{\pi}{3}} \end{pmatrix} \quad (29)$$

*Proof.* See Appendix ??.

□

Crucially, the ZX-diagram in Proposition 4.3 above is a *stabilizer diagram* in the qutrit ZX-calculus - that is, all angles are integer multiples of  $\frac{2\pi}{3}$ . Therefore if we can find an algorithm analogous to Theorem 5.4 [8] that efficiently reduces any stabilizer diagram to a trivial one, then we will have shown that the Jones polynomial of any knot at the lattice roots of unity  $\pm e^{i\frac{\pi}{3}}$  is efficiently computable. [ToDo: justify the  $\pm$ ]. In the next subsection, we will do exactly that.

## 4.2 Graph Colouring

## 5 Outlook

This also motivates further work in circuit extraction for qutrit circuits so that one can attempt graph-theoretic simplification with the qutrit ZX calculus, analogous to the case of qubits [8, 4]. **harny's crazy generalisations: what can they potentially do for general CSP?**

Future work, scalar exact version of the qutrit rewrite rules, for concrete applications to problems where 3-state systems are involved. Further, generalisation of local complementation and pivot rewrites to arbitrary prime dimension. Software implementations a la pyzx could be extended accordingly.

## 6 Acknowledgments

Alex would like to thank Aleks Kissinger and Quanlong (Harny) Wang for their help throughout this research. KM wishes to thank Niel de Beaudrap, Aleks Kissinger, Stefanos Kourtis, and Quanlong (Harny) Wang for inspiring and helpful discussions. KM acknowledges financial support from the Royal Commission for the Exhibition of 1851 through a research fellowship.

## References

- [1] Scott Aaronson & Daniel Gottesman (2004): *Improved simulation of stabilizer circuits*. *Physical Review A* 70(5), doi:10.1103/physreva.70.052328. Available at <http://dx.doi.org/10.1103/PhysRevA.70.052328>.
- [2] Miriam Backens (2015): *Making the stabilizer ZX-calculus complete for scalars*.
- [3] Miriam Backens & Aleks Kissinger (2018): *ZH: A Complete Graphical Calculus for Quantum Computations Involving Classical Non-linearity*.
- [4] Miriam Backens, Hector Miller-Bakewell, Giovanni de Felice, Leo Lobski & John van de Wetering (2020): *There and back again: A circuit extraction tale*.
- [5] Niel de Beaudrap, Aleks Kissinger & Konstantinos Meichanetzidis (2020): *Tensor Network Rewriting Strategies for Satisfiability and Counting*.
- [6] Bob Coecke & Ross Duncan (2011): *Interacting quantum observables: categorical algebra and diagrammatics*. *New Journal of Physics* 13(4), p. 043016, doi:10.1088/1367-2630/13/4/043016. Available at <http://dx.doi.org/10.1088/1367-2630/13/4/043016>.
- [7] Carsten Damm, Markus Holzer & Pierre McKenzie (2002): *The complexity of tensor calculus*. *computational complexity* 11(1), pp. 54–89, doi:10.1007/s00037-000-0170-4. Available at <https://doi.org/10.1007/s00037-000-0170-4>.
- [8] Ross Duncan, Aleks Kissinger, Simon Perdrix & John van de Wetering (2020): *Graph-theoretic Simplification of Quantum Circuits with the ZX-calculus*.

- [9] Xiaoyan Gong & Quanlong Wang (2017): *Equivalence of Local Complementation and Euler Decomposition in the Qutrit ZX-calculus.*
- [10] Johnnie Gray & Stefanos Kourtis (2020): *Hyper-optimized tensor network contraction.*
- [11] Quanlong Wang (2018): *Qutrit ZX-calculus is Complete for Stabilizer Quantum Mechanics.*
- [12] Quanlong Wang (2020): *Completeness of algebraic ZX-calculus over arbitrary commutative rings and semirings.*
- [13] John van de Wetering (2020): *ZX-calculus for the working quantum computer scientist.*