

Classifying Complexity with the ZX Calculus

Alex Townsend-Teague^{0,1} and Konstantinos Meichanetzidis^{1,2}

⁰ Mathematical Institute, University of Oxford

¹ Department of Computer Science, University of Oxford

² 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. The generators are tensors called *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

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

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:

$$\left[\begin{array}{c} \overbrace{\quad}^m \\ \vdots \\ \text{green circle } \alpha \\ \vdots \\ \underbrace{\quad}_n \end{array} \right] = |0\rangle^m \langle 0|^n + e^{i\alpha} |1\rangle^m \langle 1|^n, \quad \left[\begin{array}{c} \overbrace{\quad}^m \\ \vdots \\ \text{red circle } \alpha \\ \vdots \\ \underbrace{\quad}_n \end{array} \right] = |+\rangle^m \langle +|^n + e^{i\alpha} |-\rangle^m \langle -|^n, \quad (2)$$

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*:

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

The rewrite rules of the ZX calculus (see Fig.2 in B) 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. 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:

$$\begin{array}{c} \text{graph-like diagram with spiders } \alpha_1, \dots, \alpha_n \text{ and a central spider } \pm \frac{\pi}{2} \end{array} = \begin{array}{c} \text{rewritten graph-like diagram with spiders } \alpha_1 \mp \frac{\pi}{2}, \dots, \alpha_n \mp \frac{\pi}{2} \end{array} \quad (3)$$

$$(4)$$

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 $\bar{\omega}$ its complex conjugate. The qutrit X-basis consists of the three vectors:

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

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

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

Hadamard boxes are now neither self-conjugate nor self-adjoint, though they remain self-transpose. Therefore, in keeping with the diagrammatic paradigm, the 3-dimensional Hadamard gate is denoted by a yellow parallelogram, as it has neither vertical nor horizontal symmetry, but is invariant under rotation by π . We also use a dashed blue line for the *Hadamard edge* (H -edge) and a purple dashed line for its adjoint (H^\dagger -edge):

$$\begin{array}{c} \text{yellow parallelogram} \end{array} = \begin{array}{c} \text{blue dashed line} \end{array} = \begin{array}{c} \text{green circle with } 2/2 \\ \text{red circle with } 2/2 \\ \text{green circle with } 2/2 \end{array}, \quad \begin{array}{c} \text{yellow parallelogram} \end{array} = \begin{array}{c} \text{purple dashed line} \end{array} = \begin{array}{c} \text{green circle with } 1/1 \\ \text{red circle with } 1/1 \\ \text{green circle with } 1/1 \end{array} \quad (9)$$

$$\text{Diagram 1} \neq \text{Diagram 2}, \quad \text{Diagram 3} \neq \text{Diagram 4} \quad (10)$$

Lemma 3.1. In the qutrit ZX-calculus, the following results hold:

The figure consists of two parts, each illustrating the string equation. Each part shows an equality between two diagrams. In the left part, a diagram with a blue dashed line connecting a vertex labeled $\frac{\alpha}{\beta}$ to a vertex labeled $\frac{\gamma}{\delta}$ is equal to a diagram where the line is reversed. In the right part, a diagram with a purple dashed line connecting a vertex labeled $\frac{\alpha}{\beta}$ to a vertex labeled $\frac{\gamma}{\delta}$ is equal to a diagram where the line is reversed. The vertices are green circles with the labels $\frac{\alpha}{\beta}$ and $\frac{\gamma}{\delta}$ inside. The lines are blue and purple dashed lines. The diagrams are set against a background of other lines and vertices, representing a larger graph structure.

Secondly, the ‘snake equations’ continue to hold in some form; given a cap and cup of different colours, we can still straighten out the wire between them as before, only now it comes at the cost of adding two Hadamard boxes to the wire. For a cup and cap of the same colour, we can straighten out the wire as usual.

Lemma 3.2. In the qutrit ZX-calculus, the following results hold. Moreover, they hold with the roles of green and red interchanged.

$$\begin{array}{c} \text{green cap} \\ \text{green cup} \end{array} = | \quad , \quad \begin{array}{c} \text{green cap} \\ \text{red cup} \end{array} = \begin{array}{c} \text{Hadamard} \\ \text{Hadamard} \end{array} . \quad (11)$$

Proof. As we will soon see, these are just rules (f) and (s). \square

We now give a full set of rules defining the qutrit ZX-calculus. Our presentation aims for clarity and accessibility; for a more rigorous description, see [11].

Definition 3.3. The *qutrit ZX-calculus* is a graphical calculus generated by the following diagrams, where $\alpha, \beta \in [0, 2\pi]$:

$$\begin{array}{c} \dots \\ \text{green spider } \alpha, \beta \\ \dots \end{array} , \quad \begin{array}{c} \dots \\ \text{red spider } \alpha, \beta \\ \dots \end{array} , \quad \begin{array}{c} \text{Hadamard} \end{array} , \quad \begin{array}{c} \text{crossing} \end{array} , \quad | \quad (12)$$

and their adjoints $(-)^{\dagger}$. The non-spider generators’ adjoints are their vertical reflections, whereas spiders’ adjoints are found by swapping inputs and outputs and negating angles:

$$\left(\begin{array}{c} \overbrace{\dots}^m \\ \dots \\ \text{green spider } \alpha, \beta \\ \dots \\ \underbrace{\dots}_n \end{array} \right)^{\dagger} = \begin{array}{c} \overbrace{\dots}^n \\ \dots \\ \text{green spider } -\alpha, -\beta \\ \dots \\ \underbrace{\dots}_m \end{array} , \quad \left(\begin{array}{c} \overbrace{\dots}^m \\ \dots \\ \text{red spider } \alpha, \beta \\ \dots \\ \underbrace{\dots}_n \end{array} \right)^{\dagger} = \begin{array}{c} \overbrace{\dots}^n \\ \dots \\ \text{red spider } -\alpha, -\beta \\ \dots \\ \underbrace{\dots}_m \end{array} \quad (13)$$

These generators can be composed in parallel (\otimes) and sequentially (\circ), and the resulting diagrams are governed by the rewrite rules in Figure 1, wherein addition is modulo 2π . The fusion rule (f) applies to spiders of the same colour connected by at least one wire. Importantly, all the rules hold under taking adjoints, where for diagrams D and E we have:

$$(D \otimes E)^{\dagger} = D^{\dagger} \otimes E^{\dagger} \quad (14)$$

$$(D \circ E)^{\dagger} = D^{\dagger} \circ E^{\dagger} \quad (15)$$

Furthermore, all but the commutation equations (cm) and the colour change equations (cc) continue to hold when the roles of green and red (i.e. Z and X) are interchanged. For these four exceptions, however, analogous equations can be derived from the existing ones; for example, the corresponding colour change equations will be relevant for us later.

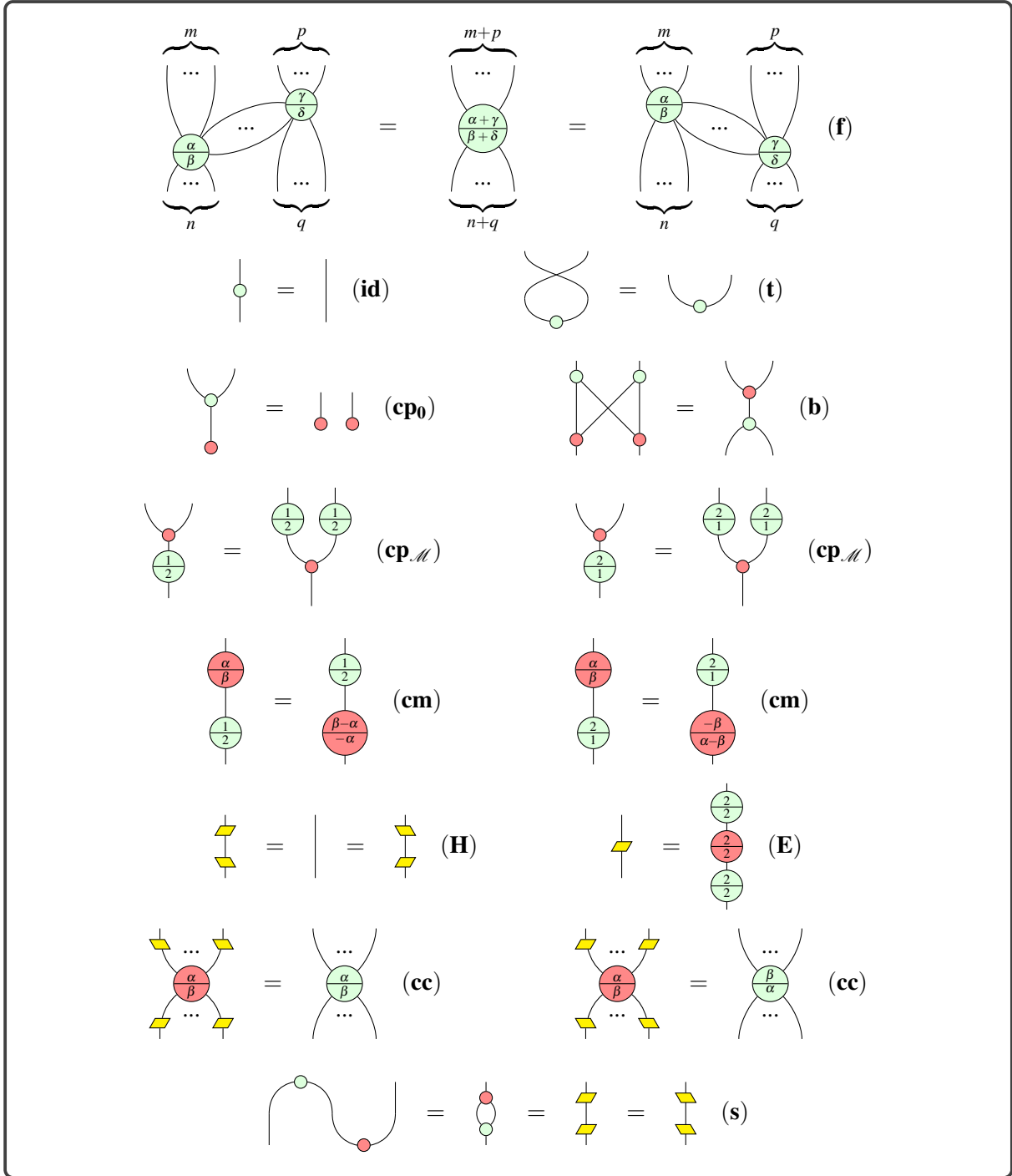


Figure 1: Rewrite rules for the qutrit ZX-calculus.

Proposition 3.4. The following equations are derivable in the qutrit ZX-calculus:

$$\begin{array}{c} \dots \\ \text{red spider } \begin{smallmatrix} \alpha \\ \beta \end{smallmatrix} \\ \dots \end{array} = \begin{array}{c} \text{green spider } \begin{smallmatrix} \alpha \\ \beta \end{smallmatrix} \\ \dots \end{array}, \quad \begin{array}{c} \dots \\ \text{red spider } \begin{smallmatrix} \beta \\ \alpha \end{smallmatrix} \\ \dots \end{array} = \begin{array}{c} \text{green spider } \begin{smallmatrix} \alpha \\ \beta \end{smallmatrix} \\ \dots \end{array} \quad (16)$$

Proof. Add H - and H^\dagger -boxes to both sides of the original colour change equations in such a way that we can then cancel Hadamards on the legs of the red spiders via **(H)**. \square

3.2 Graph-Like Qutrit ZX Diagrams

[Make all of the following scalar-exact (?)]

We first define a graph-like diagram in the qutrit ZX calculus.

Definition 3.5. A qutrit ZX-diagram is *graph-like* when:

1. Every spider is a Z-spider.
2. Spiders are only connected by Hadamard edges (H -edges) or their adjoints (H^\dagger -edges).
3. Every pair of spiders is connected by at most one H -edge or H^\dagger -edge.
4. Every input and output is connected to a spider.
5. Every spider is connected to at most one input or output.

Thanks to Lemma 3.1 we can ignore which of a spider's H - and H^\dagger -legs are inputs and outputs. For our specific needs, the last two items above will not be relevant, since ZX-diagrams arising from knots will be closed, but we include them to keep the definition consistent with the qubit case. However, 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, which holds with the roles of H and H^\dagger reversed, helps us get around this:

$$\begin{array}{c} \dots \\ \text{green spider } \begin{smallmatrix} \alpha \\ \beta \end{smallmatrix} \\ \dots \end{array} \begin{array}{c} \dots \\ \text{green spider } \begin{smallmatrix} \gamma \\ \delta \end{smallmatrix} \\ \dots \end{array} \stackrel{\text{(H)}}{=} \begin{array}{c} \dots \\ \text{green spider } \begin{smallmatrix} \alpha \\ \beta \end{smallmatrix} \\ \dots \end{array} \begin{array}{c} \dots \\ \text{green spider } \begin{smallmatrix} \gamma \\ \delta \end{smallmatrix} \\ \dots \end{array} \stackrel{\text{(id)}}{=} \begin{array}{c} \dots \\ \text{green spider } \begin{smallmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{smallmatrix} \\ \dots \end{array} \quad (17)$$

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

Lemma 3.6. The following equation holds in the qutrit ZX-calculus. Moreover, it holds with the roles of H and H^\dagger interchanged:

$$\begin{array}{c} \dots \\ \text{green spider } \begin{smallmatrix} \alpha \\ \beta \end{smallmatrix} \\ \dots \end{array} \begin{array}{c} \dots \\ \text{green spider } \begin{smallmatrix} \gamma \\ \delta \end{smallmatrix} \\ \dots \end{array} = \begin{array}{c} \dots \\ \text{green spider } \begin{smallmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{smallmatrix} \\ \dots \end{array} \quad (18)$$

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

$$(19)$$

Therefore we can argue as follows:

$$(20)$$

□

Lemma 3.7. The following two equations hold in the qutrit ZX-calculus. Moreover, they hold with the roles of H and H^\dagger interchanged:

$$(21)$$

Proof. This is Lemma 3.4 in [9].

□

As we will formalise later, the lemmas above say that 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. This motivates defining ‘parametrised’ Hadamard gates, which will come in use later:

$$(22)$$

The first equation above just says that a ‘0-Hadamard edge’ is in fact the empty diagram, and not an edge at all. 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 3.8. The following three equations hold in the qutrit ZX calculus. Moreover, they hold with the roles of H and H^\dagger interchanged:

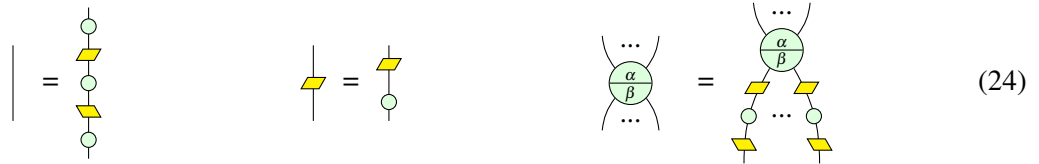
$$(23)$$

Proof. For the first equation, turn the top two H -boxes into H^\dagger -boxes via (s), then cancel twice via (H). Similarly, for the second equation turn the top two H -boxes into H^\dagger -boxes via (s), then cancel once via (H), leaving a single H^\dagger -box. The third equation is just a simple application of (id). Analogous reasoning applies when the roles of H and H^\dagger are swapped. \square

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

Corollary 3.9. 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 3.8 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 (17) where fusion is not possible, so that no plain edge connects two spiders. Apply Lemmas 3.6 and 3.7 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:



$$(24)$$

\square

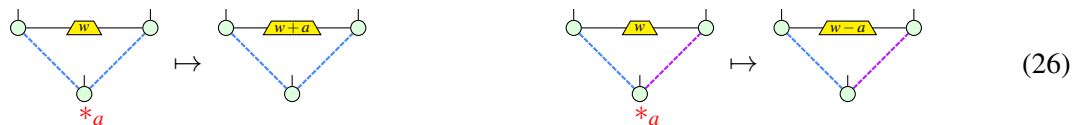
Definition 3.10. A graph-like qutrit ZX diagram is a *graph state* when every spider has zero phase (top and bottom) and is connected to an output.

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 3.11. 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:

$$w'_{i,j} = w_{i,j} + aw_{i,k}w_{j,k} \quad (25)$$

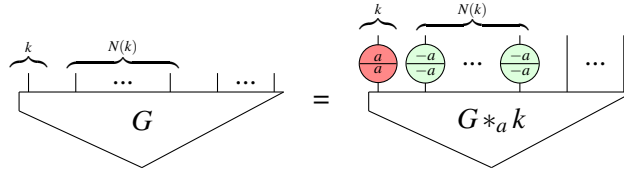
So only those edges between neighbours of node k are affected, but rather than just having their weight increased by 1 (modulo 2) as in the qubit case, the increase in weight also depends on the weights of the edges from i and j to k . As always, this is best seen graphically. For two nodes i and j both 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:



$$(26)$$

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 3.12. 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:



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

Composing local complementations gives a local pivot operation.

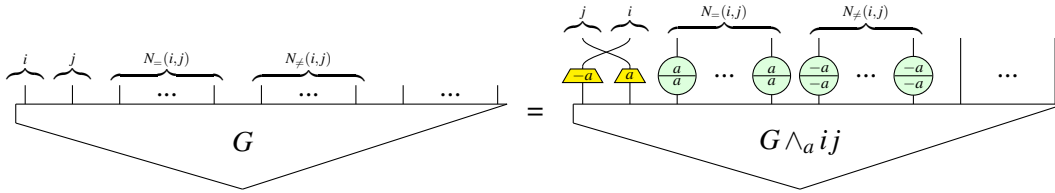
Definition 3.13. Given $a, b, c \in \mathbb{Z}_3$ and a graph state G containing nodes i and j , the (a, b, c) -local pivot along ij is the new graph state $G \wedge_{(a,b,c)} ij := ((G * a i) * b j) * c i$.

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 3.14. 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 :



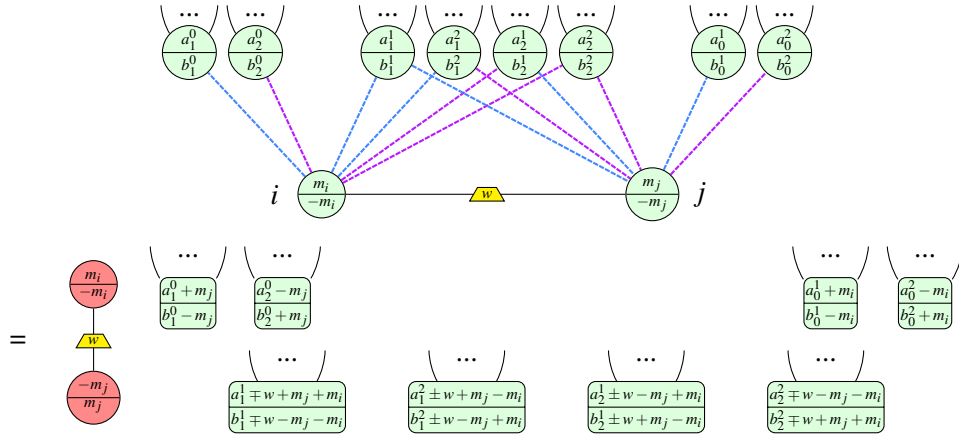
Proof. See Appendix C.1. \square

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

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

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.

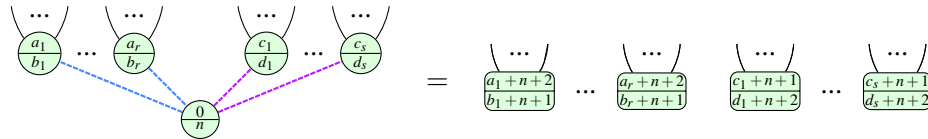
Theorem 3.15. Given any graph-like ZX-diagram containing two interior \mathcal{M} -spiders i and j connected by edge ij of weight $w_{i,j} =: w \in \{1, 2\}$, suppose we perform a proper $\pm w$ -local pivot along ij . Then the new ZX-diagram is related to the old one by the equality:



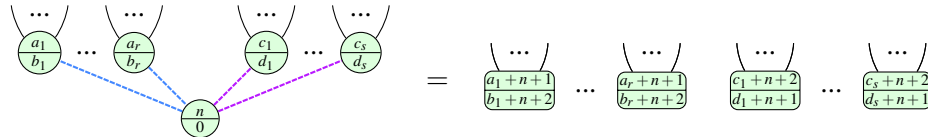
where all changes to weights of edges where neither endpoint is i or j are omitted. Furthermore, in order to save space, each node with phase $\begin{pmatrix} a_x^y \\ b_x^y \end{pmatrix}$ is representative of *all* nodes connected to i by an x -weighted edge and to j by a y -weighted edge.

Proof. See D.3 in the Appendix. \square

Theorem 3.16. Given any graph-like ZX-diagram containing an interior \mathcal{N} -spider k with phase $\begin{pmatrix} 0 \\ n \end{pmatrix}$ 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:

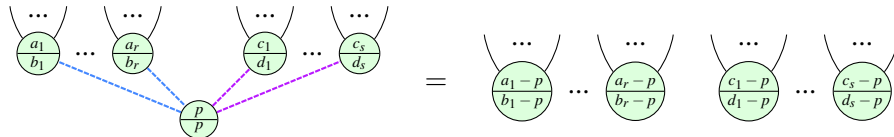


where all changes to weights of edges where neither endpoint is k are omitted. If instead k has phase $\begin{pmatrix} n \\ 0 \end{pmatrix}$ for $n \in \{1, 2\}$, suppose we perform the same $(-n)$ -local complementation at k . Then the equality relating the new and old diagrams becomes:



Proof. See Appendix D.6. \square

Theorem 3.17. Given any graph-like ZX-diagram containing an interior \mathcal{P} -spider k with phase $\begin{pmatrix} p \\ p \end{pmatrix}$ 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 D.8. \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 3.6 and 3.7 that these can always be reduced to just one edge.

Theorem 3.18. 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 3.6 and 3.7 as needed until the resulting diagram is graph-like:

1. Eliminate an \mathcal{N} -spider via Theorem 3.16.
2. Eliminate a \mathcal{P} -spider via Theorem 3.17.
3. Eliminate two adjacent \mathcal{M} -spiders via Theorem 3.15.

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 3.19. 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.

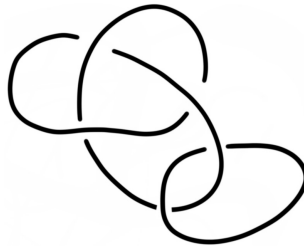
4 Case studies

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:



(27)

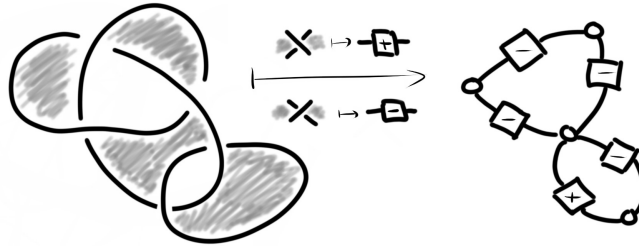
We say that $L \simeq L'$ if there is a sequence of Reidemeister moves from L to L' , i.e. reversible local strand deformations that do not change the topology, for example by cutting or gluing strands (see Appendix).

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.27 returns the following tensor network.



(28)

For $q = 2$ in ZX notation:

$$\llbracket \text{---} \boxed{+} \text{---} \rrbracket = \llbracket \text{---} \text{---} \text{---} \rrbracket \sqrt{2} e^{i\frac{\pi}{4}}$$

$$\llbracket \text{---} \boxed{-} \text{---} \rrbracket = \llbracket \text{---} \text{---} \text{---} \rrbracket \sqrt{2} e^{-i\frac{\pi}{4}}$$

(29)

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:

$$\llbracket S \otimes T \rrbracket = \llbracket S \rrbracket \otimes \llbracket T \rrbracket \quad (30)$$

$$\llbracket S \circ T \rrbracket = \llbracket S \rrbracket \llbracket T \rrbracket \quad (31)$$

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:

$$\llbracket \text{---} \text{---} \text{---} \rrbracket \simeq \llbracket \text{---} \boxed{\pm} \text{---} \rrbracket_{q=2}, \quad \llbracket \text{---} \text{---} \text{---} \rrbracket \simeq \llbracket \text{---} \boxed{\pm} \text{---} \rrbracket_{q=4}. \quad (32)$$

Proof. See Appendix B.1. □

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 ± 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 $k\pi$:

$$\llbracket \textcircled{k\pi} \rrbracket = \left[\begin{array}{c} \circ \\ | \\ \textcircled{k\pi} \end{array} \right] = (\langle 0| + \langle 1|)(|0\rangle + e^{ik\pi}|1\rangle) = 1 + e^{ik\pi} \quad (33)$$

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:

$$\left[\begin{array}{c} \pm 1 \\ \pm 1 \\ \pm 1 \end{array} \right] \simeq \left[\begin{array}{c} \pm \end{array} \right]_{q=3} = \begin{pmatrix} e^{\mp i \frac{\pi}{3}} & 1 & 1 \\ 1 & e^{\mp i \frac{\pi}{3}} & 1 \\ 1 & 1 & e^{\mp i \frac{\pi}{3}} \end{pmatrix} \quad (34)$$

Proof. See Appendix B.1. □

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

Teague? KM wishes to thank Niel de Beaudrap, Aleks Kissinger, Stefanos Kourtis, and Quanlong 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*.

A Qubit ZX Calculus

The rewrite rules of the qubit ZX calculus are shown in Fig.2.

B PM-Boxes in ZX Language

We begin with proofs of the translations of the matrix $T_{\pm}^{(q)}$ into the ZX-calculus.

Proposition B.1. / Propositions 4.2, 4.3. The following equalities hold up to a scalar under the standard interpretation:

$$\left[\begin{array}{c} \text{red circle with } \pm \frac{\pi}{2} \end{array} \right] \simeq \left[\begin{array}{c} \text{white box with } \pm \end{array} \right]_{q=2}, \quad \left[\begin{array}{c} \text{red circle with } \pm \frac{\pi}{3} \end{array} \right] \simeq \left[\begin{array}{c} \text{white box with } \pm \end{array} \right]_{q=3}, \quad \left[\begin{array}{c} \text{red circle with } \pi \text{ --- yellow square --- red circle with } \pi \end{array} \right] \simeq \left[\begin{array}{c} \text{white box with } \pm \end{array} \right]_{q=4}$$

Proof. Recalling $\omega = e^{i\frac{2\pi}{3}}$, the standard interpretations of phase gates in matrix form are:

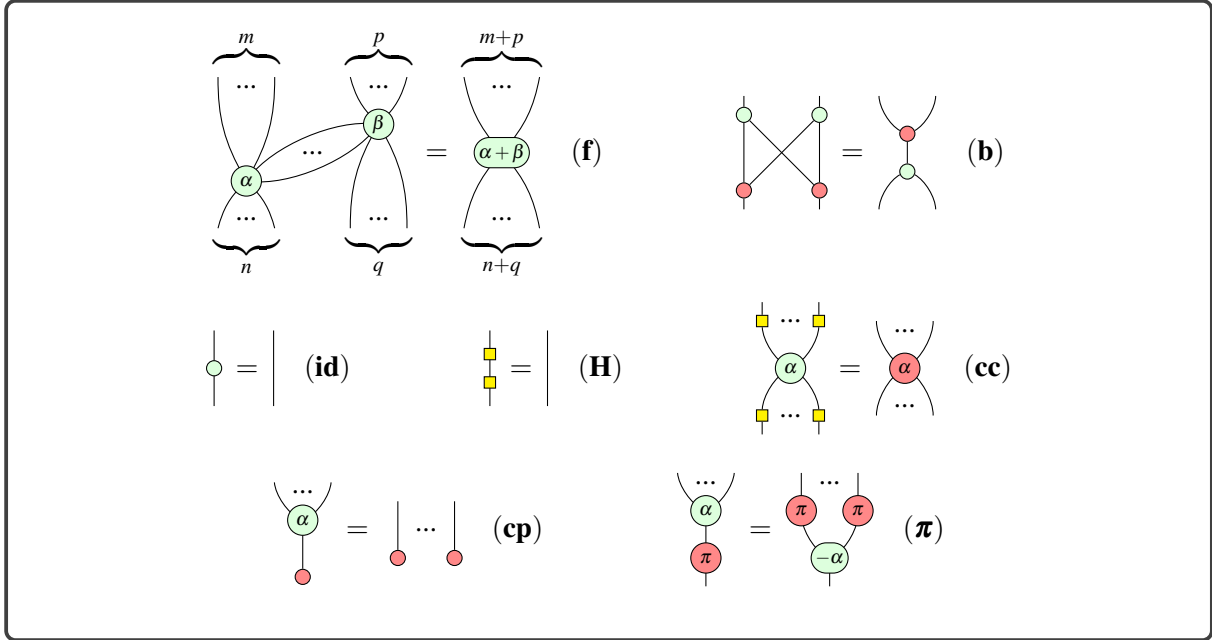


Figure 2: Rewrite rules for the qubit ZX-calculus where spiders are interpreted as tensors over \mathbb{C} .

$$\left[\begin{array}{c} \alpha \\ | \end{array} \right] = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\alpha} \end{pmatrix}, \quad \left[\begin{array}{c} \alpha \\ | \end{array} \right] = \frac{1}{2} \begin{pmatrix} 1 + e^{i\alpha} & 1 - e^{i\alpha} \\ 1 - e^{i\alpha} & 1 + e^{i\alpha} \end{pmatrix}, \quad \left[\begin{array}{c} \alpha \\ \beta \\ | \end{array} \right] = \begin{pmatrix} 1 & 0 & 0 \\ 0 & e^{i\alpha} & 0 \\ 0 & 0 & e^{i\beta} \end{pmatrix}$$

$$\left[\begin{array}{c} \alpha \\ \beta \\ | \end{array} \right] = \frac{1}{3} \begin{pmatrix} 1 + e^{i\alpha} + e^{i\beta} & 1 + \bar{\omega}e^{i\alpha} + \omega e^{i\beta} & 1 + \omega e^{i\alpha} + \bar{\omega}e^{i\beta} \\ 1 + \omega e^{i\alpha} + \bar{\omega}e^{i\beta} & 1 + e^{i\alpha} + e^{i\beta} & 1 + \bar{\omega}e^{i\alpha} + \omega e^{i\beta} \\ 1 + \bar{\omega}e^{i\alpha} + \omega e^{i\beta} & 1 + \omega e^{i\alpha} + \bar{\omega}e^{i\beta} & 1 + e^{i\alpha} + e^{i\beta} \end{pmatrix}$$

So in the simplest case $q = 2$ it is fairly straightforward to see that:

$$\left[\begin{array}{c} \pm \frac{\pi}{2} \\ | \end{array} \right] = \frac{1}{2} \begin{pmatrix} 1 \pm i & 1 \mp i \\ 1 \mp i & 1 \pm i \end{pmatrix} = \frac{\sqrt{2}}{2} e^{\mp i \frac{\pi}{4}} \begin{pmatrix} \pm i & 1 \\ 1 & \pm i \end{pmatrix} = \frac{\sqrt{2}}{2} e^{\mp i \frac{\pi}{4}} \left[\begin{array}{c} \pm \\ | \end{array} \right]_{q=2} \quad (35)$$

The next case $q = 3$ is proved similarly:

$$\begin{aligned}
 \left[\begin{array}{c} \text{red circle with } \pm 1 \\ \text{red circle with } \pm 1 \end{array} \right] &= \frac{1}{3} \begin{pmatrix} 1 + e^{\pm i \frac{2\pi}{3}} + e^{\pm i \frac{2\pi}{3}} & 1 + \bar{\omega} e^{\pm i \frac{2\pi}{3}} + \omega e^{\pm i \frac{2\pi}{3}} & 1 + \omega e^{\pm i \frac{2\pi}{3}} + \bar{\omega} e^{\pm i \frac{2\pi}{3}} \\ 1 + \omega e^{\pm i \frac{2\pi}{3}} + \bar{\omega} e^{\pm i \frac{2\pi}{3}} & 1 + e^{\pm i \frac{2\pi}{3}} + e^{\pm i \frac{2\pi}{3}} & 1 + \bar{\omega} e^{\pm i \frac{2\pi}{3}} + \omega e^{\pm i \frac{2\pi}{3}} \\ 1 + \bar{\omega} e^{\pm i \frac{2\pi}{3}} + \omega e^{\pm i \frac{2\pi}{3}} & 1 + \omega e^{\pm i \frac{2\pi}{3}} + \bar{\omega} e^{\pm i \frac{2\pi}{3}} & 1 + e^{\pm i \frac{2\pi}{3}} + e^{\pm i \frac{2\pi}{3}} \end{pmatrix} \\
 &= \frac{1}{3} \begin{pmatrix} \sqrt{3} e^{\pm i \frac{\pi}{2}} & \sqrt{3} e^{\mp i \frac{\pi}{6}} & \sqrt{3} e^{\mp i \frac{\pi}{6}} \\ \sqrt{3} e^{\mp i \frac{\pi}{6}} & \sqrt{3} e^{\pm i \frac{\pi}{2}} & \sqrt{3} e^{\mp i \frac{\pi}{6}} \\ \sqrt{3} e^{\mp i \frac{\pi}{6}} & \sqrt{3} e^{\mp i \frac{\pi}{6}} & \sqrt{3} e^{\pm i \frac{\pi}{2}} \end{pmatrix} \\
 &= \frac{\sqrt{3}}{3} e^{\mp i \frac{\pi}{6}} \begin{pmatrix} e^{\pm i \frac{2\pi}{3}} & 1 & 1 \\ 1 & e^{\pm i \frac{2\pi}{3}} & 1 \\ 1 & 1 & e^{\pm i \frac{2\pi}{3}} \end{pmatrix} \\
 &= \frac{\sqrt{3}}{3} e^{\mp i \frac{\pi}{6}} \left[\begin{array}{c} \text{box with } \pm \\ \text{box with } \pm \end{array} \right]_{q=3}
 \end{aligned} \tag{36}$$

For the other qubit case $q = 4$ we first note:

$$\left[\begin{array}{c} \text{yellow square} \\ \text{yellow square} \end{array} \right] = \left[\begin{array}{c} \text{green circle with } \frac{\pi}{2} \\ \text{red circle with } \frac{\pi}{2} \\ \text{green circle with } \frac{\pi}{2} \end{array} \right] = \left[\begin{array}{c} \text{green circle with } \frac{\pi}{2} \\ \text{red circle with } \frac{\pi}{2} \\ \text{green circle with } \frac{\pi}{2} \end{array} \right] = \frac{1}{2\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{37}$$

Then using the standard interpretation for spiders (Definition ??) we decompose the diagram in such a way that we can apply the standard interpretation:

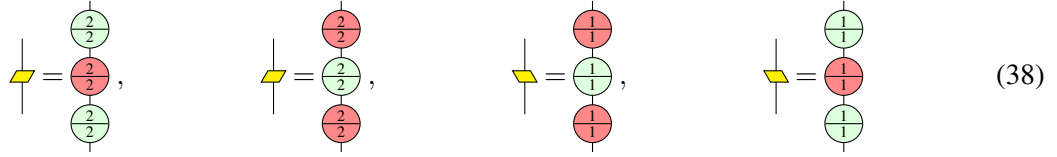
$$\begin{aligned}
 \left[\begin{array}{c} \text{red circle with } \pi \\ \text{yellow square} \\ \text{red circle with } \pi \end{array} \right] &= \left[\begin{array}{c} \text{red circle with } \pi \\ \text{yellow square} \\ \text{red circle with } \pi \end{array} \right] \\
 &= \left(\left[\begin{array}{c} | \\ | \end{array} \right] \otimes \left[\begin{array}{c} \text{red circle with } \pi \end{array} \right] \right) \left(\left[\begin{array}{c} | \\ | \end{array} \right] \otimes \left[\begin{array}{c} \text{yellow square} \end{array} \right] \otimes \left[\begin{array}{c} | \\ | \end{array} \right] \right) \left(\left[\begin{array}{c} \text{red circle with } \pi \end{array} \right] \otimes \left[\begin{array}{c} | \\ | \end{array} \right] \right) \\
 &= \frac{\sqrt{2}}{8} \begin{pmatrix} -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \end{pmatrix} \\
 &= \frac{\sqrt{2}}{8} \left[\begin{array}{c} \text{box with } \pm \\ \text{box with } \pm \end{array} \right]_{q=4}
 \end{aligned}$$

□

C Local Pivot in Qutrit ZX

Next we prove the local pivot equality from Theorem 3.14. For this, recall that all but the **(cm)** and **(cc)** qutrit rewrite rules hold under taking adjoints, and with the roles of green and red swapped, so in

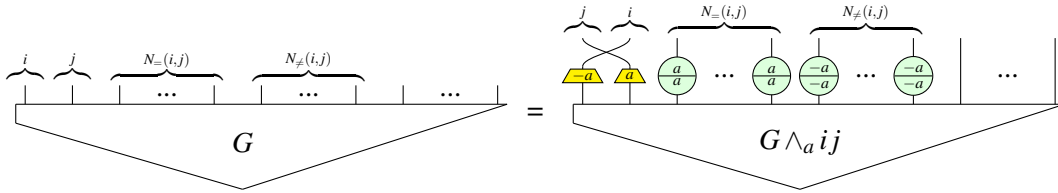
particular rule (E) also gives:



Theorem C.1. / Theorem 3.14. Given $a \in \mathbb{Z}_3$ and a graph state (G, W) containing connected nodes i and j , define the following:

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

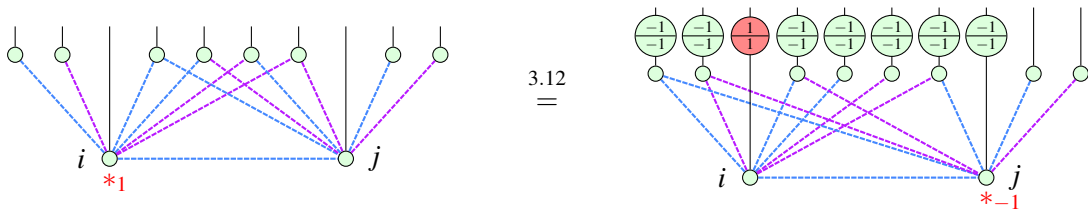
Then the following equation relates G and its proper a -local pivot along ij :

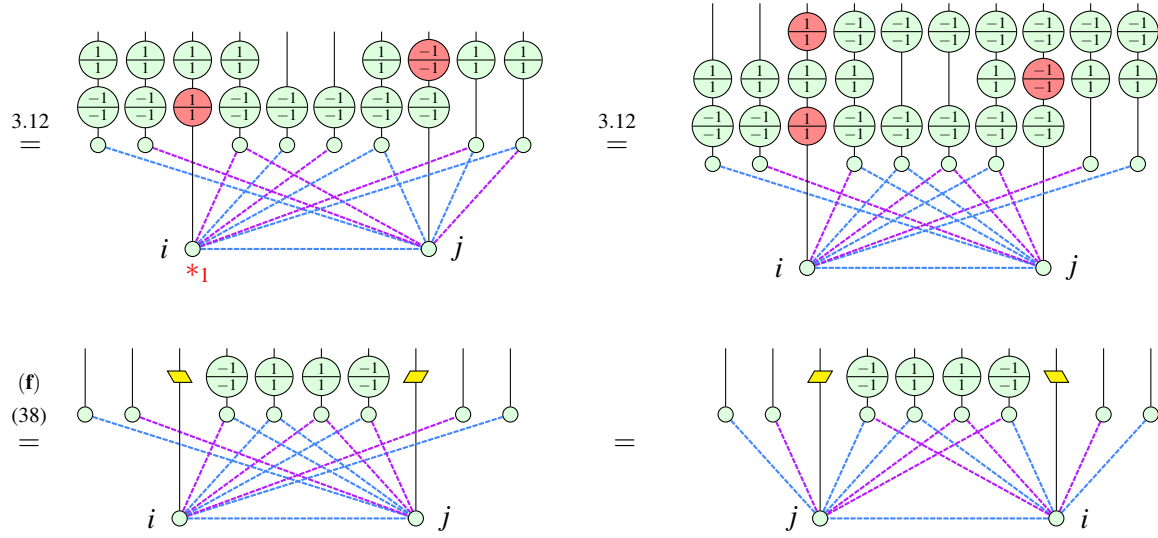


Proof. Annoyingly, proving this in full generality in one go - i.e. for a proper a -local pivot along an edge ij of weight b - becomes a bit tricky diagrammatically, because it becomes hard to keep track of all the variable edge weights. Fortunately the four cases $(a, b \in \{1, 2\})$ split into two pairs of symmetric cases: $a = b$ and $a \neq b$.

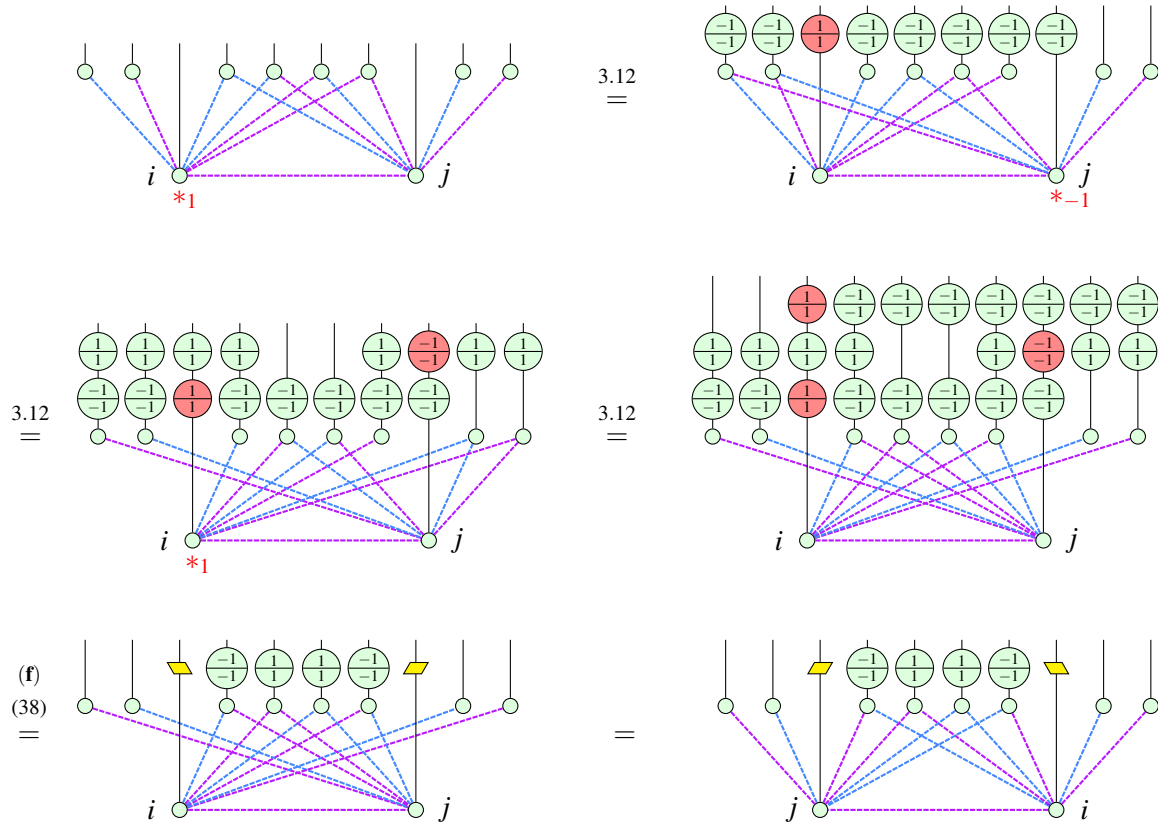
Now, it suffices to only draw a fragment of a graph state. Certainly we consider nodes i and j and the edge ij between them. Then define N_x^y to be the set $\{k \mid w_{k,i} = x, w_{k,j} = y\}$, for $x, y \in \mathbb{Z}_3$. We will consider a representative node k_x^y from each $N_x^y \neq N_0^0$, as well as its edges ik_x^y and jk_x^y . All other nodes and edges are irrelevant; this is because we are only interested in nodes and edges that *affect* the three local complementation operations on i and j - we aren't concerned with those that are only *affected by* the operations.

So for the case $a = b$, we show the proper 1-local pivot along ij of weight 1:





The story is similar for the case $a \neq b$; here we show the proper 1-local pivot along ij of weight 2:



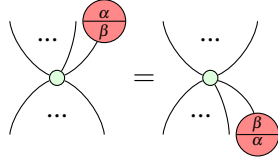
The case $a = b = 2$ is the same as $a = b = 1$, except with the roles of purple and blue edges inter-

changed, so by symmetry of the diagram the only difference is in the phase gates added to the outputs. Namely, on each phase gate we replace all instances of 1 with a 2. Thus the roles of H and H^\dagger are swapped too. Likewise for the case $(a, b) = (2, 1)$ with respect to the case $(a, b) = (1, 2)$. \square

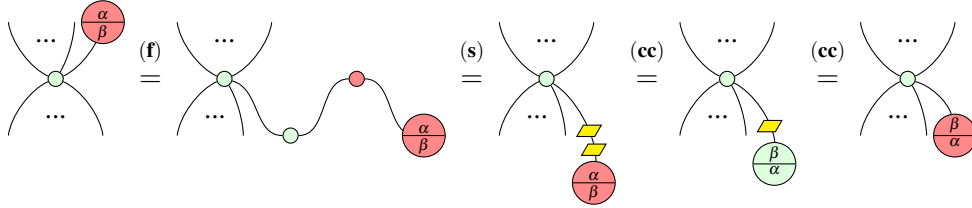
D Spider Elimination Rules for Qutrit Stabiliser ZX

Now we prove the three elimination theorems for \mathcal{M} -, \mathcal{N} - and \mathcal{P} -spiders. First, we require two lemmas.

Lemma D.1. The following ‘leg flip’ equation holds in the qutrit ZX-calculus. Moreover, it holds with the roles of green and red swapped.

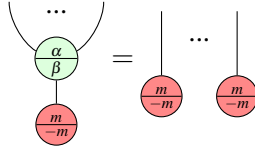


Proof.



\square

Lemma D.2. The following more substantial ‘ \mathcal{M} -copy’ rule holds in the qutrit ZX-calculus, for any \mathcal{M} -spider state (i.e. $m \in \{0, 1, 2\}$ below). Moreover, it holds with the roles of green and red swapped.



Proof. First we prove that an \mathcal{M} -spider with non-trivial phase (i.e. $m \in \{1, 2\}$) satisfies a copy rule exactly like the rule (\mathbf{cp}_0) :

$$\begin{array}{c} \text{...} \\ \diagup \quad \diagdown \\ \text{green spider} \\ \diagdown \quad \diagup \\ \text{red circle } (m, -m) \end{array} \stackrel{(\mathbf{f})}{=} \begin{array}{c} \text{...} \\ \diagup \quad \diagdown \\ \text{green spider} \\ \diagdown \quad \diagup \\ \text{red circle } (m, -m) \end{array} \stackrel{(\mathbf{cp}_{\mathcal{M}})}{=} \begin{array}{c} \text{...} \\ \diagup \quad \diagdown \\ \text{green spider} \\ \diagdown \quad \diagup \\ \text{red circle } (m, -m) \end{array} \stackrel{(\mathbf{cp}_0)}{=} \begin{array}{c} \text{...} \\ \diagup \quad \diagdown \\ \text{green spider} \\ \diagdown \quad \diagup \\ \text{red circle } (m, -m) \end{array} \stackrel{(\mathbf{f})}{=} \begin{array}{c} \text{...} \\ \diagup \quad \diagdown \\ \text{green spider} \\ \diagdown \quad \diagup \\ \text{red circle } (m, -m) \end{array} \quad (39)$$

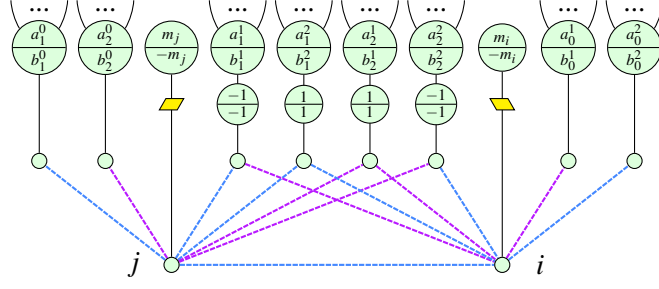
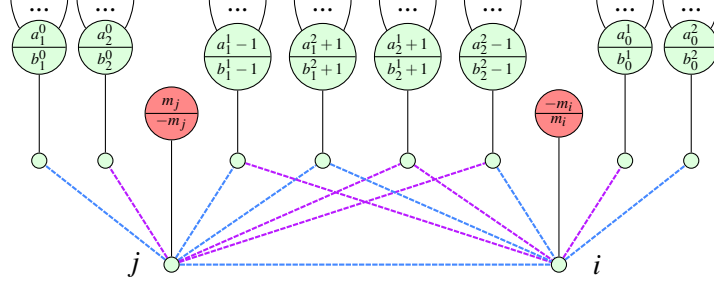
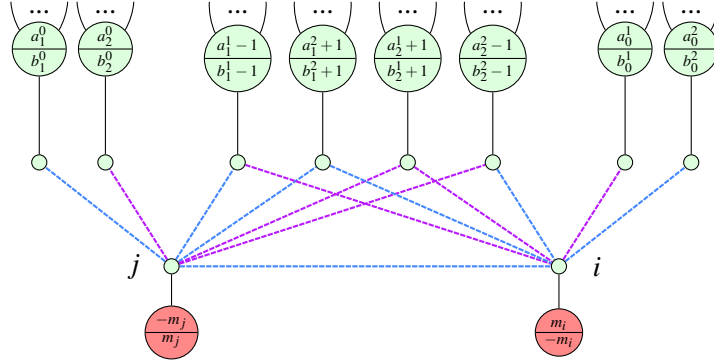
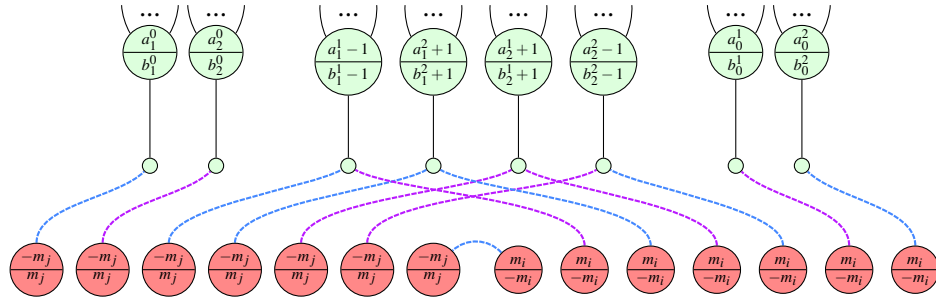
Then we prove the case $\alpha = \beta = 0$ by induction:

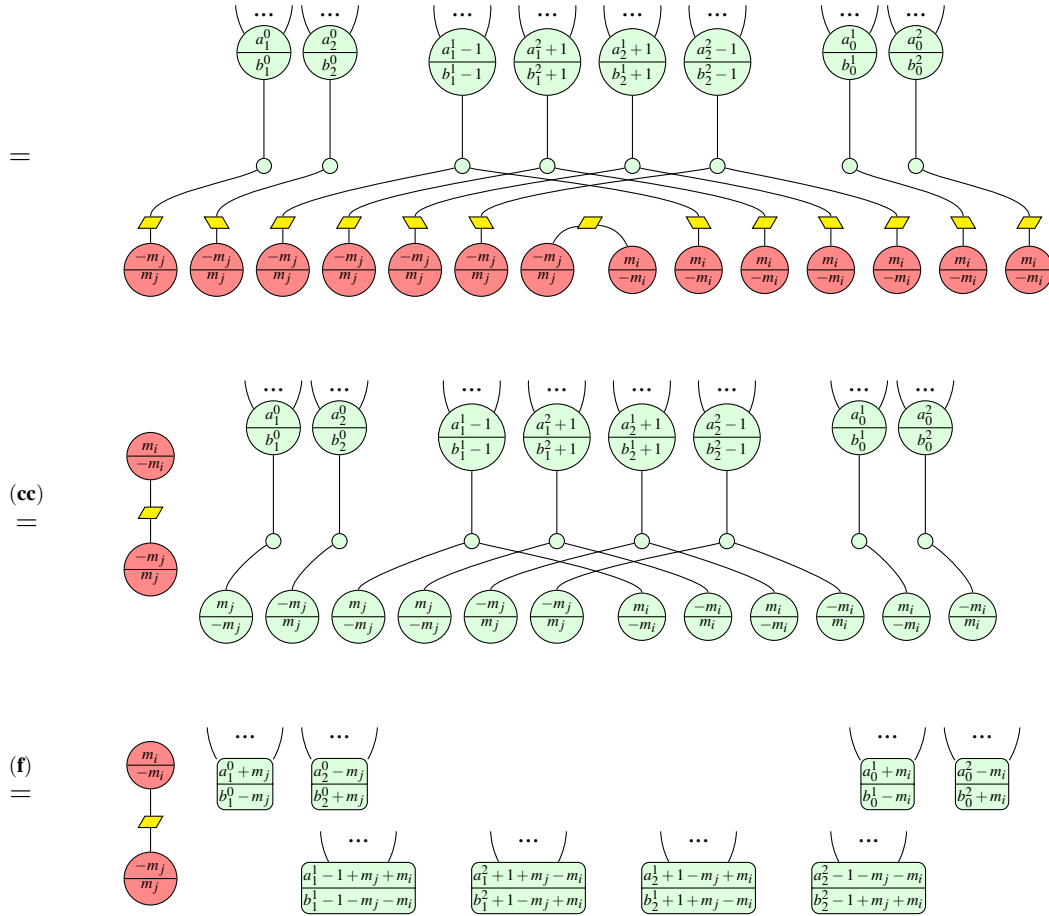
$$\begin{array}{c} \text{...} \\ \diagup \quad \diagdown \\ \text{green spider} \\ \diagdown \quad \diagup \\ \text{red circle } (m, -m) \end{array} \stackrel{(\mathbf{f})}{=} \begin{array}{c} \text{...} \\ \diagup \quad \diagdown \\ \text{green spider} \\ \diagdown \quad \diagup \\ \text{red circle } (m, -m) \end{array} \stackrel{(39)}{=} \begin{array}{c} \text{...} \\ \diagup \quad \diagdown \\ \text{green spider} \\ \diagdown \quad \diagup \\ \text{red circle } (m, -m) \end{array} \stackrel{\text{ind.}}{=} \begin{array}{c} \text{...} \\ \diagup \quad \diagdown \\ \text{green spider} \\ \diagdown \quad \diagup \\ \text{red circle } (m, -m) \end{array} \quad (40)$$

$$\begin{array}{c} \dots \\ \diagup \quad \diagdown \\ \textcircled{\frac{\alpha}{\beta}} \\ | \\ \textcircled{\frac{m}{-m}} \end{array} \quad \text{(f)} \quad = \quad \begin{array}{c} \dots \\ \diagup \quad \diagdown \quad \diagup \\ \textcircled{\phantom{\frac{\alpha}{\beta}}} \\ | \\ \textcircled{\frac{m}{-m}} \end{array} \quad \begin{array}{c} \textcircled{\frac{\alpha}{\beta}} \end{array} \quad (40) \quad = \quad \begin{array}{c} | \quad \dots \quad | \quad \textcircled{\frac{\alpha}{\beta}} \quad | \\ | \quad \quad \quad | \quad \quad \quad | \\ \textcircled{\frac{m}{-m}} \quad \textcircled{\frac{m}{-m}} \quad \textcircled{\frac{m}{-m}} \end{array} = \quad \begin{array}{c} | \quad \dots \quad | \\ | \quad \quad \quad | \\ \textcircled{\frac{m}{-m}} \quad \textcircled{\frac{m}{-m}} \end{array}$$

□

The diagram illustrates the construction of a bipartite graph G from a graph G . The top part shows a graph G with nodes i and j , and various other nodes connected to them. The bottom part shows the construction of G , where nodes i and j are connected by a yellow triangle labeled w , and other nodes are connected to them based on the graph structure.

C.1
=(cc)
=D.1
=D.2
=



□

Proving the corresponding \mathcal{N} -spider elimination theorem requires a further lemma, which allows us to turn an \mathcal{N} -spider state of one colour into an \mathcal{N} -spider state of the other.

Lemma D.4. The following rules hold in the qutrit ZX-calculus:

$$\begin{array}{c} | \\ \hline \frac{0}{1} \end{array} = \begin{array}{c} | \\ \hline \frac{0}{2} \end{array}, \quad \begin{array}{c} | \\ \hline \frac{0}{2} \end{array} = \begin{array}{c} | \\ \hline \frac{0}{1} \end{array}, \quad \begin{array}{c} | \\ \hline \frac{1}{0} \end{array} = \begin{array}{c} | \\ \hline \frac{0}{2} \end{array}, \quad \begin{array}{c} | \\ \hline \frac{2}{0} \end{array} = \begin{array}{c} | \\ \hline \frac{1}{0} \end{array}$$

Proof. The key observation is that for any green \mathcal{N} -spider state with phase $\frac{n}{n'}$, we have a choice of two colour change rules which we could use to turn it into a red \mathcal{N} -spider state with a H - or H^\dagger -box on top:

$$\begin{array}{c} | \\ \hline \frac{n}{n'} \end{array} \xrightarrow{(cc)} \begin{array}{c} | \\ \hline \frac{n}{n'} \end{array} \xrightarrow{(cc)} \begin{array}{c} | \\ \hline \frac{n'}{n} \end{array}$$

Of these two choices, exactly one has a decomposition of the H -/ H^\dagger -box as in (38) that allows the bottom two red spiders to fuse into an \mathcal{M} -spider, which we can then move past the green spider above it via D.2. For brevity we only show the case $\frac{n}{n'} = \frac{0}{1}$:

□

Corollary D.5. The following equations hold in the qutrit ZX-calculus:

Proof. Again we only prove one case, the rest being analogous. Each case uses D.4 in its adjoint form - recall that the adjoint of a spider is found by swapping inputs and outputs and negating angles.

□

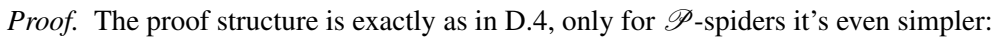
Theorem D.6. / Theorem 3.16. Given any graph-like ZX-diagram containing an interior \mathcal{N} -spider k with phase $\frac{0}{n}$ for $n \in \{1, 2\}$, suppose we perform a $(-n)$ -local complementation at k . Then the new ZX-diagram is related to the old one by the equality:

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:

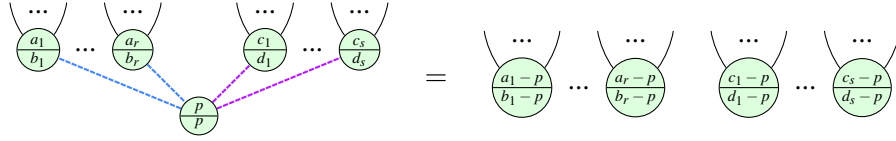
Proof. We prove the case where k has phase $\frac{0}{n}$ for $n \in \{1, 2\}$, the other case being near-identical.



Lemma D.7. The following rule holds in the qutrit ZX-calculus:

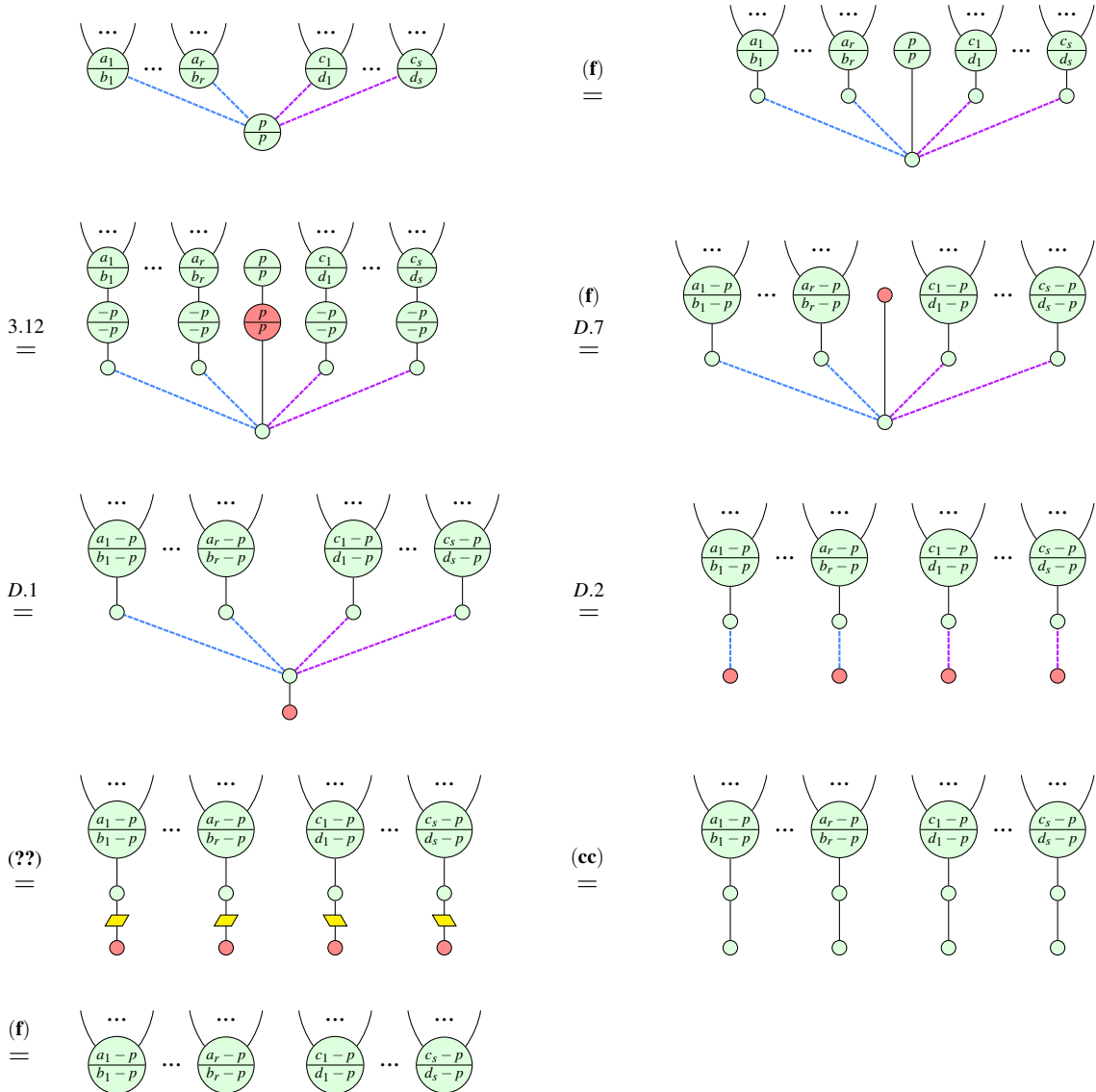


Theorem D.8. / Theorem 3.17. Given any graph-like ZX-diagram containing an interior \mathcal{P} -spider k with phase $\frac{p}{p}$ for $p \in \{1, 2\}$, suppose we perform a p -local complementation at k . Then the new ZX-diagram is related to the old one by the equality:



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

Proof.



□