

Qudit ZX-Diagram Simplification

Candidate Number: 1025724

Abstract

The majority of quantum literature focuses on two-dimensional qubit systems. Recently, advantages have been found in using higher-dimensional systems. The qudit *ZX-calculus* is a graphical language which allows for reasoning about suitably represented *string diagrams* – called *ZX-diagrams* – in terms of *rewrite rules*. *ZX-diagrams* in a particular subset of this calculus – the *stabilizer fragment* – are known by the Gottesman-Knill theorem to be *efficiently simplifiable*. The rewrite rules integral to this are known for the qubit case. Here we derive the corresponding rules for the qudit case, before generalising to dimension d . We are able to completely characterise the stabilizer fragment of the qudit *ZX-calculus* in terms of *spider phases*, and we discuss what further results are required to demonstrate efficient simplification of qudit stabilizer diagrams. These results have ramifications in many areas, chiefly among them quantum circuit optimisation. However, in this work we instead use these results to emphasise the usefulness of the *ZX-calculus* outside of quantum computing. We look at problems which amount to exactly computing a scalar encoded as a *closed tensor network*. In general, such problems are $\#\mathbf{P}$ -hard. However, there are families of such problems which are known to be in \mathbf{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 stabilizer fragment of the *ZX-calculus*. We look at the specific examples of evaluating the *Jones polynomial* and of counting *graph colourings*.

Declaration

The final word count is 9967. A subset of this work was packaged into a joint paper with my co-supervisor; this was submitted for publication and appears on arXiv.

Contents

| | | |
|----------|--|-----------|
| 1 | Introduction | 4 |
| 1.1 | History | 4 |
| 1.2 | The ZX-Calculus | 5 |
| 1.3 | Contribution | 6 |
| 2 | The Qudit ZX-Calculus: Part One | 8 |
| 3 | Simplifying Qubit ZX-Diagrams | 16 |
| 3.1 | The Qubit ZX-Calculus | 16 |
| 3.2 | Stabilizer Qubit ZX Diagrams | 18 |
| 4 | Simplifying Qutrit ZX-Diagrams | 20 |
| 4.1 | The Qutrit ZX-Calculus | 20 |
| 4.2 | Graph-Like Qutrit ZX Diagrams | 22 |
| 4.3 | Qutrit Elimination Theorems | 26 |
| 5 | Qudit ZX-Calculus: Part Two | 30 |
| 5.1 | The Stabilizer Fragment | 30 |
| 5.2 | Local Complementation and Pivot | 35 |
| 5.3 | Elimination Theorems | 38 |
| 6 | Case studies | 42 |
| 6.1 | The Jones Polynomial at Lattice Roots of Unity | 42 |
| 6.2 | Graph Colouring | 46 |
| 7 | Outlook | 48 |
| A | Local Complementation | 53 |
| B | Eliminating $\langle \pm 1, x \rangle$-Spiders | 60 |
| C | Pivoting in the Qutrit ZX-Calculus | 65 |
| D | Eliminating $\langle 0, 2x \rangle$-Spiders | 70 |
| E | \pm-Boxes in the ZX-Calculus | 78 |

1 Introduction

1.1 History

The introduction of the *ZX-calculus* in Ref. [10] tied together two strands of history. In the 1930s, von Neumann tackled problems in quantum mechanics using a *Hilbert space* – a complex vector space equipped with an inner product [49]. This formalism became – and remains to this day – the dominant mode of reasoning about quantum mechanics, and it has enabled much progress to be made. For example, in 1993, researchers found that under certain conditions it is possible to ‘teleport’ a quantum state from one place to another [5]. The question asked in Coecke and Kissinger’s quantum mechanics textbook *Picturing Quantum Processes* [11] is: why did this fairly simple feature take sixty years to discover? Therein, it is posited that the Hilbert space framework itself might be culpable; the authors suggest that linear algebraic manipulation is unintuitive and clunky, and hinders rather than helps those working with it. Compellingly, the ranks of those who do not champion the Hilbert space contains one particularly influential member: its populariser, von Neumann himself. He later said: “I would like to make a confession which may seem immoral: I do not believe absolutely in Hilbert space no more.” [44].

Meanwhile, the 1940s witnessed the birth of a new branch of mathematics, called category theory [22]. This field shifted the focus away from mathematical objects (sets, spaces, groups, etc) and onto the relationships between such objects (functions, linear maps, homomorphisms). In category theory speak, these are termed *objects* and *morphisms* respectively, and – when combined with a few axioms – constitute a *category* [2]. All of this can be presented in terms of *string diagrams*. These diagrams, which are here read bottom-to-top, consist of boxes (which represent morphisms) connected by wires (which represent objects). Two diagrams are equivalent if one can be ‘deformed’ into another in some well-defined sense [29]. Since humans have an innate proficiency for such deformations, string diagrams are incredibly intuitive and fruitful to work with, yet don’t compromise on rigour. An illuminating example of their power is a proof that the trace of a matrix is cyclic; that is, $\text{Tr}(AB) = \text{Tr}(BA)$. Without string diagrams this would require a few lines of fairly opaque algebra, but diagrammatically we need only slide a box around a wire, like a bead on a necklace. Rigourously setting up the semantics here is beyond the scope of this introduction: the point is only that this proof is on some level intuitive and obvious:

$$\text{Tr}(AB) = \begin{array}{c} \text{---} \\ \text{---} \end{array} \begin{array}{|c|} \hline A \\ \hline B \\ \hline \end{array} \begin{array}{c} \text{---} \\ \text{---} \end{array} = \begin{array}{c} \text{---} \\ \text{---} \end{array} \begin{array}{|c|} \hline A \\ \hline \end{array} \begin{array}{c} \text{---} \\ \text{---} \end{array} \begin{array}{|c|} \hline B \\ \hline \end{array} \begin{array}{c} \text{---} \\ \text{---} \end{array} = \begin{array}{c} \text{---} \\ \text{---} \end{array} \begin{array}{|c|} \hline A \\ \hline \end{array} \begin{array}{c} \text{---} \\ \text{---} \end{array} \begin{array}{|c|} \hline B \\ \hline \end{array} \begin{array}{c} \text{---} \\ \text{---} \end{array} = \begin{array}{c} \text{---} \\ \text{---} \end{array} \begin{array}{|c|} \hline B \\ \hline A \\ \hline \end{array} \begin{array}{c} \text{---} \\ \text{---} \end{array} = \text{Tr}(BA) \quad (1)$$

1.2 The ZX-Calculus

The ZX-calculus seeks to unshackle quantum mechanics from the unwieldy linear algebraic foundations first developed by von Neumann, and instead replace these with the intuitive yet rigorous diagrammatic approach developed by category theorists. It is a graphical language that allows for reasoning about *ZX-diagrams*: string diagrams which are expressed in terms of primitive generators, called red and green (or *X*- and *Z*-) spiders [48]. The generators have a concrete interpretation as matrices over a (semi)ring and they obey a set of *rewrite rules* which *respect semantics* [52]. That is, the diagrams on the left and right hand side of a rewrite rule have equal concrete interpretations as matrices, up to a scalar. When this (semi)ring is the field \mathbb{C} of complex numbers, the wires represent Hilbert spaces and the spiders linear maps between them, so we are just using a different framework to work with von Neumann’s original structures. A computer science-themed analogy given in *Picturing Quantum Processes* [11] is that von Neumann’s approach of working directly with Hilbert spaces is akin to coding in very a low-level language, like machine code, whereas the ZX-calculus constitutes a high-level language, like Python.



Figure 1: Examples of rewrite rules that hold in the qubit ZX-calculus. Graphical algebraists may recognise the first of these as declaring that the red and green spiders form a *bialgebra*. Similarly, they may recognise that the second equation says the yellow box (*Hadamard gate*) is a *Frobenius transport* between the red and green spiders.

Key fields in which the the ZX-calculus has found applications include quantum circuit optimisation and compilation [20, 35, 14, 15, 12], measurement-based quantum computing [19, 34], quantum error-correcting codes [8, 16], (semi-)automated proof assistance [36, 25], quantum machine learning [47], and even natural language processing [39, 9]. The first of these particularly are where the ZX-calculus has been most convincingly able to refute a charge often levelled at it: that diagrammatic calculi don’t help to prove anything new, but rather only repackage known results. Indeed, the work we present here is particularly inspired by Ref. [20], in which a ZX-calculus-based quantum circuit optimisation algorithm is devised and shown to have state-of-the-art performance.

However, the ZX-calculus is not limited to quantum mechanical settings. A plethora of hard problems of interest to physics and computer science reduce to exactly computing a single scalar. Examples include partition functions in classical statistical mechanics, counting problems, probabilistic inference, and many more. A common method of computing a scalar of interest is to encode it as a *closed tensor network* – a type of string diagram whose boxes represent linear maps – and then perform *full tensor contraction*. This involves

repeatedly performing a particular linear algebraic operation on pairs of connected boxes in the diagram – graphically this fuses the two boxes together to create a single box – until only one box remains. This box represents the desired scalar. In general this is $\#\mathbf{P}$ -hard [13], and in fact, finding the optimal contraction path for a general tensor network is NP-hard.

As we will see, a ZX-diagram can be interpreted as a tensor network, thus we could instead encode our scalar as a ZX-diagram and perform a sequence of simplifying rewrite rules, until the diagram is sufficiently simple as to be able to just ‘read off’ the scalar it represents. Note that given an arbitrary tensor network, one could attempt to invent rewrite rules by inspecting the contents of the tensors and performing linear algebraic operations [28]. The advantage of leveraging the ZX-calculus for such problems is that the rewrite rules can immediately be used out-of-the-box. For example, as a proof-of-concept demonstration of these ideas, ZX-diagrams have been used to give novel graphical proofs of complexity results in boolean satisfiability (SAT) and model counting ($\#\text{SAT}$) [17].

1.3 Contribution

The ZX-calculus, which we introduce in Section 2, concerns d -state quantum systems, for some fixed d . Throughout the quantum computing literature, *qubit* systems (the case $d = 2$) are by far the most commonly studied. Of late, higher-dimensional systems have been gaining prominence, due to work suggesting they give efficiency gains in areas like circuit construction [6] and magic state distillation [30]. In Section 3 we recall some of the work of Ref. [20], then in Section 4 we generalise it to the *qutrit* case ($d = 3$), where the ZX-calculus takes on a slightly stranger, more complex flavour. We derive a qutrit *pivot rule* and prove three *elimination theorems*, which we combine into an algorithm for efficiently simplifying certain ZX-diagrams. Effectively, this constitutes one half of a qutrit circuit optimisation algorithm: the remaining work would involve re-extracting a circuit from our algorithm’s output. Given that the results we are generalising lead to a state-of-the-art qubit circuit optimisation algorithm, it is not unreasonable to expect to obtain a similarly-performing algorithm in the qutrit case. As a further corollary, all of this slightly generalises a qutrit version the famous Gottesman-Knill theorem, which concerns conditions under which a quantum circuit is efficiently classically simulable.

In Section 5 we expand our scope to general qudits, where we completely characterise a subset of the qudit ZX-calculus – called the *stabilizer fragment* – in terms of *spider phases*. Though this result does not have the same immediate practical applications as the elimination theorems we derive, it is equally as important. We formulate and go a long way to proving a local complementation rule, from which we derive a pivot rule and higher-dimensional elimination theorems. We comment on what further results are required to be able to fully simplify any stabilizer qudit ZX-diagram.

In Section 6 we close by emphasising the applications of the ZX-calculus – and our new

elimination rules in particular – to problems outside of quantum computing. To this end, we present two case studies. First, we provide a novel graphical proof of an established result from pure topology, namely that the *Jones polynomial* of any *link* can be efficiently evaluated at the *lattice roots of unity*. We then graphically recover a known complexity result on counting graph-colourings. Both of these examples reduce to evaluating a *closed* ZX-diagram, and show a transition in complexity at a particular dimension d , below which the diagram is in the stabilizer fragment. Here we open ourselves up to the complaint mentioned earlier: that graphical calculi are only good for rederiving known results. As a counter to this, we stress that observations in this section are effectively just toy examples, included only to demonstrate that the ZX-calculus has uses beyond quantum computing. Our point is broadly this: a large number of important (and perhaps unexpected) problems can be cast as ZX-diagrams; the techniques in this paper then constitute a new tool with which to tackle them.

2 The Qudit ZX-Calculus: Part One

At its most formal, the ZX-calculus is defined categorically. A good reference for this is [48], and a much deeper treatment is found in [29]. Since the foundational categorical aspects of the ZX-calculus are not the focus of this work, and we assume familiarity with the courses *Categorical Quantum Mechanics* and *Quantum Processes and Computation*, we favour a more relaxed approach. However, here we give a quick overview of the category-theoretic underpinnings. One of many equivalent ways to proceed is to define a category **ZX**, whose objects are the natural numbers, and whose morphisms are the following diagrams:

$$\begin{array}{c} \overbrace{\quad\quad\quad}^m \\ \vdots \\ \text{green circle with } \tilde{\alpha} \\ \vdots \\ \underbrace{\quad\quad\quad}_n \end{array}, \begin{array}{c} \overbrace{\quad\quad\quad}^m \\ \vdots \\ \text{red circle with } \alpha \\ \vdots \\ \underbrace{\quad\quad\quad}_n \end{array}, \quad |, \quad \begin{array}{c} \diagup \\ \diagdown \end{array} \quad (2)$$

We call the first two diagrams *green* and *red* (or *Z*- and *X*-) *spiders* respectively, and we call the $(d - 1)$ -dimensional vectors which decorate them *phases*. At first glance, directly defining the morphisms using diagrams may seem very strange – it is natural to expect these diagrams to be only a shorthand for something more ‘concrete’ or ‘mathematical’, such as a function or linear map. But this is exactly the power of category theory’s generality: provided we can compose morphisms, and this composition operation has a unit (an *identity morphism*) for each object, our morphisms are well-defined. If one is more comfortable with symbols, one could instead write the morphisms depicted above as (for example) $Z_{\tilde{\alpha}}^{m,n}$, $X_{\alpha}^{m,n}$, id_1 and σ respectively, then (2) denotes their representation as string diagrams, as described in the introduction. But these symbols would still have no ‘concrete’ interpretation, and since we always use graphical manipulation as opposed to symbolic, we cut out the middleman and directly define morphisms via diagrams.

In the case of the ZX-calculus, morphism composition (\circ) means vertically stacking compatible diagrams and fusing wires together:

$$\left(\begin{array}{c} | \\ \text{red circle with } \alpha \\ \diagdown \quad \diagup \end{array} \right) \circ \left(\begin{array}{c} \diagup \quad \diagdown \\ \text{green circle with } \beta \\ | \end{array} \right) = \begin{array}{c} | \\ \text{red circle with } \alpha \\ \diagdown \quad \diagup \\ \text{green circle with } \beta \\ | \end{array} \quad (3)$$

The identity morphism id_m on a object (i.e. natural number) m is just m ‘plain’ wires side-by-side:

$$id_m = \begin{array}{c} \overbrace{\quad\quad\quad}^m \\ \left| \quad \left| \quad \dots \quad \left| \right. \right. \end{array} \quad (4)$$

$$\tilde{\alpha}_j = \vec{\alpha}_{d-j} \quad (11)$$

$$\vec{\alpha}_j^k = \vec{\alpha}_{j-k} - \vec{\alpha}_{-k} \quad (12)$$

The *ZX-calculus* then just broadly refers to our ability to manipulate ZX-diagrams within **ZX** under these equations. We use special notation of either a yellow box or a blue dashed line for the *Hadamard gate*, which is defined to be:

$$\begin{array}{c} | \\ \boxed{1} \\ | \end{array} = \begin{array}{c} | \\ \text{---} \\ | \end{array} = \begin{array}{c} | \\ \textcircled{\tau} \\ \textcircled{\tau} \\ \textcircled{\tau} \\ | \end{array} \quad (13)$$

This distinct notation emphasises its unique role in the ZX-calculus: it alone allows us to switch between green spiders and red spiders, according to the colour change equations (**cc**) in Figure 2. Similarly we use special notation suggestive of two Hadamard gates squished together for the *dualiser* defined in the snake rule (**s**):

$$\begin{array}{c} | \\ \boxed{1} \\ | \end{array} := \begin{array}{c} | \\ \boxed{1} \\ | \\ \boxed{1} \\ | \end{array} = \begin{array}{c} | \\ \boxed{-1} \\ | \\ \boxed{-1} \\ | \end{array} = \begin{array}{c} \textcircled{\tau} \\ \vdots \\ \textcircled{\tau} \end{array} = \begin{array}{c} \textcircled{\tau} \\ \vdots \\ \textcircled{\tau} \end{array} \quad (14)$$

We stress that these rules are not necessarily a minimal set, but are a convenient presentation. Moreover, they are *complete* for the *stabilizer fragment* of the ZX-calculus. We will define the stabilizer fragment later, but what this means is the following: for any two stabilizer ZX-diagrams whose images under $\llbracket - \rrbracket$ are equal up to a scalar, there is a finite sequence of diagrammatic equalities that transforms one diagram to the other.

All of these rules hold under taking adjoints of both sides. Furthermore, it can be proved that rules (**f**), (**id**), (**b**), (**z**), (**E**), (**s**) all continue to hold when the roles of red and green are interchanged. There are analogous rules to (κ) and (**cc**) that hold when red and green are swapped; an example is shown below. However, no analogue exists for the green cup rule (**g**).

$$\begin{array}{c} \textcircled{\alpha} \\ \vdots \\ \textcircled{\alpha} \end{array} \stackrel{(\text{cc})}{=} \begin{array}{c} \boxed{1} \quad \dots \quad \boxed{1} \\ \vdots \\ \boxed{-1} \quad \dots \quad \boxed{-1} \end{array} \stackrel{(\text{H})}{=} \begin{array}{c} \textcircled{\alpha} \\ \vdots \\ \textcircled{\alpha} \end{array} \quad (15)$$

So far, so abstract, but quantum mechanics takes place in the arena of Hilbert spaces – how do we turn ZX-diagrams into something more concrete? Category theory contains the notion of a *functor*, which lets us map between two categories. The category **Hilb** has

as objects Hilbert spaces and as morphisms complex linear maps. We can thus define a functor $\llbracket - \rrbracket : \mathbf{ZX} \rightarrow \mathbf{Hilb}$ that interprets any ZX-diagram as a linear map between Hilbert spaces. Let $\omega = e^{i\frac{2\pi}{d}}$, and pick orthonormal vectors $\{|0\rangle, \dots, |d-1\rangle\}$ to be a basis of \mathbb{C}^d . We call this the *Z-basis* or *computational basis*. We define the *X-basis* to be $\{|\omega_0\rangle, \dots, |\omega_{d-1}\rangle\}$, where:

$$|\omega_j\rangle = \frac{1}{\sqrt{d}} \sum_{k=0}^{d-1} \omega^{jk} |k\rangle \quad (16)$$

Our desired functor, defined on generating morphisms of \mathbf{ZX} , is then:

$$\left[\begin{array}{c} \overbrace{\quad}^m \\ \vdots \\ \text{---} \alpha \text{---} \\ \vdots \\ \underbrace{\quad}_n \end{array} \right] = \sum_{j=0}^{d-1} e^{i\vec{\alpha} \cdot j} |j\rangle^{\otimes m} \langle j|^{\otimes n} \quad (17)$$

$$\left[\begin{array}{c} \overbrace{\quad}^m \\ \vdots \\ \text{---} \vec{\alpha} \text{---} \\ \vdots \\ \underbrace{\quad}_n \end{array} \right] = \sum_{j=0}^{d-1} e^{i\vec{\alpha} \cdot j} |\omega_j\rangle^{\otimes m} \langle \omega_j|^{\otimes n} \quad (18)$$

$$\left[\begin{array}{c} | \\ | \end{array} \right] = id \quad (19)$$

$$\left[\begin{array}{c} \diagup \quad \diagdown \\ \diagdown \quad \diagup \end{array} \right] = (u \otimes v \mapsto v \otimes u) \quad (20)$$

In particular, wires are interpreted under $\llbracket - \rrbracket$ as complex vector spaces \mathbb{C}^d of dimension d . This functor has the property that two diagrams are equal in \mathbf{ZX} if and only if their images under $\llbracket - \rrbracket$ are equal up to a scalar in \mathbf{Hilb} . Since in this work we are ultimately only concerned about making statements about complexity, this suffices. The reason we don't demand exact equality is because such scalars are often simply not important in quantum mechanical applications. Nonetheless, a scalar-exact formulation of the ZX-calculus is available, and widely-used [3]. Importantly, using this formulation, keeping track of multiplicative scalar factors that arise under diagram rewriting can be done efficiently. Omitting them for the purposes of this work therefore doesn't invalidate any of our claims about complexity.

Under $\llbracket - \rrbracket$, parallel composition becomes the matrix tensor product, and sequential com-

position becomes matrix multiplication:

$$\llbracket \begin{array}{c} \text{---} \alpha \text{---} \beta \text{---} \\ \diagup \quad \diagdown \end{array} \rrbracket = \llbracket \begin{array}{c} \text{---} \alpha \text{---} \\ \diagup \end{array} \rrbracket \otimes \llbracket \begin{array}{c} \text{---} \beta \text{---} \\ \diagdown \end{array} \rrbracket, \quad \llbracket \begin{array}{c} \text{---} \alpha \text{---} \\ \diagup \quad \diagdown \\ \text{---} \beta \text{---} \end{array} \rrbracket = \llbracket \begin{array}{c} \text{---} \alpha \text{---} \\ \diagup \end{array} \rrbracket \llbracket \begin{array}{c} \text{---} \beta \text{---} \\ \diagdown \end{array} \rrbracket \quad (21)$$

It is under this functor $\llbracket - \rrbracket$ that a ZX-diagram can be seen as a *tensor network*. To a category theorist, a tensor network is simply a string diagram in the category $\mathbf{Mat}_{\mathbb{F}}$, wherein morphisms are linear maps and objects are vector spaces \mathbb{F}^d , for some field \mathbb{F} . When $\mathbb{F} = \mathbb{C}$ this category differs from \mathbf{Hilb} only in that no inner product is given distinguished status, we can trivially view a complex tensor network as living in \mathbf{Hilb} instead.

In categorical terms, every spider is a morphism from one object $m \in \mathbb{N}$ to another object $n \in \mathbb{N}$. We say that this spider has m *input* wires and n *output* wires. It will generally be important to distinguish between these two types of wires; we fix a convention that any wires incident to the bottom half of a spider are inputs, and any incident to the top half are outputs:


(22)

As an example of where this distinction is important, suppose a wire is simultaneously an input leg of a green spider and an output leg of a red spider. Then the snake rule (**s**) says that a dualiser penalty is incurred when we try to move the spiders such that this wire becomes an input leg of the red spider and output leg of the green one:

$$\begin{array}{c} \dots \\ \diagup \quad \diagdown \\ \alpha \quad \beta \\ \diagdown \quad \diagup \\ \dots \end{array} \stackrel{(f)}{=} \begin{array}{c} \dots \\ \diagup \quad \diagdown \\ \alpha \quad \beta \\ \diagdown \quad \diagup \\ \dots \end{array} \stackrel{(s)}{=} \begin{array}{c} \dots \\ \diagup \quad \diagdown \\ \alpha \quad \beta \\ \diagdown \quad \diagup \\ \dots \end{array} \quad (23)$$

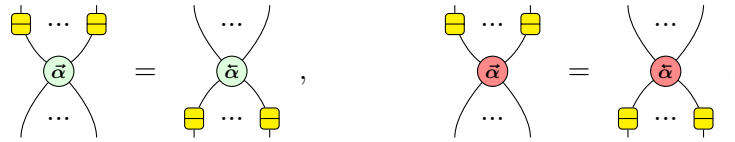
Provided this input/output distinction is respected, we can otherwise move spiders around the plane freely and the diagram remains equivalent; during proofs, we denote this by *iso*. (for *isotopy*). A string diagram can also have ‘open-ended’ wires, wherein either one or both endpoints are not incident to any morphism, but instead simply stop at either the bottom or top of the plane we’re drawing in. Such wires are called *input* and *output* wires (respectively) of the whole diagram. Occasionally this terminology can be confusing; for

example, the leftmost wire below is an input wire to the spider but an output wire of the overall diagram:

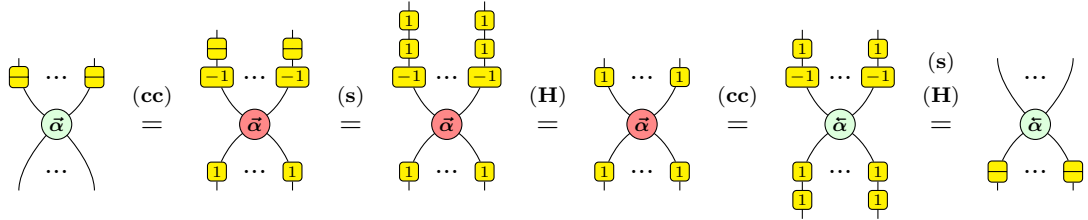

(24)

Where potential confusion arises, we will take care to be verbose about what we mean. To get the reader familiar with the kind of diagrammatic manipulation we will use throughout this work, we prove here two very useful lemmas:

Lemma 2.1. The following *dualiser rules* are derivable in the qudit ZX-calculus:


(25)

Proof. We just prove the green spider case - the red spider case is near-identical.

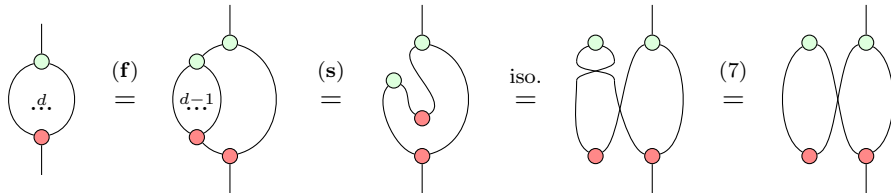


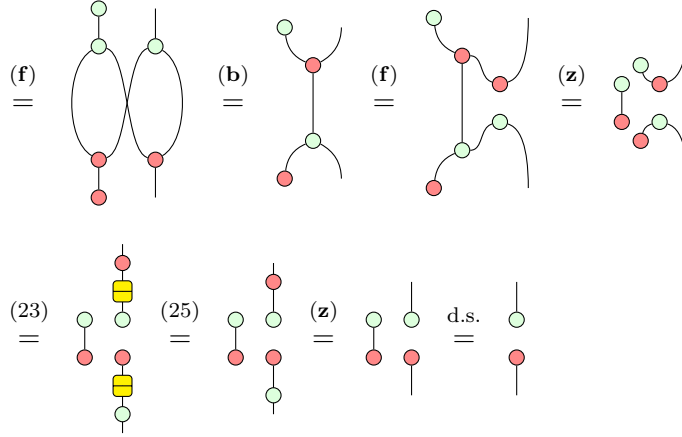
□

Lemma 2.2. The following *Hopf rule* is derivable in the qudit ZX-calculus:


(26)

Proof.





□

The last equality is justified as follows: the diagram $\begin{array}{c} \circ \\ | \\ \circ \end{array}$ is a phaseless green *effect* $\begin{array}{c} \circ \\ | \end{array}$ composed with a phaseless red *state* $\begin{array}{c} | \\ \circ \end{array}$. Under $\llbracket - \rrbracket$, this corresponds to a $1 \times d$ row vector multiplied by a $d \times 1$ column vector – this gives a scalar. Since we said equality as ZX-diagrams is exactly equality up to a scalar under $\llbracket - \rrbracket$, we can disregard $\begin{array}{c} \circ \\ | \\ \circ \end{array}$. In fact, any diagram with no inputs and no outputs can be disregarded: we call this *dropping a scalar* (d.s.).

3 Simplifying Qubit ZX-Diagrams

In this section we restrict the definition of the qudit ZX-calculus to the simplest non-trivial case $d = 2$, and highlight the properties specific to this case that make the calculus particularly easy to work with. We recall the definitions of *graph-like* diagrams and *stabilizer* diagrams. Crucially, we also recall the *elimination theorems* that enable the efficient simplification of stabilizer diagrams, and which lead to a state-of-the-art circuit optimisation algorithm. None of the work in this section is new, but it is helpful to briefly restate it here before we generalise it to higher dimensions.

3.1 The Qubit ZX-Calculus

The full, simplified qubit ZX-calculus rules are given in Figure 3. The snake rule (s) has a particularly simple form for qubits:

$$\begin{array}{c} \text{green circle} \\ \text{red circle} \end{array} \text{ snake} = \begin{array}{c} \text{red circle} \\ \text{green circle} \end{array} = \begin{array}{c} \boxed{1} \\ \boxed{1} \end{array} = \begin{array}{c} \boxed{-1} \\ \boxed{-1} \end{array} \quad (27)$$

By the (id) rule, the second diagram above is just the identity wire. As a consequence, the following identities hold in the qubit ZX-calculus.

$$\boxed{1} = \boxed{-1}, \quad \text{cup} = \text{cup with green circle} = \text{cup with red circle} \quad (28)$$

Proof. For the first equation:

$$\boxed{1} = \begin{array}{c} \boxed{1} \\ | \end{array} \stackrel{(27)}{=} \begin{array}{c} \boxed{1} \\ \boxed{-1} \\ \boxed{-1} \end{array} \stackrel{(\mathbf{H})}{=} \boxed{-1}$$

For the second:

$$\text{cup with green circle} = \text{cup with green circle and red circle} \stackrel{(27)}{=} \text{cup with green circle and red circle} \stackrel{(\mathbf{f})}{=} \text{cup with green circle and red circle} \stackrel{(\mathbf{id})}{=} \text{cup with red circle}$$

□

The first of these results permits us to just use an empty yellow box to denote the Hadamard gate:

$$\boxed{1} = \boxed{-1} =: \boxed{} \quad (29)$$

The second endows the qubit ZX-calculus with a crucial property, referred to as *Only Connectivity Matters*. That is, suppose we place some green and red spiders somewhere in the plane, and connect them however we like. We may now move these spiders around the plane freely, and, provided the connections between spiders remain invariant, our ZX-diagram remains equivalent. One way to see this is to note that the dualiser in Equation (23) is now just the identity wire. In particular, we do not have to worry about which of a spider's legs are output legs and which are inputs.

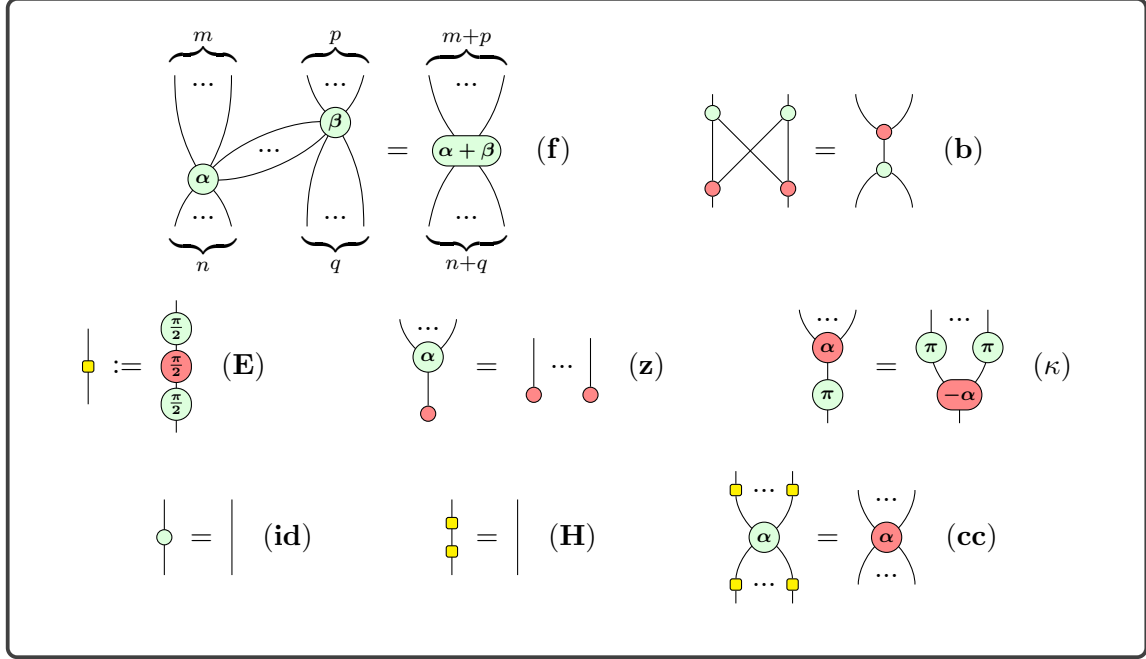


Figure 3: Qubit ZX-calculus rules. The fusion rule (f) only holds if the two spiders are connected by at least one wire. α and β are taken mod 2π . Using (cc), it can be shown that all of these hold when the roles of green and red are swapped.

The *standard interpretation* of the spider generators as tensors over \mathbb{C} (i.e. the functor $\llbracket - \rrbracket$) becomes:

$$\left[\left[\begin{array}{c} \overbrace{\quad \quad \quad}^m \\ \vdots \\ \text{green spider } \alpha \\ \vdots \\ \underbrace{\quad \quad \quad}_n \end{array} \right] \right] = |0\rangle^{\otimes m} \langle 0|^{\otimes n} + e^{i\alpha} |1\rangle^{\otimes m} \langle 1|^{\otimes n} \quad (30)$$

$$\left[\begin{array}{c} \overbrace{\quad\quad\quad}^m \\ \dots \\ \textcolor{red}{\alpha} \\ \dots \\ \underbrace{\quad\quad\quad}_n \end{array} \right] = |+\rangle^{\otimes m} \langle +|^{\otimes n} + e^{i\alpha} |-\rangle^{\otimes m} \langle -|^{\otimes n} \quad (31)$$

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. Note that the $(d-1)$ -dimensional vectors used as spider decorations are now just scalars. When such a scalar equals 0, we will omit it.

3.2 Stabilizer Qubit ZX Diagrams

In Section 5.1 we will give a more general definition of the *stabilizer fragment* of the qudit ZX-calculus. For now, we just note that the stabilizer fragment of the qubit ZX-calculus consists of all diagrams in which all phases are integer multiples of $\frac{\pi}{2}$ (see Example 5.8). In [20, Theorem 5.4] 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. This is a ZX-diagram in which:

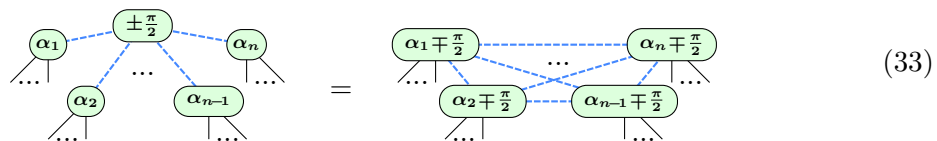
1. Every spider is green.
2. Spiders are only connected by blue (Hadamard) edges.
3. No spider has a self-loop.
4. Every pair of spiders is connected by at most one edge.
5. Every input and output wire of the diagram is connected to a spider.
6. Every spider is connected to at most one of the diagram's input or output wires.

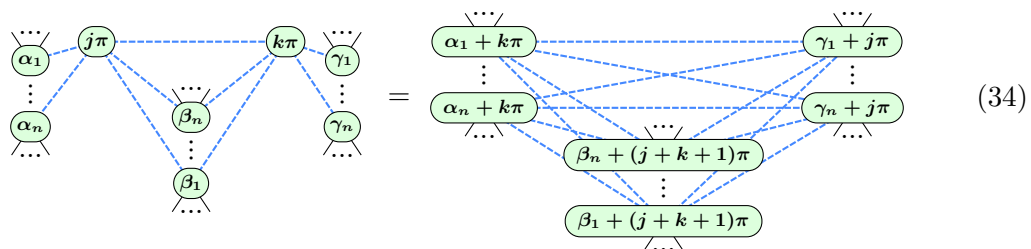
The following derivable rules are key to this:

$$\begin{array}{c} \dots \\ \textcolor{green}{\alpha} \\ \textcolor{blue}{\text{---}} \\ \textcolor{green}{\beta} \\ \dots \end{array} = \begin{array}{c} \dots \\ \textcolor{green}{\alpha} \\ \textcolor{green}{\beta} \\ \dots \end{array}, \quad \begin{array}{c} \dots \\ \textcolor{green}{\alpha} \\ \textcolor{blue}{\text{---}} \end{array} = \begin{array}{c} \dots \\ \textcolor{green}{\alpha} \end{array}, \quad \begin{array}{c} \dots \\ \textcolor{green}{\alpha} \\ \textcolor{blue}{\text{---}} \end{array} = \begin{array}{c} \dots \\ \textcolor{green}{\alpha + \pi} \end{array} \quad (32)$$

An *interior* spider is one that is not connected to any of the diagram's input or output wires. The following two rewrite rules, derived via *local complementation* and *pivoting*, can

be used to eliminate interior spiders:





We call these *elimination theorems*. For more details, see [20, Section 4]. Equation (33) says that we can remove any interior spider with phase $\pm\frac{\pi}{2}$ at the cost of performing a local complementation at said spider. Furthermore, Equation (34) says we can remove any pair of interior spiders with phases in $\{0, \pi\}$ connected by a Hadamard edge at the cost of performing a pivot along said edge. After each application of (33) or (34), we can use (32) to remove any parallel Hadamard edges and ensure the diagram remains graph-like.

A *closed* diagram is one with no input or output wires: under $\llbracket - \rrbracket$, it represents a scalar. In particular, and importantly for our case studies later, the algorithm described above algorithm will *efficiently* simplify any closed stabilizer diagram until it contains at most one spider, at which point the scalar it represents can be easily read off. By ‘efficiently’ we mean via a sequence of spider elimination rewrites whose cost is polynomial in the initial number of spiders. Note each such rewrite updates only a polynomial number of edges in the diagram, which prevents an overwhelming memory cost of the simplification procedure.

4 Simplifying Qutrit ZX-Diagrams

We now turn to the qutrit ZX-calculus (the case $d = 3$) and develop the story analogous to that of the previous section. A local complementation result is known for qutrits; we build on this, defining qutrit graph-like diagrams and showing that every qutrit ZX-diagram can be put in this form. We then derive a qutrit pivot rule and use it to find elimination theorems for all *stabilizer* qutrit spiders. We saw in the last section that the ZX-calculus is especially simple when $d = 2$, so $d = 3$ is the easiest case that behaves more like the general qudit ZX-calculus. Since some of the general theory that will be developed in the next section can get quite complicated, it is instructive to see it ‘in action’ in this easier setting first. Also, we have only managed a partial proof of the qudit local complementation result, so $d = 3$ is the largest d for which our pivot and elimination theorems hold unconditionally.

4.1 The Qutrit ZX-Calculus

First, we note spider decorations are now vectors of dimension $d - 1 = 2$. Rather than write these as vectors (α, β) , we draw spiders as:

$$\begin{array}{c} \dots \\ \text{---} \text{---} \text{---} \\ \text{---} \text{---} \text{---} \\ \dots \end{array} \quad \text{and} \quad \begin{array}{c} \dots \\ \text{---} \text{---} \text{---} \\ \text{---} \text{---} \text{---} \\ \dots \end{array} \quad (35)$$

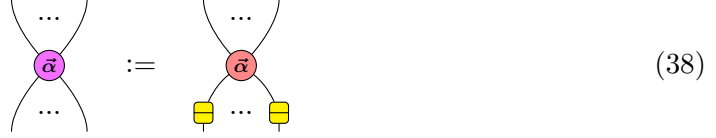
When $\alpha = \beta = 0$ we will again omit the angles entirely, and just draw a small green or red dot. When a spider’s phase is a long expression, we will use a rectangular box instead of a circle. Crucially, the snake equation ceases to take the nice form that it took for the qubit case, so now we must be careful to remember that we have:

$$\begin{array}{c} | \\ \text{---} \end{array} \neq \begin{array}{c} | \\ \text{---} \end{array} , \quad \cup = \begin{array}{c} \cup \\ \text{---} \end{array} \neq \begin{array}{c} \cup \\ \text{---} \end{array} \quad (36)$$

Consequently, the maxim that *Only Connectivity Matters* no longer applies. Diagram components can still be isotoped around the plane but the resulting diagrams only remain equivalent so long as this input/output distinction is respected. This gives the qutrit calculus a slightly more rigid flavour than its qubit counterpart. It is again vital that we remember to demonstrate which of a spider’s legs are outputs and which are inputs:

$$\begin{array}{c} \text{outputs} \\ \text{---} \text{---} \text{---} \\ \text{---} \text{---} \text{---} \\ \text{---} \text{---} \text{---} \\ \text{inputs} \end{array} \quad (37)$$

Remark 4.1. Sadly, losing *Only Connectivity Matters* makes this calculus slightly harder to work with. Hence it's of interest to investigate ways this might be restored. In a recent talk [7], Titouan Carrette shows that one method to do this is to replace the red spider with a modified generator that satisfies what he calls *flexsymmetry*:

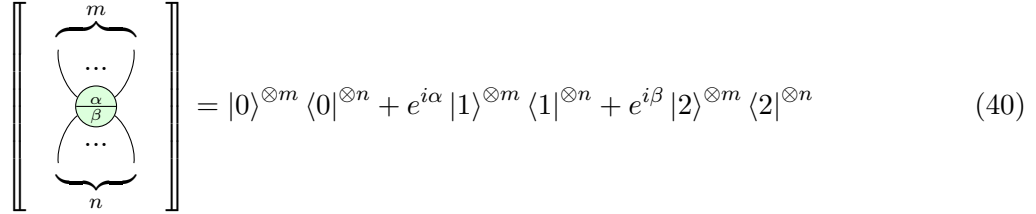


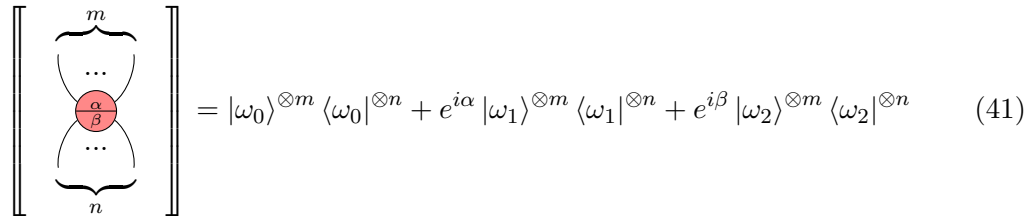
However, this comes at the cost of complicating certain other rules, including the red spider fusion rule.

In concrete linear algebraic terms, let vectors $\{|0\rangle, |1\rangle, |2\rangle\}$ constitute the *Z-basis*, and let $\omega = e^{i\frac{2\pi}{3}}$. The *X-basis* is then:

$$\begin{aligned} |\omega_0\rangle &= \frac{1}{\sqrt{3}} (|0\rangle + |1\rangle + |2\rangle) \\ |\omega_1\rangle &= \frac{1}{\sqrt{3}} (|0\rangle + \omega |1\rangle + \omega^2 |2\rangle) \\ |\omega_2\rangle &= \frac{1}{\sqrt{3}} (|0\rangle + \omega^2 |1\rangle + \omega |2\rangle) \end{aligned} \quad (39)$$

Spiders have the following *standard interpretation* $\llbracket - \rrbracket$ as linear maps:





Warning 4.2. Throughout this section, whenever we use an integer n in a spider decoration, this is a shorthand for $\frac{2\pi}{3}n$. Since spider phases hold mod 2π , these integer decorations hold mod 3. Unless otherwise stated, we will use Greek letters to denote general angles,

and Roman letters for these integer shorthands. We believe this shouldn't cause confusion.

$$\begin{array}{c} \dots \\ \diagup \quad \diagdown \\ \textcircled{\frac{a}{b}} \\ \diagdown \quad \diagup \\ \dots \end{array} := \begin{array}{c} \dots \\ \diagup \quad \diagdown \\ \textcircled{\frac{a \frac{2\pi}{3}}{b \frac{2\pi}{3}}} \\ \diagdown \quad \diagup \\ \dots \end{array} \quad (42)$$

We again use a dashed blue line for the *Hadamard edge* (H -edge), and we additionally introduce a purple dashed line for its adjoint (H^\dagger -edge).

$$\begin{array}{c} | \\ \text{1} \\ | \end{array} = \begin{array}{c} | \\ \text{---} \\ | \end{array} = \begin{array}{c} \textcircled{\frac{2}{2}} \\ | \\ \textcircled{\frac{2}{2}} \\ | \end{array}, \quad \begin{array}{c} | \\ \text{---1} \\ | \end{array} = \begin{array}{c} | \\ \text{---} \\ | \end{array} = \begin{array}{c} \textcircled{\frac{1}{1}} \\ | \\ \textcircled{\frac{1}{1}} \\ | \end{array}, \quad (43)$$

In Ref. [27, Lemma 3.2] it is proved that the usual input/output distinction is irrelevant for H - and H^\dagger -edges:

$$\begin{array}{c} \dots \\ \diagup \quad \diagdown \\ \textcircled{\frac{\alpha}{\beta}} \\ \diagdown \quad \diagup \\ \dots \end{array} \begin{array}{c} \text{---} \\ \diagup \quad \diagdown \\ \textcircled{\frac{\gamma}{\delta}} \\ \diagdown \quad \diagup \\ \dots \end{array} = \begin{array}{c} \dots \\ \diagup \quad \diagdown \\ \textcircled{\frac{\alpha}{\beta}} \\ \diagdown \quad \diagup \\ \dots \end{array} \begin{array}{c} \text{---} \\ \diagup \quad \diagdown \\ \textcircled{\frac{\gamma}{\delta}} \\ \diagdown \quad \diagup \\ \dots \end{array}, \quad \begin{array}{c} \dots \\ \diagup \quad \diagdown \\ \textcircled{\frac{\alpha}{\beta}} \\ \diagdown \quad \diagup \\ \dots \end{array} \begin{array}{c} \text{---} \\ \diagup \quad \diagdown \\ \textcircled{\frac{\gamma}{\delta}} \\ \diagdown \quad \diagup \\ \dots \end{array} = \begin{array}{c} \dots \\ \diagup \quad \diagdown \\ \textcircled{\frac{\alpha}{\beta}} \\ \diagdown \quad \diagup \\ \dots \end{array} \begin{array}{c} \text{---} \\ \diagup \quad \diagdown \\ \textcircled{\frac{\gamma}{\delta}} \\ \diagdown \quad \diagup \\ \dots \end{array} \quad (44)$$

The full set of rules governing the qutrit ZX-calculus is shown in Figure 4.

4.2 Graph-Like Qutrit ZX Diagrams

A *graph-like* qutrit ZX-diagram satisfies exactly the same conditions as for the qubit case, only we also allow purple H^\dagger -edges to connect spiders, as well as blue H -edges. A graph-like qutrit ZX-diagram is a *graph state* when every spider has zero phase (top and bottom) and is connected to one of the diagram's outputs.

Again, we can show that every qutrit ZX-diagram is equivalent to a graph-like one. The following equations, proved more generally for qudits in (76), are vital to this:

$$\begin{array}{c} \text{---} \\ \diagup \quad \diagdown \\ \text{1} \quad \text{1} \quad \text{1} \\ \diagdown \quad \diagup \\ \text{---} \end{array} = \begin{array}{c} \text{---} \\ \diagup \quad \diagdown \\ \text{---1} \quad \text{---1} \quad \text{---1} \\ \diagdown \quad \diagup \\ \text{---} \end{array}, \quad \begin{array}{c} \text{---} \\ \diagup \quad \diagdown \\ \text{1} \quad \text{1} \\ \diagdown \quad \diagup \\ \text{---} \end{array} = \begin{array}{c} \text{---} \\ \text{---1} \\ \text{---} \end{array}, \quad \begin{array}{c} \text{---} \\ \diagup \quad \diagdown \\ \text{---1} \quad \text{---1} \\ \diagdown \quad \diagup \\ \text{---} \end{array} = \begin{array}{c} \text{---} \\ \text{1} \\ \text{---} \end{array} \quad (45)$$

So we can think of Hadamard box decorations as being mod 3 (i.e. as elements of \mathbb{Z}_3), and satisfying a kind of 'fusion rule' across parallel edges. That is, for any integer n :

$$\begin{array}{c} | \\ \text{n} \\ | \end{array} := \left. \begin{array}{c} \text{---} \\ \diagup \quad \diagdown \\ \text{1} \quad \dots \quad \text{1} \\ \diagdown \quad \diagup \\ \text{---} \end{array} \right\} n \text{ times} \quad (46)$$

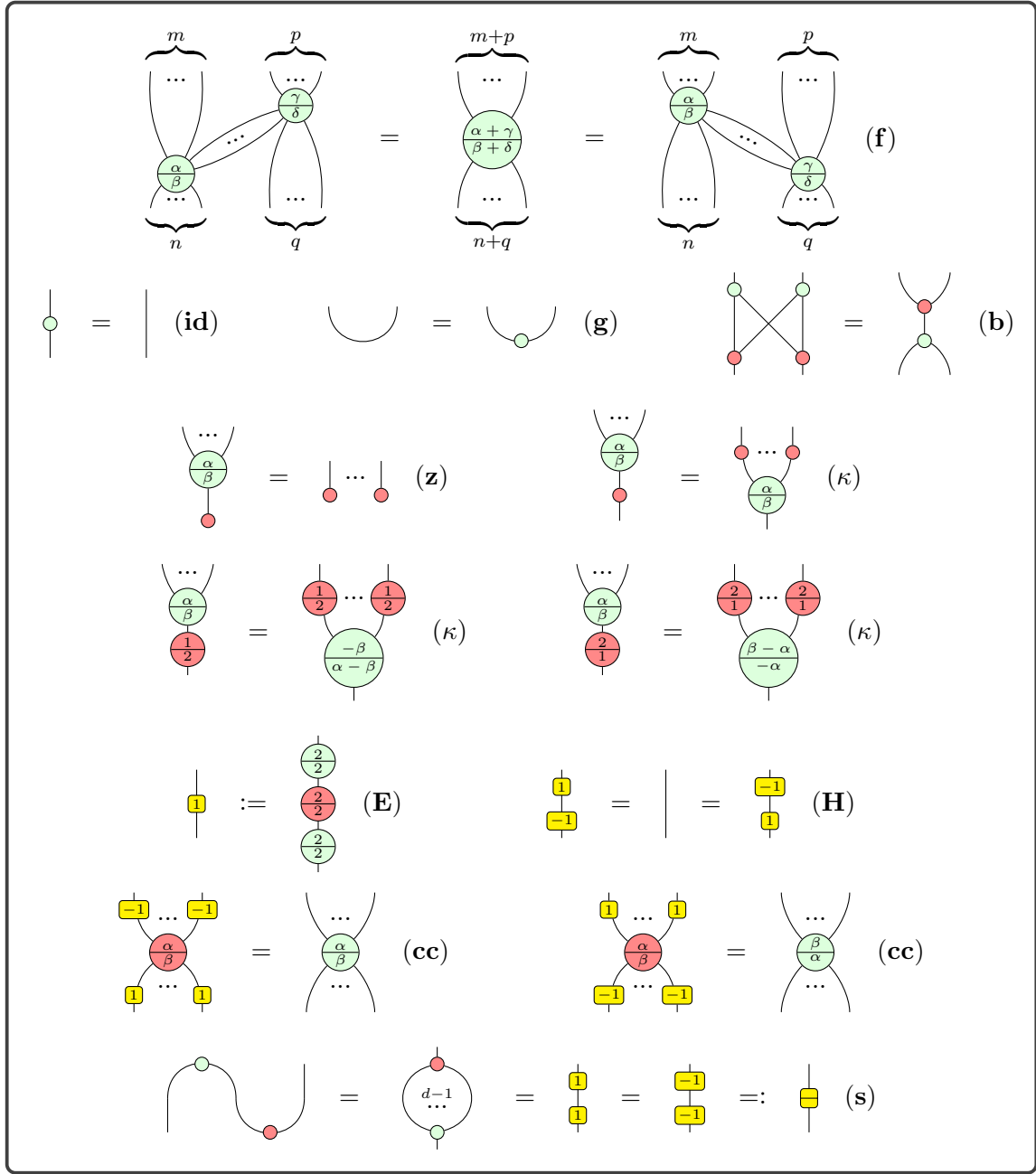


Figure 4: Qutrit ZX-calculus rules. The fusion rule (f) only holds if the two spiders are connected by at least one wire. α, β, γ and δ are all taken mod 2π . Integers appearing in spider phases are shorthands for integer multiples of $\frac{2\pi}{3}$.

Where the previous equations relate single H - and H^\dagger -boxes across multiple edges, the next three relate multiple H - and H^\dagger - boxes on single edges. They hold for $h \in \{1, -1\}$, and are proved via simple applications of rules **(id)**, **(H)** and **(s)**.

$$\begin{array}{c} \text{---} \\ | \\ \text{---} \end{array} \begin{array}{c} \boxed{h} \\ \boxed{h} \\ \boxed{h} \\ \boxed{h} \end{array} = \begin{array}{c} | \\ | \\ | \\ | \end{array}, \quad \begin{array}{c} \text{---} \\ | \\ \text{---} \end{array} \begin{array}{c} \boxed{h} \\ \boxed{h} \\ \boxed{h} \\ \boxed{h} \end{array} = \begin{array}{c} \text{---} \\ | \\ \boxed{-h} \\ | \\ \text{---} \end{array}, \quad \begin{array}{c} \text{---} \\ | \\ \text{---} \end{array} \begin{array}{c} \boxed{h} \\ \boxed{h} \end{array} = \begin{array}{c} \text{---} \\ | \\ \bigcirc \\ | \\ \text{---} \end{array} \begin{array}{c} \boxed{h} \\ \boxed{h} \end{array} \quad (47)$$

The final equations we'll need concern self-loops. In the qutrit ZX-calculus, due to spiders' input/output leg distinction, there are several ways one could define a self-loop; with two input legs, two output legs, or one input leg and one output leg. For a Hadamard self-loop there are then further choices about where to put the yellow box. In fact, all of these choices lead to the same result. We defer the proof to the general qudit case – see Lemma 5.10:

$$\begin{array}{c} \dots \\ \text{---} \end{array} \begin{array}{c} \bigcirc \\ \alpha \\ \beta \\ \bigcirc \end{array} \begin{array}{c} \text{---} \\ \dots \end{array} = \begin{array}{c} \dots \\ \text{---} \end{array} \begin{array}{c} \bigcirc \\ \alpha \\ \beta \\ \bigcirc \end{array} \begin{array}{c} \text{---} \\ \dots \end{array}, \quad \begin{array}{c} \dots \\ \text{---} \end{array} \begin{array}{c} \bigcirc \\ \alpha \\ \beta \\ \bigcirc \end{array} \begin{array}{c} \text{---} \\ \dots \end{array} = \begin{array}{c} \dots \\ \text{---} \end{array} \begin{array}{c} \bigcirc \\ \alpha+1 \\ \beta+1 \\ \bigcirc \end{array} \begin{array}{c} \text{---} \\ \dots \end{array}, \quad \begin{array}{c} \dots \\ \text{---} \end{array} \begin{array}{c} \bigcirc \\ \alpha \\ \beta \\ \bigcirc \end{array} \begin{array}{c} \text{---} \\ \dots \end{array} = \begin{array}{c} \dots \\ \text{---} \end{array} \begin{array}{c} \bigcirc \\ \alpha-1 \\ \beta-1 \\ \bigcirc \end{array} \begin{array}{c} \text{---} \\ \dots \end{array} \quad (48)$$

Proposition 4.3. 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 (47) 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, so that no plain edge connects two distinct spiders. Remove self-loops via (48), then apply (45) to all connected pairs of spiders until at most one H - or H^\dagger -edge remains between them. Finally, to ensure each of the diagram's inputs and outputs is connected to a spider and every spider is connected to at most one of the diagram's inputs or outputs, we can use **(H)** and **(id)** to add a few spiders, H - and H^\dagger -edges as needed:

$$\begin{array}{c} | \\ | \\ | \end{array} = \begin{array}{c} \bigcirc \\ | \\ \bigcirc \\ | \\ \bigcirc \end{array}, \quad \begin{array}{c} \text{---} \\ | \\ \text{---} \end{array} \boxed{h} = \begin{array}{c} \text{---} \\ | \\ \boxed{h} \\ | \\ \text{---} \end{array} \begin{array}{c} \bigcirc \\ | \\ \bigcirc \end{array}, \quad \begin{array}{c} \dots \\ \text{---} \end{array} \begin{array}{c} \bigcirc \\ \alpha \\ \beta \\ \bigcirc \end{array} \begin{array}{c} \text{---} \\ \dots \end{array} = \begin{array}{c} \dots \\ \text{---} \end{array} \begin{array}{c} \bigcirc \\ \alpha \\ \beta \\ \bigcirc \end{array} \begin{array}{c} \text{---} \\ \dots \end{array} \begin{array}{c} \bigcirc \\ | \\ \bigcirc \end{array} \begin{array}{c} \text{---} \\ \dots \end{array} \quad (49)$$

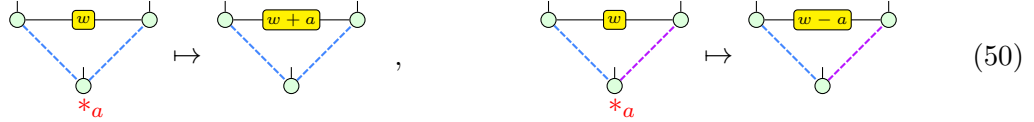
□

A graph state is described fully by its underlying multigraph, or equivalently by an adjacency matrix, where edges take weights in \mathbb{Z}_3 [51, Lemma 4.2]. Nodes correspond to phaseless green spiders, edges of weight 1 correspond to Hadamard edges, and edges of weight 2 correspond to H^\dagger -edges. Each spider is additionally given an output leg which is

an output for the overall diagram. As in the qubit case, graph states admit a *local complementation* operation [51, Definition 2.6], though the effect is now slightly more complicated. We'll give the intuition after the formal definition:

Definition 4.4. Given $a \in \mathbb{Z}_3$ and a graph state G with adjacency matrix $W = (w_{i,j})$, the a -local complementation at node x is the new graph state $G *_a x$, whose adjacency matrix $W' = (w'_{i,j})$ is given by $w'_{i,j} = w_{i,j} + aw_{i,x}w_{j,x}$.

So only those edges between neighbours of node x are affected. Specifically, for two nodes i and j both connected to x by the *same* colour edge, a -local complementation at x *increases* weight $w_{i,j}$ by a . If instead i and j are connected to x by edges of *different* colours, a -local complementation at x *decreases* $w_{i,j}$ by a . This is shown graphically below, and holds with the roles of blue and purple interchanged:



Theorem 4.5. [51, Theorem 4.4, Corollary 4.5] Given $a \in \mathbb{Z}_3$ and a graph state (G, W) containing a node x , let $N(x)$ denote the neighbours of x ; that is, nodes i with weight $w_{i,x} \in \{1, 2\}$. Then the following equality holds:

Definition 4.6. Given $a, b, c \in \mathbb{Z}_3$ and a graph state G containing nodes i and j , the (a, b, c) -pivot along ij is the new graph state $G \wedge_{(a,b,c)} ij := ((G *_a i) *_b j) *_c i$.

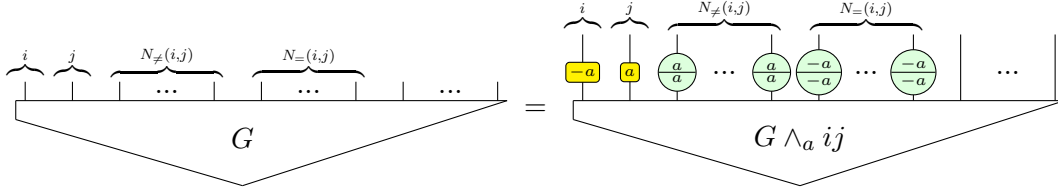
This pivot operation again leads to an equality, up to introducing some extra gates on outputs, whose proof is found in Appendix C.1. Here we shall only consider an $(a, -a, a)$ -pivot along an edge ij of non-zero weight, for $a \in \{1, 2\}$. We will call this a *proper a-pivot* along ij , and denote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

Theorem 4.7. Given $a \in \mathbb{Z}_3$ and a graph state (G, W) containing connected nodes i and j , define:

$$N_{=}(i, j) := \{x \in N(i) \cap N(j) \mid w_{x,i} = w_{x,j}\} \quad (52)$$

$$N_{\neq}(i, j) := \{x \in N(i) \cap N(j) \mid w_{x,i} \neq w_{x,j}\} \quad (53)$$

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



4.3 Qutrit Elimination Theorems

Again, we defer a proper definition of the stabilizer fragment to Section 5.1, but we note that a qutrit stabilizer ZX-diagram is exactly one in which every component of every spider's phase is an integer multiple of $\frac{2\pi}{3}$ (see Equation (106)). We then classify green stabilizer spiders into three families exactly as in Ref. [51, Theorem 3.1]:

$$\mathcal{M} = \left\{ \begin{pmatrix} \cdots \\ 0 \\ \cdots \end{pmatrix}, \begin{pmatrix} \cdots \\ 1 \\ \cdots \end{pmatrix}, \begin{pmatrix} \cdots \\ 2 \\ \cdots \end{pmatrix} \right\} \quad (54)$$

$$\mathcal{N} = \left\{ \begin{pmatrix} \cdots \\ 0 \\ 1 \\ \cdots \end{pmatrix}, \begin{pmatrix} \cdots \\ 1 \\ 0 \\ \cdots \end{pmatrix}, \begin{pmatrix} \cdots \\ 0 \\ 2 \\ \cdots \end{pmatrix}, \begin{pmatrix} \cdots \\ 2 \\ 0 \\ \cdots \end{pmatrix} \right\} \quad (55)$$

$$\mathcal{P} = \left\{ \begin{pmatrix} \cdots \\ 1 \\ 1 \\ \cdots \end{pmatrix}, \begin{pmatrix} \cdots \\ 2 \\ 2 \\ \cdots \end{pmatrix} \right\} \quad (56)$$

We again call a spider in a graph-like ZX-diagram *interior* if it isn't connected to an input or output of the diagram. Given any graph-like ZX-diagram, we will show that we can eliminate standalone interior \mathcal{P} - and \mathcal{N} -spiders by local complementation, and pairs of connected interior \mathcal{M} -spiders by pivoting.

First, we define (a very small extension of) the *!-box* notation (pronounced 'bang-box'), as introduced in [18] for general string diagrams. A *!-box* in a ZX-diagram is a compressed notation for a family of diagrams; the contents of the *!-box*, along with any wires into or out of it, are 'unfolded' (copied) $n \geq 0$ times and placed side-by-side. Following the style of [4], we also allow a parameter over which a *!-box* unfolds. We extend this notation as follows: in the top-left corner of the *!-box* we introduce a further diagram, which denotes that the spiders unfolded by the *!-box* have a complete graph imposed on them, whose edges take the form of this 'corner diagram'. In qutrit ZX-diagrams this notation will only be well-defined in certain scenarios: in particular, it is well-defined when the corner diagram is a H - or H^\dagger -edge (since then by (44) it doesn't matter whether this is an input or output of the spider) and the main contents of the *!-box* is equivalent to a single spider (since then there is no ambiguity about which spiders are connected by the corner diagram). For

example, letting $[K] = \{1, \dots, K\}$, we have:

$$\left\{ \begin{array}{c} k \in [K] \\ \downarrow \\ \text{Diagram of } \alpha_k, \beta_k \end{array} : K \in \{0, 1, 2, 3, \dots\} \right\} = \left\{ \begin{array}{c} \text{Diagram 0} \\ \text{Diagram 1} \\ \text{Diagram 2} \\ \text{Diagram 3} \\ \vdots \end{array} \right\} \quad (57)$$

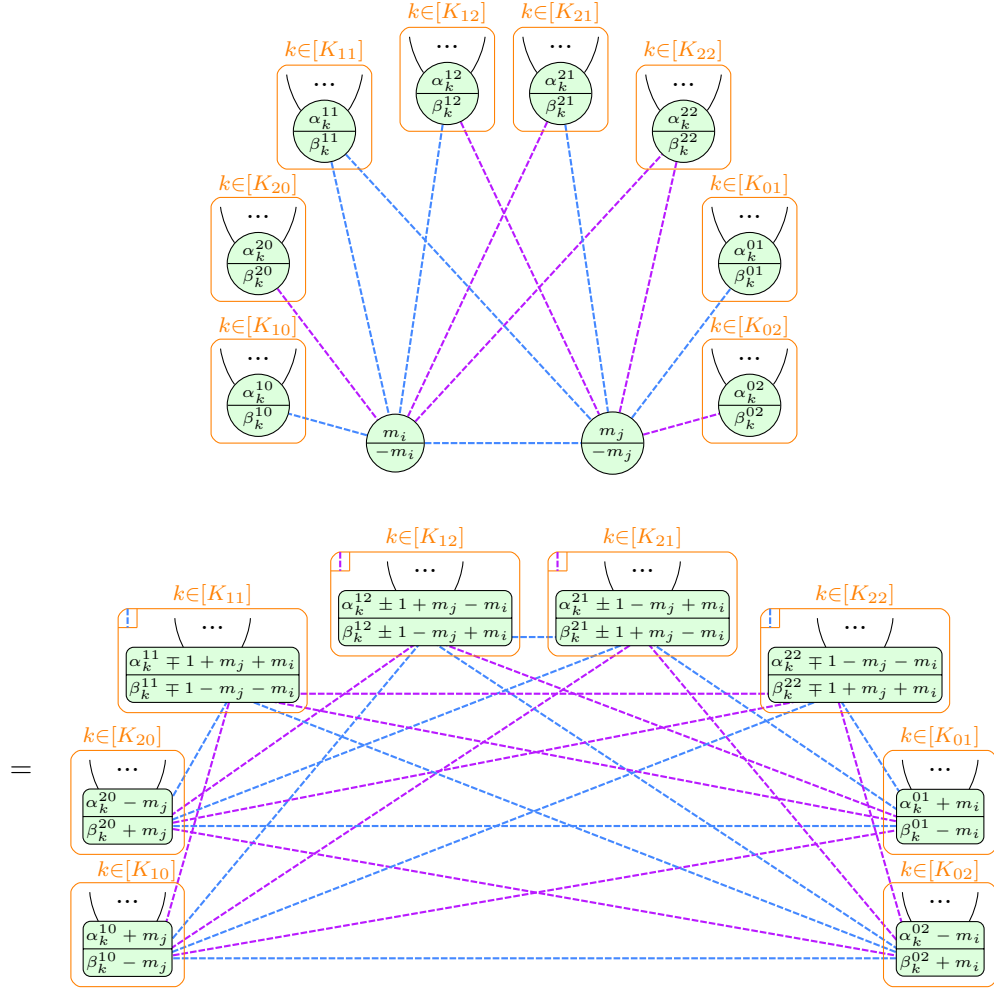
Theorem 4.8. Given any graph-like ZX-diagram containing an interior \mathcal{P} -spider x with phase $\frac{p}{p}$ for $p \in \{1, 2\}$, suppose we perform a p -local complementation at x . Then the new ZX-diagram is related to the old one by the following equality:

The figure illustrates a graphical equation between two network structures. On the left, a central node p (green circle) is connected to two nodes, α_k and γ_k (green circles), which are in turn connected to two sets of nodes, K_1 and K_2 (orange boxes). The nodes α_k and γ_k are labeled with α_k and γ_k respectively, and the nodes in K_1 and K_2 are labeled with β_k and δ_k respectively. On the right, the same structure is shown, but the connection between the two sets of nodes, K_1 and K_2 , is different, representing a different network configuration. The equation is represented by an equals sign between the two structures.

Theorem 4.9. Given any graph-like ZX-diagram containing an interior \mathcal{N} -spider x with phase $\frac{0}{n}$ or $\frac{n}{0}$ for $n \in \{1, 2\}$, suppose we perform a $(-n)$ -local complementation at x . Then, treating the two cases separately, the new ZX-diagrams are related to the old ones by the equalities:

Theorem 4.10. 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 $\pm w$ -pivot along ij (both choices give the same result). For the case $w = 1$, the new ZX-diagram is

related to the old one by the following equality:



The case $w = 2$ differs only as follows: in the first diagram (the left hand side of the equation), the edge ij is purple (by definition), while in the lower diagram (the right hand side of the equation), a $\pm 2 = \mp 1$ will replace all occurrences of ± 1 , and the roles of purple and blue will be swapped throughout.

Proofs of the first two of these theorems are given for general qudit systems in Appendix B.1, while the latter is proved in Appendix D.1. We can now combine them 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, and perhaps removing parallel H - or H^\dagger -edges via (45), we again have a graph-like diagram.

Theorem 4.11. 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 (45) as needed until the resulting diagram is graph-like:

1. Eliminate a \mathcal{P} -spider via Theorem 4.8.
2. Eliminate an \mathcal{N} -spider via Theorem 4.9.
3. Eliminate two adjacent \mathcal{M} -spiders via Theorem 4.10.

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

In particular, if we start with a stabilizer diagram, we can eliminate all but perhaps one spider, depending on whether the initial number of \mathcal{M} -spiders was odd or even. This is because no step introduces any non-stabilizer phases. The algorithm above could be extended to a *non-closed* graph-like diagrams as in Ref. [20, Theorem 5.4]) – for example, as part of a qutrit circuit optimisation algorithm. This is not required, however, for the applications we discuss in Section 6, so we have not done so.

5 Qudit ZX-Calculus: Part Two

We close by investigating stabilizer diagrams and elimination theorems in the general qudit ZX-calculus. Here we finally give a full, general definition of the stabilizer fragment, and find we can completely classify it in terms of spider phase decorations. This is perhaps the deepest and most structural result this work has to offer. We define graph-like qudit ZX-diagrams and show every qudit ZX-digram can be put in this form. We then go a long way to proving a local complementation rule, from which we derive a pivot rule. Finally, we find elimination theorems for two classes of stabilizer qudit spiders. In the last section, the occasional lemma was already known; here all work is entirely novel.

5.1 The Stabilizer Fragment

The *Pauli gates*, symbolically denoted Z and X , are the following two ZX-diagrams:

$$\begin{array}{c} | \\ \textcircled{\vec{\kappa}} \\ | \end{array} , \quad \begin{array}{c} | \\ \textcircled{\vec{\kappa}} \\ | \end{array} \quad (58)$$

Recall we let $\omega = e^{i\frac{2\pi}{d}}$. The *Pauli group* \mathcal{P}_n on n qudits is exactly the set of elements of the form $\omega^k Z^{a_1} X^{b_1} \otimes \dots \otimes Z^{a_n} X^{b_n}$, for $k, a_i, b_i \in \mathbb{Z}_d$. Diagrammatically (recall we omit global phases such as ω^k in the ZX-calculus), such an element has the form:

$$\begin{array}{c} | \quad | \quad | \quad \dots \quad | \\ \textcircled{a_1 \vec{\kappa}} \quad \textcircled{a_2 \vec{\kappa}} \quad \dots \quad \textcircled{a_n \vec{\kappa}} \\ \textcircled{b_1 \vec{\kappa}} \quad \textcircled{b_2 \vec{\kappa}} \quad \dots \quad \textcircled{b_n \vec{\kappa}} \\ | \quad | \quad | \quad \dots \quad | \end{array} \quad (59)$$

The *Clifford group* \mathcal{C}_n is the unitary normalizer of \mathcal{P}_n :

$$\mathcal{C}_n = \left\{ Q \mid Q \mathcal{P}_n Q^\dagger = \mathcal{P}_n, \quad Q^\dagger = Q^{-1} \right\} \quad (60)$$

For any d , this group is generated by three ZX-diagrams [30]. The first is the now-familiar Hadamard gate, while the second and third are new to us: they are the *shift gate* and the *generalised CNOT*, respectively:

$$\begin{array}{c} | \\ \textcircled{1} \\ | \end{array} , \quad \begin{array}{c} | \\ \textcircled{\vec{s}} \\ | \end{array} , \quad \begin{array}{c} | \quad | \\ \textcircled{\phantom{\vec{s}}} \quad \textcircled{\phantom{\vec{s}}} \\ | \quad | \end{array} \quad (61)$$

where the phase \vec{s} has components \vec{s}_j given by:

$$\vec{s}_j = \frac{j}{2}(j+2-d) \frac{2\pi}{d} \quad (62)$$

The *shift gate* S has not been defined as a ZX-diagram before. We have derived its definition from its action under unitary conjugation on the Pauli gates X and Z [23]. Specifically, it satisfies:

$$\begin{array}{c} \textcircled{\vec{s}} \\ \textcircled{\vec{\kappa}} \\ \textcircled{-\vec{s}} \end{array} = \begin{array}{c} \textcircled{\vec{\kappa}} \\ \textcircled{-\vec{\kappa}} \end{array}, \quad \begin{array}{c} \textcircled{\vec{s}} \\ \textcircled{\vec{\kappa}} \\ \textcircled{-\vec{s}} \end{array} = \begin{array}{c} \textcircled{\vec{\kappa}} \end{array} \quad (63)$$

Note that for even d , components of the phase \vec{s} can be *half*-integer multiples of $\frac{2\pi}{d}$, e.g. $\vec{s}_{d-1} = \frac{d-1}{2} \frac{2\pi}{d}$. We will see in Theorem 5.2 that this is a consequence of a key difference between the stabilizer fragments of odd and even dimensioned ZX-calculi. Indeed, we now finally define the *stabilizer fragment* of the qudit ZX-calculus: it is the set of all diagrams generated by the Clifford group \mathcal{C}_n and the *computational basis states* and *effects*:

$$\begin{array}{c} \textcircled{\vec{\kappa}} \end{array}, \quad \begin{array}{c} \textcircled{2\vec{\kappa}} \end{array}, \quad \dots, \quad \begin{array}{c} \textcircled{(d-1)\vec{\kappa}} \end{array}, \quad \begin{array}{c} \textcircled{\vec{\kappa}} \\ | \end{array}, \quad \begin{array}{c} \textcircled{2\vec{\kappa}} \\ | \end{array}, \quad \dots, \quad \begin{array}{c} \textcircled{(d-1)\vec{\kappa}} \\ | \end{array} \quad (64)$$

Remark 5.1. Initially it might seem odd that the computational basis states are red; in the ZX-calculus, Z generally means green, and we've used the term Z -basis and computational basis interchangeably. Plus, we defined $\llbracket - \rrbracket$ on the green Z -spider in terms of this computational basis. But if one crunches the numbers one sees that, up to a scalar:

$$\llbracket \begin{array}{c} | \\ \textcircled{k\vec{\kappa}} \end{array} \rrbracket = |k\rangle \quad , \quad \llbracket \begin{array}{c} \textcircled{k\vec{\kappa}} \\ | \end{array} \rrbracket = \langle k| \quad (65)$$

Let k be an integer and x be real. It turns out that phases $\vec{p}^{k,x}$, whose components $\vec{p}_j^{k,x}$ have the form defined below, will be of great use to us.

$$\vec{p}_j^{k,x} = \frac{-j}{2} ((j+1)k + x) \frac{2\pi}{d} \quad (66)$$

Their definition comes from the fact that these phases satisfy the following equation, proved via a single use of the (κ) rule:

$$\begin{array}{c} \textcircled{\vec{p}^{k,x}} \\ \textcircled{n\vec{\kappa}} \end{array} = \begin{array}{c} \textcircled{n\vec{\kappa}} \\ \textcircled{\vec{p}^{k,x}} \\ \textcircled{nk\vec{\kappa}} \end{array} \quad (67)$$

An immediate nice property is that the last component $\vec{p}_{d-1}^{k,x}$ is always exactly $\frac{x}{2}$. More importantly, letting \mathbb{Z}_d denote the integers mod d and \mathbb{R}_{2d} denote the reals mod $2d$, these phases can – under addition – be viewed as the Abelian group $\mathbb{Z}_d \times \mathbb{R}_{2d}$. That is:

- $\vec{p}^{k,x} + \vec{p}^{l,y} = \vec{p}^{k+l,x+y}$

- $-\vec{p}^{k,x} = \vec{p}^{-k,-x}$
- $\vec{p}^{0,0}$ is the identity

From now on, we will denote $\vec{p}^{k,x}$ as $\langle k, x \rangle$. This group includes some important phases:

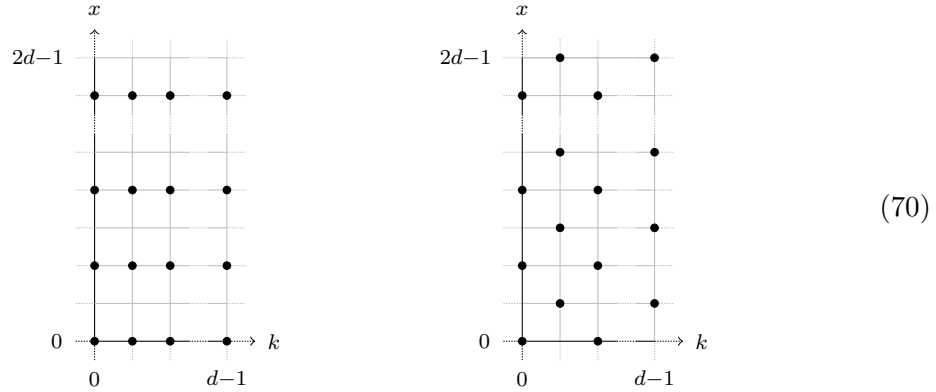
$$\vec{\kappa} = \langle 0, -2 \rangle, \quad \vec{\tau} = \langle -1, 1 - d \rangle, \quad \vec{s} = \langle -1, d - 1 \rangle \quad (68)$$

We then have the following theorem which completely characterises the stabilizer fragment of the qudit ZX-calculus in terms of spider phases:

Theorem 5.2. A ZX-diagram is stabilizer if and only if every component of every spider's phase is in a particular Abelian subgroup of $\mathbb{Z}_d \times \mathbb{R}_{2d}$. This subgroup \mathcal{S}_d takes a different form for odd and even d :

$$\mathcal{S}_d = \begin{cases} \{\langle k, 2x \rangle \mid k, x \in \mathbb{Z}_d\} & \text{if } d \text{ odd} \\ \{\langle k, x \rangle \mid k \in \mathbb{Z}_d, x \in \mathbb{Z}_{2d}, k \equiv x \pmod{2}\} & \text{if } d \text{ even} \end{cases} \quad (69)$$

The two cases of this subgroup are drawn respectively below – a dot at a vertex indicates that this point $\langle k, x \rangle$ is an element of \mathcal{S}_d :



Note that \mathcal{S}_d always has size d^2 , regardless of the parity of d . The proof of this theorem requires a few lemmas. First, via simple applications of rules **(f)** and **(cc)**, any ZX-diagram can be decomposed into the following components:

(71)

The first four of these are shown in [50] to be stabilizer diagrams. We've stated already that the Hadamard gate is stabilizer, and since its adjoint is just the composition of three Hadamards, this must be stabilizer too.

Now, crucially, the only new spiders this decomposition introduces must have trivial phases. Also, the only spiders with non-trivial phases are those with exactly one input and one output – from now on we’ll call such spiders *gates*. Since we’re interested in determining which phases can appear in a stabilizer ZX-diagram (those generated by the Clifford group and computational basis states and effects), and since the first six diagrams above are known to be stabilizer diagrams, we can restrict our attention to the 1-qudit Clifford group \mathcal{C}_1 in which gates live. We call this the *local Clifford group*. A first lemma we’ll need is:

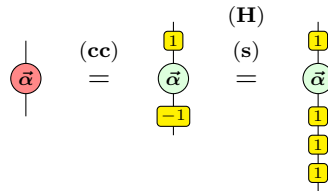
Lemma 5.3. Z is in the local Clifford group \mathcal{C}_1 .

Proof. Since \mathcal{C}_1 is defined to be the normalizer of the 1-qudit Pauli group \mathcal{P}_1 , and since Z is in \mathcal{P}_1 , we must have $Z \in \mathcal{C}_1$ too. \square

Since the Hadamard gate is in the local Clifford group \mathcal{C}_1 , we have:

Lemma 5.4. The red version of any green gate in \mathcal{C}_1 is also in \mathcal{C}_1 , and vice versa.

Proof. We prove one half – the ‘vice versa’ part is near-identical.



\square

Thus we can just focus on one colour of spider for the moment: let’s choose green. Note that when two green spiders fuse their phases are added together. So in fact we can temporarily forget about spiders altogether and just look at phases under addition, knowing that this translates back to diagrammatic ideas easily enough. To this end, we define the *local Clifford phase group* $\vec{\mathcal{C}}_1$ by the property that $\vec{\alpha} \in \vec{\mathcal{C}}_1$ if and only if any gate with phase $\vec{\alpha}$ is in \mathcal{C}_1 . The following lemma means we can subtract as well as add phases in $\vec{\mathcal{C}}_1$ (partially justifying our use of the word *group*):

Lemma 5.5. If $\langle k, x \rangle \in \mathbb{Z}_d \times \mathbb{Z}_{2d}$ is in $\vec{\mathcal{C}}_1$ then so is its inverse $\langle -k, -x \rangle$.

Proof. $2d\langle k, x \rangle = \langle 2dk, 2dx \rangle \equiv \langle 0, 0 \rangle$. So $(2d - 1)\langle k, x \rangle \equiv \langle -k, -x \rangle$. \square

Let’s look more closely at what happens when d odd:

Lemma 5.6. If d odd, $\mathcal{S}_d = \{ \langle k, 2x \rangle \mid k, x \in \mathbb{Z}_d \}$ is a subgroup of $\vec{\mathcal{C}}_1$.

Proof. Z has phase $\vec{\kappa} = \langle 0, -2 \rangle$ and is in \mathcal{C}_1 , hence $\langle 0, -2 \rangle$ is in $\vec{\mathcal{C}}_1$. Thus so is $\langle 0, 2 \rangle$ by Lemma 5.5. We know $\vec{s} = \langle -1, d-1 \rangle$ is in $\vec{\mathcal{C}}_1$. Therefore so is:

$$\langle -1, d-1 \rangle - \frac{d-1}{2} \langle 0, 2 \rangle = \langle -1, 0 \rangle$$

(Multiplication by $\frac{d-1}{2}$ is valid, since d odd). Hence its inverse $\langle 1, 0 \rangle$ is in $\vec{\mathcal{C}}_1$ too. Now note that the phases $\langle 1, 0 \rangle$ and $\langle 0, 2 \rangle$ generate \mathcal{S}_d . \square

When d is even, this same idea gives the desired analogous result.

Lemma 5.7. If d even, $\mathcal{S}_d = \{ \langle k, x \rangle \mid k \in \mathbb{Z}_d, x \in \mathbb{Z}_{2d}, k \equiv x \pmod{2} \}$ is a subgroup of $\vec{\mathcal{C}}_1$.

Proof. Again, we note $\langle 0, 2 \rangle$ and $\langle -1, d-1 \rangle$ are in $\vec{\mathcal{C}}_1$. Thus so is:

$$\langle -1, d-1 \rangle - \frac{d-2}{2} \langle 0, 2 \rangle = \langle -1, 1 \rangle$$

Hence so is its inverse $\langle 1, -1 \rangle$ and the sum $\langle 1, 1 \rangle = \langle 0, 2 \rangle + \langle 1, -1 \rangle$. Now note that $\langle 1, -1 \rangle$ and $\langle 1, 1 \rangle$ generate \mathcal{S}_d . \square

The proof of the main theorem then follows:

Proof of Theorem 5.2. In one direction, suppose we have a ZX-diagram with phases all in \mathcal{S}_d . We can decompose the diagram into components as in (71), the first six of which are known to be stabilizer. What remains to check are gates with phases in \mathcal{S}_d , each of which we can decompose into local Clifford group elements via either Lemma 5.6 (if d odd) or Lemma 5.7 (if d even). In the other direction, suppose we have a stabilizer ZX-diagram. This must be generated by the three Clifford group generators from (61), plus computational basis states and effects. The Hadamard gate further decomposes into three spider gates, each with phase $\vec{\tau}$. So then simply note that, regardless of the parity of d , the phases of all such components are in \mathcal{S}_d :

$$k\vec{\kappa} = \langle 0, -2k \rangle, \quad \vec{\tau} = \langle -1, 1-d \rangle, \quad \vec{s} = \langle -1, d-1 \rangle \quad (72)$$

\square

Example 5.8. Earlier we said that a qubit ZX-diagram is stabilizer if and only if all of its components are integer multiples of $\frac{\pi}{2}$. We can now verify this using Theorem 5.2. The stabilizer phases are $\langle 0, 0 \rangle, \langle 1, 1 \rangle, \langle 0, 2 \rangle, \langle 1, 3 \rangle$:



Recall $\langle k, x \rangle = \vec{p}^{k,x}$. Since $d = 2$, these phases are just scalars. Noting that $\frac{2\pi}{d} = \pi$ here, these scalar phases are given by:

$$\langle k, x \rangle_1 = \vec{p}_1^{k,x} = -\left(\frac{1+1}{2}k + \frac{x}{2}\right)\pi = -(k + \frac{x}{2})\pi \quad (74)$$

Thus the stabilizer phases are:

$$\begin{aligned} \langle 0, 0 \rangle_1 &= 0 \\ \langle 1, 1 \rangle_1 &= -\frac{3}{2}\pi = \frac{\pi}{2} \\ \langle 0, 2 \rangle_1 &= -\pi = \pi \\ \langle 1, 3 \rangle_1 &= -\frac{5}{2}\pi = \frac{3\pi}{2} \end{aligned}$$

Similarly, we said in Section 4.3 that a qutrit ZX-diagram is stabilizer if and only if all of its components are integer multiples of $\frac{2\pi}{3}$. This too can now be verified via Theorem 5.2 (see Equation (106)).

5.2 Local Complementation and Pivot

Now that we know what spider phases we might encounter in the stabilizer ZX-calculus, we can begin to consider corresponding elimination theorems. The first step, as ever, is to find an equality involving a graph state and its local complementation. We generalise the following shorthand from the qutrit case, for any $n \in \mathbb{Z}$:

$$\begin{array}{c} | \\ \text{---} \text{[n]} \text{---} \\ | \end{array} := \left\{ \begin{array}{c} | \\ \text{---} \text{[1]} \cdots \text{[1]} \text{---} \\ | \end{array} \right\} n \text{ times} \quad (75)$$

We call this an *n-Hadamard edge* and remark that this is well-defined mod d : in particular, for $n \in \{d-1, d\}$ we have:

$$\begin{array}{c} | \\ \text{---} \text{[d-1]} \text{---} \\ | \end{array} = \begin{array}{c} | \\ \text{---} \text{[-1]} \text{---} \\ | \end{array}, \quad \begin{array}{c} | \\ \text{---} \text{[d]} \text{---} \\ | \end{array} = \begin{array}{c} | \\ \text{---} \text{---} \\ | \end{array} \quad (76)$$

Proof. For the first equation:

$$\begin{array}{c} | \\ \text{---} \text{[d-1]} \text{---} \\ | \end{array} = \begin{array}{c} | \\ \text{---} \text{[1]} \cdots \text{[d-1]} \text{---} \\ | \end{array} \stackrel{(\text{cc})}{=} \begin{array}{c} | \\ \text{---} \text{---} \\ | \end{array} \stackrel{(\text{s})}{=} \begin{array}{c} | \\ \text{---} \text{[d]} \text{---} \\ | \end{array} \stackrel{(\text{H})}{=} \begin{array}{c} | \\ \text{---} \text{[-1]} \text{---} \\ | \end{array}$$

And the second:

□

For such edges between two spiders of the same colour, the usual input/output distinction is again irrelevant. The proof of this when $n = 1$ is exactly the same as for the qutrit case (44), then the result for general n is a trivial corollary.

Definition 5.9. A qudit *graph-like diagram* is a ZX-diagram in which:

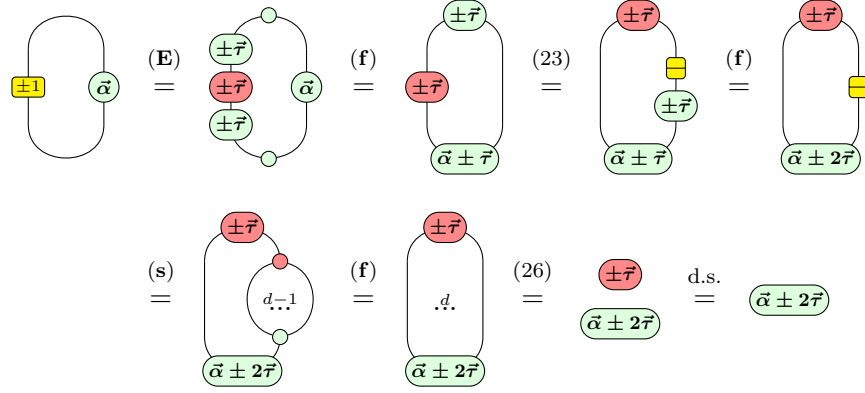
1. Every spider is green.
2. Spiders are only connected by n -Hadamard edges, for $n \in \mathbb{Z}_d$.
3. No spider has a self-loop.
4. Every pair of spiders is connected by at most one edge.
5. Every input and output wire of the diagram is connected to a spider.
6. Every spider is connected to at most one of the diagram's input or output wires.

Once again, every qudit ZX-diagram is equivalent to one that is graph-like; the proof is a simple generalisation of the qutrit case, and uses the following self-loop lemma:

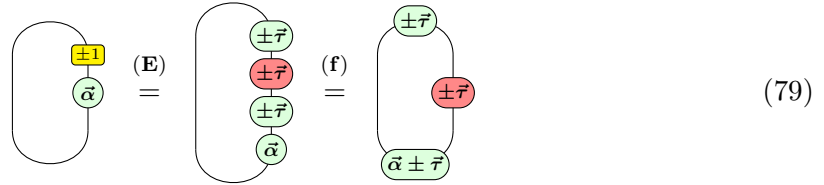
Lemma 5.10.

Proof. As mentioned in the qutrit section, there are several ways to define a self-loop, all of which give the same result. For a plain wire, the proof in all cases just involves simple

spider fusion. For a blue or purple self-loop, the hardest case follows from the following result:



All other cases quickly reduce to the case above, e.g. the final diagram below is just symmetric to the third diagram above:



□

A *graph state* is then again a graph-like diagram in which all phases are trivial and every spider has an output wire. Such a graph state can be specified by an underlying \mathbb{Z}_d -weighted multigraph with defining adjacency matrix $W = (w_{ij})$. Each node corresponds to a phaseless green spider, and edges of weight w correspond to Hadamard w -edges. Each spider is additionally given an output leg which is an output for the overall diagram. The local complementation and pivot operations generalise straightforwardly:

Definition 5.11. Given $a \in \mathbb{Z}_d$ and a graph state G with adjacency matrix $W = (w_{ij})$, the a -local complementation at node x is the new graph state $G *_a x$, whose adjacency matrix $W' = (w'_{ij})$ is given by $w'_{i,j} = w_{i,j} + aw_{i,x}w_{j,x}$.

Definition 5.12. Given $a, b, c \in \mathbb{Z}_d$ and a graph state G containing nodes i and j , the (a, b, c) -pivot along ij is the new graph state $G \wedge_{(a,b,c)} ij := ((G *_a i) *_b j) *_c i$.

As in the qutrit case, we are only interested in the $(\pm 1, \mp 1, \pm 1)$ -pivot: we call this a *proper ± 1 -pivot*. Now, the following conjecture relates a graph state and its local complementation. We say conjecture since we only have only managed to prove the result for *complete* graph states; see Appendix A for more details.

Conjecture 5.13. Given $a \in \mathbb{Z}_d$ and a graph state (G, W) containing a node x , let $N(x)$ denote the neighbours of x ; that is, nodes i with weight $w_{i,x} \neq 0$. Then the following equality holds:

Likewise the next relates a graph state and its proper pivot along an edge ij , where $w_{ij} \neq 0$. Its proof, which is conditional on the result above holding, is found in Appendix C.2:

Theorem 5.14 (Conditional). Given a graph state (G, W) containing connected nodes i and j , let $N(i \cup j) = N(i) \cup N(j)$ denote the union of neighbours of i and j . Furthermore, let $\{N_{\pm 2}(i \cup j), N_{\pm 1}(i \cup j), N_{\mp 1}(i \cup j)\}$ be disjoint subsets of $N(i, j)$, whose precise definitions are deferred to Appendix C.2. Then the following equation relates G and its proper ± 1 -pivot along ij :

5.3 Elimination Theorems

Finally, we see how far our elimination theorems generalise to the qudit case. Assuming the local complementation conjecture above hold, we will show that we can eliminate standalone interior spiders with phases $\langle \pm 1, x \rangle \in \mathcal{S}_d$, and connected interior spiders with phases $\langle 0, -2n \rangle = n\vec{\kappa}$. We will need the following two lemmas:

Lemma 5.15. For any stabilizer phase $\vec{p}^{k,x} = \langle k, x \rangle \in \mathcal{S}_d$, its reverse $\tilde{\vec{p}}^{k,x}$ equals $\langle k, -(2k+x) \rangle$.

Proof. The j -th component of $\vec{p}^{l,y}$ is $\frac{-l}{2}j^2 - \left(\frac{l+y}{2}\right)j$. The j -th component of $\vec{p}^{k,x}$ is the $(d-j)$ -th component of $\vec{p}^{k,x}$, which expands to:

$$\tilde{p}_j^{k,x} = \frac{-k}{2}j^2 + \left(\frac{k+x}{2}\right)j + \left[\frac{d}{2}(k(2j-d-1)-x)\right]$$

Provided $\langle k, x \rangle \in \mathcal{S}_d$, the term in square brackets equals zero mod d (regardless of the parity of d). So $\tilde{p}_j^{k,x} = \frac{-k}{2}j^2 + \left(\frac{k+x}{2}\right)j$. Equating this with $\vec{p}_j^{l,y} = \frac{-l}{2}j^2 - \left(\frac{l+y}{2}\right)j$ forces $l = k$ and $y = -(2k+x)$. \square

Lemma 5.16. States of one colour with phase $\langle \pm 1, x \rangle \in \mathcal{S}_d$ are equivalent to states of the other colour with phase of the form $\langle \mp 1, y \rangle \in \mathcal{S}_d$. Note any $\langle \pm 1, x \rangle \in \mathcal{S}_d$ can be written as $\langle \pm 1, \pm(2x' - 1) - d \rangle$ for some other integer x' . In this form, our lemma is:

$$\begin{array}{c} | \\ \langle -1, 1 - d - 2x \rangle \end{array} = \begin{array}{c} | \\ \langle 1, d - 1 - 2x \rangle \end{array}, \quad \begin{array}{c} | \\ \langle 1, 2x - d - 1 \rangle \end{array} = \begin{array}{c} | \\ \langle -1, 1 - d - 2x \rangle \end{array} \quad (82)$$

Proof. We split into two cases. The simplest is:

$$\begin{array}{c} | \\ \langle -1, 1 - d - 2x \rangle \end{array} \stackrel{(\text{cc})}{=} \begin{array}{c} | \\ \boxed{-1} \\ \langle -1, 1 - d - 2x \rangle \end{array} \stackrel{(\text{E})}{=} \begin{array}{c} \bar{\tau} \\ | \\ \bar{\tau} \\ | \\ \bar{\tau} \\ \langle -1, 1 - d - 2x \rangle \end{array} = \begin{array}{c} \langle 1, d - 1 \rangle \\ | \\ \langle 1, d - 1 \rangle \\ | \\ \langle 1, d - 1 \rangle \\ \langle -1, 1 - d - 2x \rangle \end{array} \\ \stackrel{(\text{f})}{=} \begin{array}{c} \langle 1, d - 1 \rangle \\ | \\ \langle 1, d - 1 \rangle \\ | \\ \langle 0, -2x \rangle \end{array} \stackrel{(\text{z})}{=} \begin{array}{c} \langle 1, d - 1 \rangle \\ | \\ \langle 0, -2x \rangle \end{array} \stackrel{(\text{f})}{=} \begin{array}{c} | \\ \langle 1, d - 1 - 2x \rangle \end{array}$$

The other makes use of Lemma 5.15:

$$\begin{array}{c} | \\ \langle 1, 2x - d - 1 \rangle \end{array} \stackrel{5.15}{=} \begin{array}{c} | \\ \boxed{1} \\ \langle 1, d - 1 - 2x \rangle \end{array} \stackrel{(\text{E})}{=} \begin{array}{c} \bar{\tau} \\ | \\ \bar{\tau} \\ | \\ \bar{\tau} \\ \langle 1, d - 1 - 2x \rangle \end{array} = \begin{array}{c} \langle -1, 1 - d \rangle \\ | \\ \langle -1, 1 - d \rangle \\ | \\ \langle -1, 1 - d \rangle \\ \langle 1, d - 1 - 2x \rangle \end{array} \\ \stackrel{(\text{f})}{=} \begin{array}{c} \langle -1, 1 - d \rangle \\ | \\ \langle -1, 1 - d \rangle \\ | \\ \langle 0, -2x \rangle \end{array} \stackrel{(\text{z})}{=} \begin{array}{c} \langle -1, 1 - d \rangle \\ | \\ \langle 0, -2x \rangle \end{array} \stackrel{(\text{f})}{=} \begin{array}{c} | \\ \langle -1, 1 - d - 2x \rangle \end{array}$$

□

Then, via a suitable choice of a -local complementation, we show we can eliminate spiders carrying such phases. We make a small presentational change as compared to the qutrit analogue: there we wished to make updates to edge weights explicit, but for general d these updates are sufficiently complicated that we are best off abstracting them away. Sadly though, this abstraction doesn't play nicely with our !-box notation, so we make do without the latter. Now, much like a graph state, a graph-like diagram can be thought of as having an underlying graph G (it does not fully define the graph-like diagram, but this doesn't matter for our purposes). We can then consider graph-theoretic operations on G ,

such as local complementation ($*$), pivot (\wedge) and vertex removal (\setminus). For example, suppose G is:

$$\begin{array}{c} \text{---} \text{---} \text{---} \text{---} \\ | \quad | \quad | \quad | \\ \boxed{G} \end{array} = \begin{array}{c} \text{---} \text{---} \text{---} \text{---} \text{---} \\ | \quad | \quad | \quad | \quad | \\ \text{---} \text{---} \text{---} \text{---} \end{array} \quad (83)$$

Let v be the bottom vertex. If we locally complement by 1 at v , then remove v , we get:

$$\begin{array}{c} \text{---} \text{---} \text{---} \text{---} \\ | \quad | \quad | \quad | \\ \boxed{(G *_1 v) \setminus v} \end{array} = \begin{array}{c} \text{---} \text{---} \text{---} \text{---} \text{---} \\ | \quad | \quad | \quad | \quad | \\ \text{---} \text{---} \text{---} \text{---} \end{array} \quad (84)$$

Even for this simple example the right hand side here is rather messy; this is why we hide these edge weights using the notation on the left hand side. We note, however, that these edge weight changes would be easy to compute explicitly – e.g. as part of a circuit optimisation algorithm.

Theorem 5.17 (Conditional). Suppose we have a graph-like diagram containing a node i with phase $\langle \pm 1, y \rangle \in \mathcal{S}_d$. Let $N(i)$ denote the neighbours of i (spiders connected to i by a Hadamard edge of non-zero weight). Then the following equation holds:

$$\begin{array}{c} \text{---} \text{---} \text{---} \text{---} \text{---} \text{---} \\ | \quad | \quad | \quad | \quad | \quad | \\ \alpha_1 \dots \alpha_a \beta_1 \dots \beta_b \dots \gamma_1 \dots \gamma_c \\ | \quad | \quad | \quad | \quad | \quad | \\ \text{---} \text{---} \text{---} \text{---} \text{---} \text{---} \\ \boxed{\langle \pm 1, \pm(2x-1) - d \rangle} \end{array} = \begin{array}{c} \text{---} \text{---} \text{---} \text{---} \text{---} \text{---} \\ | \quad | \quad | \quad | \quad | \quad | \\ \alpha_1 \pm \vec{\tau} - x\vec{\kappa} \quad \beta_1 \pm \vec{\tau} - 2x\vec{\kappa} \quad \gamma_1 \pm \vec{\tau} + x\vec{\kappa} \\ | \quad | \quad | \quad | \quad | \quad | \\ \alpha_a \pm \vec{\tau} - x\vec{\kappa} \quad \beta_b \pm \vec{\tau} - 2x\vec{\kappa} \quad \gamma_c \pm \vec{\tau} + x\vec{\kappa} \\ | \quad | \quad | \quad | \quad | \quad | \\ \text{---} \text{---} \text{---} \text{---} \text{---} \text{---} \\ \boxed{(N(i) *_1 i) \setminus i} \end{array}$$

Note that this really does eliminate a spider: the phases visible on the right hand side of the equation can fuse with the phaseless spiders hidden with the $(N(i) *_1 i) \setminus i$ notation.

Finally, we show the proper pivot elimination theorem for connected spiders with phases $\langle 0, y \rangle$. For clarity we use blue and purple dashed lines for 1- and $(d-1)$ -Hadamard edges, and hide edges of other weights within ellipses. Better notation would be welcome here, however; our neater presentation arguably comes at the cost of making the diagram a little more ambiguous.

Theorem 5.18 (Conditional). Suppose we have a graph-like diagram containing connected nodes i and j with phases $\langle 0, 2x \rangle$ and $\langle 0, 2y \rangle \in \mathcal{S}_d$ respectively. If a node is connected to i by an a -Hadamard edge and to j by a b -Hadamard edge, we subscript its phase with a, b . We also let $\vec{\beta}_{a,b}$ be a phase whose precise definition is deferred to the proof in Appendix

D.2. Then the following equation holds:

$$\begin{array}{c}
 \begin{array}{c}
 \begin{array}{cccccccc}
 \alpha_{1,0} & \dots & \alpha_{-1,0} & \alpha_{1,1} & \dots & \alpha_{1,-1} & \dots & \alpha_{-1,-1} & \dots & \alpha_{-1,-1} & \alpha_{0,1} & \dots & \alpha_{0,-1}
 \end{array} \\
 \vdots & & \vdots & \vdots & & \vdots & & \vdots & & \vdots & \vdots & & \vdots \\
 \langle 0, 2x \rangle & \text{---} & \text{yellow box } w & \text{---} & \langle 0, 2y \rangle
 \end{array} \\
 \\
 = \begin{array}{c}
 \begin{array}{cccccccc}
 \alpha_{1,0} + \beta_{1,0} & \dots & \alpha_{-1,0} + \beta_{-1,0} & \alpha_{1,1} + \beta_{1,1} & \dots & \alpha_{1,-1} + \beta_{1,-1} & \dots & \alpha_{-1,-1} + \beta_{-1,-1} & \dots & \alpha_{-1,-1} + \beta_{-1,-1} & \alpha_{0,1} + \beta_{0,1} & \dots & \alpha_{0,-1} + \beta_{0,-1}
 \end{array} \\
 \vdots & & \vdots & \vdots & & \vdots & & \vdots & & \vdots & \vdots & & \vdots \\
 \boxed{(N(i \cap j) \wedge_{\pm 1} ij) \setminus \{i, j\}}
 \end{array}
 \end{array}
 \tag{85}$$

Again, we note this really does eliminate nodes i and j . Of course, we would ideally show that *any* qudit stabilizer ZX-diagram can be efficiently simplified. A possible strategy to prove this is to try find elimination theorems for the remaining standalone interior spiders with phases $\langle k, x \rangle$ for $k \in \{2, \dots, d-2\}$. For example, suppose we could show that a state of one colour with phase $\langle k, x \rangle \in \mathcal{S}_d$ is equivalent to a state of the other colour with phase $\langle l, y \rangle \in \mathcal{S}_d$, for $k, l \in \{2, \dots, d-2\}$:

$$\begin{array}{c} | \\ \text{green circle } \langle k, x \rangle \end{array} = \begin{array}{c} | \\ \text{red circle } \langle l, y \rangle \end{array}
 \tag{86}$$

Then a suitable choice of a -local complementation would again lead to an elimination theorem for an interior spider with such a phase. As yet, we have neither been able to find k, l, x and y satisfying equation (86) nor prove it isn't always satisfiable.

6 Case studies

In this section we present two problems which can naturally be cast in tensor network form. Tensor networks are widely used throughout physics, so methods to simplify them are highly sought-after [43]. The problems we discuss here are interesting in that they show a transition in complexity when the dimension d carried by the wires of the network is greater than a specific value.

The *Jones polynomial* is a prominent *link invariant* in knot theory, a branch of topology. One of the most common tasks in knot theory is to decide whether two *links* are in some sense the same or genuinely different. The Jones polynomial was the first effective tool for tackling such problems, and its discovery in 1984 won Vaughan Jones the Fields medal [32]. here we present the problem of showing it can be efficiently evaluated at the *lattice roots of unity*. Knot theory and quantum computing have a surprisingly large overlap [26], and we take a couple of quick detours to remark upon how our work fits into the *topological quantum computing* paradigm [53]. These remarks can be safely skipped over by any reader who has insufficient interest or background to digest them. Likewise, the bridge connecting tensor networks to the Jones polynomial is one largely sketched out by physicists, and so we avoid the jargon ordinarily required to traverse it.

Though the result we present is well-known to knot theorists, the proof given here is entirely novel, and demonstrates the broad range of problems that graphical calculi like the ZX-calculus can tackle. Subsequently, we briefly look at the problem of counting graph colourings.

6.1 The Jones Polynomial at Lattice Roots of Unity

We wish to introduce only as much knot theory here as is needed to set up and solve our particular problem. For the interested reader, we highly recommend Prasolov and Sossinsky’s introductory text on low-dimensional topology [42].

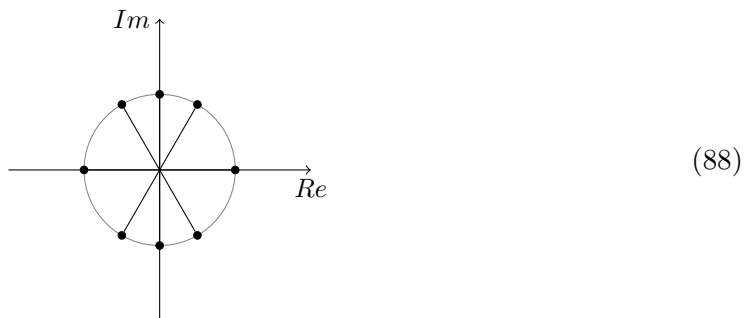
A *link* is a smooth embedding of n disjoint circles into \mathbb{R}^3 . A *knot* is a 1-component link, i.e. it is a single circle smoothly embedded in \mathbb{R}^3 . A link L can be represented by a *link diagram* by projecting it onto a plane but retaining the information of *over or under crossings*. For example, a link diagram for the *trefoil knot* linked with the *unknot* is:



(87)

We say that two links are isotopic, written $L \simeq L'$, if and only if the link L can be deformed to the link L' without cutting or gluing strands, or passing strands through each other. A more rigorous description can be found in [42]. The *Jones polynomial* $V_L(t)$ of a link L is a Laurent polynomial in a variable $t \in \mathbb{C}$. It is a *link invariant*: if two links have different Jones polynomials then they cannot be isotopic. Again, for more details, see [42].

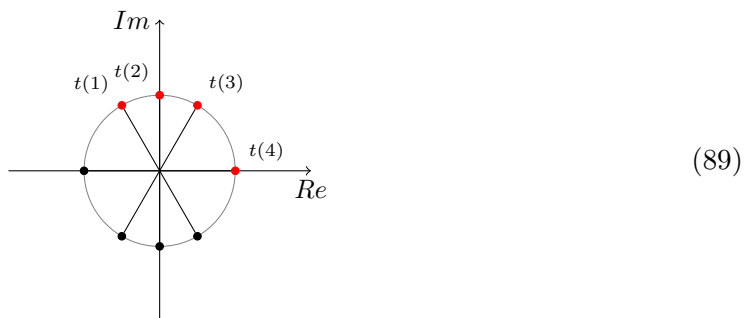
In general, computing $V_L(t)$ is exponentially costly in the number of crossings c of the link L , something made explicit when one uses the Kauffman bracket method [33]. Exactly evaluating the Jones polynomial at points $t \in \mathbb{C}$ is $\#\mathbf{P}$ -hard, except at the *lattice roots of unity* $\Lambda = \{\pm 1, \pm i, \pm e^{i2\pi/3}, \pm e^{i4\pi/3}\}$ shown below, where it can be evaluated at cost $O(\text{poly}(c))$ [31]:



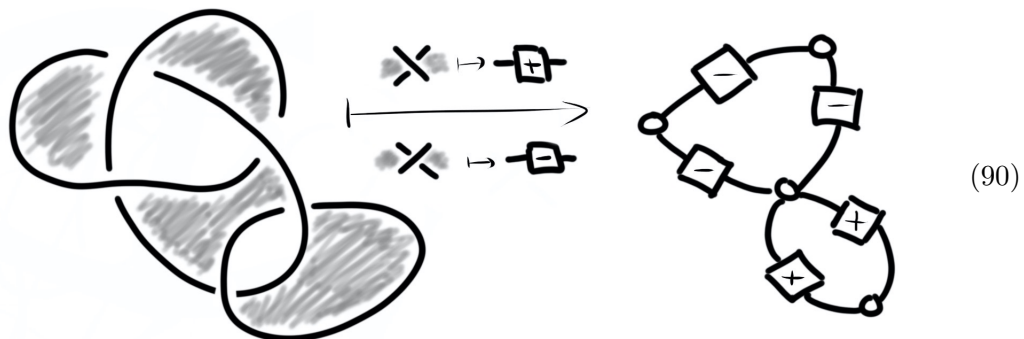
Remark 6.1 (For topological quantum computing enthusiasts). In the context of quantum computation, additively approximating the Jones polynomial at *non-lattice roots of unity* is the paradigmatic BQP-complete problem [1, 37]. Topological quantum computation [24, 41] is the most natural model for such knot theoretic questions. In this model, quantum states are defined in the fusion space of *non-abelian anyons*, emergent quasi-particles with non-trivial exchange statistics arising in two-dimensional exotic phases of matter. Quantum computation is performed by creating anyons from the vacuum, then braiding them, and finally fusing them back to the vacuum. Thus braids play the role of unitary gates. The world-lines of the anyons define a closed braid, i.e. a link. Such a link encodes a quantum amplitude corresponding to its Jones polynomial evaluated at a root of unity depending on the anyon theory at hand [54]. Specifically, in the case of $SU(2)_k$ anyons, the Jones polynomial is evaluated at $t(k) = e^{i2\pi/(2+k)}$ [45].

Let $t(d) = \frac{1}{2}(d - 2 \pm \sqrt{d(d-4)})$, for some natural number d . For $d \in \{1, 2, 3, 4\}$, we draw

these points – note that these are all lattice roots of unity:



Remarkably, the evaluation of a Jones polynomial at a point $t(d)$ can be expressed as a tensor network, obtained as follows. First, take a link diagram for L and colour it black and white checkerboard-style, i.e. such that each region in the diagram for L has a different colour to all of its neighbouring regions. Up to permuting the two colours, there is exactly one way to checkerboard colour any link diagram; fixing the background to be white gives a unique colouring. Next, map every black (‘shaded’) area to a vertex and every crossing to a *signed* edge according to its orientation relative to the surrounding colours, as in (90) below. This sign is denoted by a box marked with a \pm sign.



What we end up with is a string diagram with two types of morphism: the vertices and the \pm -boxes. We interpret wires as d -dimensional complex vector spaces \mathbb{C}^d with basis $\{|0\rangle, \dots, |d-1\rangle\}$, and the morphisms as the linear maps below. The second of these is a square $d \times d$ matrix. We also note the former is commonly called the COPY tensor:

$$\left[\begin{array}{c} \overbrace{\quad}^m \\ \vdots \\ \text{---} \\ \vdots \\ \underbrace{\quad}_n \end{array} \right] = \sum_{j=0}^{d-1} |j\rangle^{\otimes m} \langle j|^{\otimes n} \quad (91)$$

$$\left[\begin{array}{c} | \\ \boxed{\pm} \\ | \end{array} \right] = \begin{pmatrix} -t(d)^{\mp 1} & 1 & \dots & 1 \\ 1 & -t(d)^{\mp 1} & \dots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \dots & -t(d)^{\mp 1} \end{pmatrix} \quad (92)$$

The evaluation of the Jones polynomial at $t(d)$ is then equal to the scalar represented by this tensor network, up to an efficiently computable scalar which depends on the link diagram [40].

Remark 6.2. Surprisingly, this method is independent of the chosen link diagram for L . That is, we could choose a different link diagram for L , which would lead to a different string diagram, but the scalar this string diagram represents under the interpretation above would be identical to the scalar we would have obtained using the original link diagram for L [40].

Remark 6.3 (For topological quantum computing enthusiasts). Note the correspondence between the dimension d in the tensor network approach and the level k in the anyon-braiding approach to the Jones polynomial: $\{t(d) \mid d \in \{1, 2, 3, 4\}\} = \{t(k) \mid k \in \{1, 2, 4, \infty\}\} \subseteq \Lambda$. This is consistent with the fact that braiding $SU(2)_2$ anyons (Ising) or $SU(2)_4$ anyons is not universal (unless the $SU(2)_4$ anyons are augmented by fusion and measurements [38]).

We can equivalently express the tensor network above as a qudit ZX-diagram. Recall the standard interpretation functor $\llbracket - \rrbracket$ for the qudit ZX-calculus (17). Under this, we can see that the white vertices are exactly phaseless green spiders:

$$\left[\begin{array}{c} \overbrace{\quad}^m \\ \dots \\ \text{white vertex} \\ \dots \\ \underbrace{\quad}_n \end{array} \right] = \left[\begin{array}{c} \overbrace{\quad}^m \\ \dots \\ \text{green spider} \\ \dots \\ \underbrace{\quad}_n \end{array} \right] \quad (93)$$

Then the \pm -matrices above for $d \in \{2, 4\}$ are equal up to a scalar to standard interpretations of *qubit stabilizer* ZX-diagrams, and the \pm -matrices for $d = 3$ of *qutrit stabilizer* ZX-diagrams (see Appendix E.1):

$$\left[\begin{array}{c} \text{red circle with } \pm \frac{1}{2} \\ | \end{array} \right] \simeq \left[\begin{array}{c} | \\ \boxed{\pm} \\ | \end{array} \right]_{d=2}, \quad \left[\begin{array}{c} \text{red circle with } \pm \frac{1}{3} \\ | \end{array} \right] \simeq \left[\begin{array}{c} | \\ \boxed{\pm} \\ | \end{array} \right]_{d=3}, \quad \left[\begin{array}{c} \text{red circle with } \pi \text{ --- yellow square --- red circle with } \pi \\ | \end{array} \right] \simeq \left[\begin{array}{c} | \\ \boxed{\pm} \\ | \end{array} \right]_{d=4} \quad (94)$$

Thus our tensor networks representing evaluations of the Jones polynomial can be evaluated *efficiently*, using either the qubit stabilizer simplification algorithm from Ref. [20] or our

corresponding qutrit version (4.11). For the case $d = 1$, the result is trivial: our $d \times d$ matrices are just scalars, so finding the scalar represented by the tensor network only requires efficient arithmetic operations. In fact, it is well-known amongst knot theorists that evaluating any Jones polynomial of any link at $t(1) = e^{\frac{2\pi i}{3}}$ just gives 1 [31].

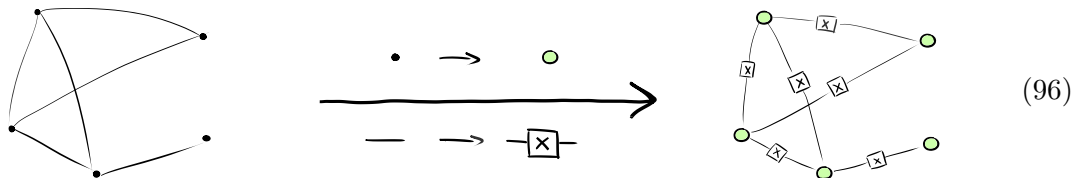
Finally, knowing the Jones polynomials of a link L at the first four lattice roots of unity $\{t(d) \mid d = 1, 2, 3, 4\}$ allows us to calculate the value of the Jones at the remaining four lattice roots of unity [31]. Thus, we recover the known result that evaluating the Jones polynomial at $t \in \Lambda$ is in \mathbf{P} . For $d \geq 5$, $t(d)$ is not a lattice root of unity, and indeed computing the scalar represented by our tensor networks in dimension $d \geq 5$ is $\#\mathbf{P}$ -hard. We expect this to be reflected in the fact that a d -dimensional \pm -box is not in the stabilizer fragment of the ZX-calculus.

6.2 Graph Colouring

Finally, let us briefly look at the *graph colouring problem*. A d -colouring of a graph G is an assignment of colours $\{1, \dots, d\}$ to the vertices of G so that no neighbouring vertices have the same colour. Given a graph G and an integer d , we wish to count the number of such d -colourings. Again, this problem can be interpreted as a tensor network. All we need to do is place on each edge of G an X -box, then interpret each vertex as the COPY tensor and each X -box as the following linear map:

$$\left[\begin{array}{c} \text{---} \\ | \\ \boxed{X} \\ | \\ \text{---} \end{array} \right] = \begin{pmatrix} 0 & 1 & \dots & 1 \\ 1 & 0 & \dots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \dots & 0 \end{pmatrix} \quad (95)$$

Now the scalar represented by this tensor network is exactly the number of possible d -colourings of G [46]. Again, note that the COPY tensor represented by a vertex is exactly the standard interpretation $\llbracket - \rrbracket$ of the green spider in the qudit ZX-calculus. As an aside, the X -box is just a special case of the \pm -box, where $t = 0$. An example:



Counting d -colourings for $d \geq 3$ is a canonical $\#\mathbf{P}$ -complete problem, while for $d \leq 2$ the problem is in \mathbf{P} [31]. Indeed, for $d = 0, 1$ the problem is trivial. For $d = 2$ the X -matrix is

just the (standard interpretation of the) Pauli X in the qubit ZX-calculus. So the tensor network can be expressed as a ZX-diagram, and moreover, this diagram is in the stabilizer fragment, thus can be efficiently simplified.

$$\left[\begin{array}{c} | \\ \boxed{X} \\ | \end{array} \right]_{d=2} \simeq \left[\begin{array}{c} | \\ \textcolor{red}{\pm\pi} \\ | \end{array} \right] \quad (97)$$

However, for $d = 3$ the X -matrix cannot be expressed as a stabilizer qutrit diagram; we prove this using our qutrit elimination theorems in Appendix E.3. Thus it is not expected that there exists an efficient simplification strategy for the corresponding qutrit ZX-diagram. This is consistent with the fact that counting 3-colourings is $\#\mathbf{P}$ -complete.

7 Outlook

In this work, we have derived crucial *elimination theorems* for the qutrit ZX-calculus, enabling us to demonstrate that any qutrit stabilizer diagram can be efficiently simplified, and laying the groundwork for a qutrit circuit optimisation algorithm. This also generalises the Gottesman-Knill theorem for qutrits. We have investigated how far our work generalises to the qudit case; we have classified what it means for a ZX-diagram to be stabilizer in terms of spider phases, and have conjectured a local complementation rule, which we have partially proved. Assuming this holds in all cases, we derived pivot rules and found elimination theorems for certain families of spiders.

We have used these results to advocate for the use of the ZX-calculus as a convenient graphical framework for treating a broad range of many-body problems on equal footing, and for diagnosing the complexity of interesting families of problems. To this end, we looked at two case studies. Specifically, we recovered known complexity results on evaluating the Jones polynomial and counting graph colourings – two problems which can be cast in tensor network form.

There are many avenues for future work here. The most glaring entails completing the proof of the qudit local complementation result (Conjecture 5.13), closely followed by completing the generalisation of stabilizer elimination theorems to dimension d ; we are currently lacking results for spiders with phases $\langle k, x \rangle$ for $k \in \{2, \dots, d-2\}$. A substantial next goal would be to flesh out qudit circuit extraction ideas – it would be nice to effectively complete the other half of our qutrit circuit optimisation algorithm. It would also be of interest to investigate how the work here translates when the red spider is replaced by its flexsymmetric cousin from Equation (38); the qudit ZX-calculus would become a much more attractive language were *Only Connectivity Matters* to be restored.

Our claim that the ZX-calculus is a useful tool outside of quantum computing would be strengthened if we could find a problem – likely one that can be encoded as a tensor network – for which the ZX-calculus either finds a solution where none could previously be found, or finds a solution in a much faster time than other methods, or similar. While the methods we’ve developed here work well for answering complexity-based questions, calculating specific scalars with these techniques requires scalar-exact versions of all of our results – deriving these constitutes a further possible path for future work.

References

- [1] D. Aharonov, V. Jones, and Z. Landau. A polynomial quantum algorithm for approximating the jones polynomial. *Algorithmica*, 55(3):395–421, Mar 2008.
- [2] S. Awodey. *Category theory*. Oxford university press, 2010.
- [3] M. Backens. Making the stabilizer zx-calculus complete for scalars. *Electronic Proceedings in Theoretical Computer Science*, 195:17–32, Nov 2015.
- [4] M. Backens and A. Kissinger. Zh: A complete graphical calculus for quantum computations involving classical non-linearity, 2018.
- [5] C. H. Bennett, G. Brassard, C. Crépeau, R. Jozsa, A. Peres, and W. K. Wootters. Teleporting an unknown quantum state via dual classical and einstein-podolsky-rosen channels. *Physical review letters*, 70(13):1895, 1993.
- [6] T. Bækkegaard, L. B. Kristensen, N. J. S. Loft, C. K. Andersen, D. Petrosyan, and N. T. Zinner. Realization of efficient quantum gates with a superconducting qubit-qutrit circuit. *Scientific Reports*, 9(1), Sep 2019.
- [7] T. Carlette. When only topology matters.
- [8] N. Chancellor, A. Kissinger, J. Roffe, S. Zohren, and D. Horsman. Graphical structures for design and verification of quantum error correction, 2018.
- [9] B. Coecke, G. de Felice, K. Meichanetzidis, and A. Toumi. Foundations for near-term quantum natural language processing, 2020.
- [10] B. Coecke and R. Duncan. Interacting quantum observables: categorical algebra and diagrammatics. *New Journal of Physics*, 13(4):043016, Apr 2011.
- [11] B. Coecke and A. Kissinger. Picturing quantum processes. In *International Conference on Theory and Application of Diagrams*, pages 28–31. Springer, 2018.
- [12] A. Cowtan, S. Dilkes, R. Duncan, W. Simmons, and S. Sivarajah. Phase Gadget Synthesis for Shallow Circuits. In B. Coecke and M. Leifer, editors, Proceedings 16th International Conference on *Quantum Physics and Logic*, Chapman University, Orange, CA, USA., 10-14 June 2019, volume 318 of *Electronic Proceedings in Theoretical Computer Science*, pages 213–228. Open Publishing Association, 2020.
- [13] C. Damm, M. Holzer, and P. McKenzie. The complexity of tensor calculus. *computational complexity*, 11(1):54–89, 2002.
- [14] N. de Beaudrap, X. Bian, and Q. Wang. Fast and effective techniques for T-count reduction via spider nest identities. *arXiv preprint arXiv:2004.05164*, 2020.

- [15] N. de Beaudrap, X. Bian, and Q. Wang. Techniques to Reduce $\pi/4$ -Parity-Phase Circuits, Motivated by the ZX Calculus. In B. Coecke and M. Leifer, editors, Proceedings 16th International Conference on *Quantum Physics and Logic*, Chapman University, Orange, CA, USA., 10-14 June 2019, volume 318 of *Electronic Proceedings in Theoretical Computer Science*, pages 131–149. Open Publishing Association, 2020.
- [16] N. de Beaudrap and D. Horsman. The zx calculus is a language for surface code lattice surgery. *Quantum*, 4:218, Jan 2020.
- [17] N. de Beaudrap, A. Kissinger, and K. Meichanetzidis. Tensor network rewriting strategies for satisfiability and counting, 2020.
- [18] L. Dixon and R. Duncan. Graphical reasoning in compact closed categories for quantum computation. *Annals of Mathematics and Artificial Intelligence*, 56(1):23–42, 2009.
- [19] R. Duncan. A graphical approach to measurement-based quantum computing. In M. S. Chris Heunen and E. Grefenstette, editors, *Quantum Physics and Linguistics: A Compositional, Diagrammatic Discourse*. 2013.
- [20] R. Duncan, A. Kissinger, S. Perdrix, and J. van de Wetering. Graph-theoretic simplification of quantum circuits with the zx-calculus. *Quantum*, 4:279, Jun 2020.
- [21] R. Duncan and S. Perdrix. Graphs states and the necessity of euler decomposition, 2009.
- [22] S. Eilenberg and S. MacLane. General theory of natural equivalences. *Transactions of the American Mathematical Society*, 58(2):231–294, 1945.
- [23] J. M. Farinholt. An ideal characterization of the clifford operators. *Journal of Physics A: Mathematical and Theoretical*, 47(30):305303, Jul 2014.
- [24] M. H. Freedman, A. Kitaev, M. J. Larsen, and Z. Wang. Topological quantum computation. *Bulletin of the American Mathematical Society*, 40(01):31–39, Oct. 2002.
- [25] L. Garvie and R. Duncan. Verifying the Smallest Interesting Colour Code with Quantumomatic. In B. Coecke and A. Kissinger, editors, *Proceedings 14th International Conference on Quantum Physics and Logic, Nijmegen, The Netherlands, 3-7 July 2017*, volume 266 of *Electronic Proceedings in Theoretical Computer Science*, pages 147–163. Open Publishing Association, 2018.
- [26] R. Gaudreau and D. Ledvinka. Knot theory and quantum computing, 2019.
- [27] X. Gong and Q. Wang. Equivalence of local complementation and euler decomposition in the qutrit zx-calculus, 2017.
- [28] J. Gray and S. Kourtis. Hyper-optimized tensor network contraction, 2020.

- [29] C. Heunen and J. Vicary. *Categories for Quantum Theory: an introduction*. Oxford University Press, USA, 2019.
- [30] E. Hostens, J. Dehaene, and B. De Moor. Stabilizer states and clifford operations for systems of arbitrary dimensions and modular arithmetic. *Physical Review A*, 71(4), Apr 2005.
- [31] F. Jaeger, D. L. Vertigan, and D. J. A. Welsh. On the computational complexity of the jones and tutte polynomials. *Mathematical Proceedings of the Cambridge Philosophical Society*, 108(1):35–53, 1990.
- [32] V. F. Jones. A polynomial invariant for knots via von neumann algebras. In *Fields Medallists’ Lectures*, pages 448–458. World Scientific, 1997.
- [33] L. H. Kauffman. *Knots and Physics*. WORLD SCIENTIFIC, July 2001.
- [34] A. Kissinger and J. van de Wetering. Universal mbqc with generalised parity-phase interactions and pauli measurements. *Quantum*, 3:134, Apr 2019.
- [35] A. Kissinger and J. van de Wetering. Reducing T-count with the ZX-calculus. *Physical Review A*, 102:022406, 8 2020.
- [36] A. Kissinger and V. Zamdzhiev. Quantomatic: A proof assistant for diagrammatic reasoning. In *International Conference on Automated Deduction*, pages 326–336. Springer, 2015.
- [37] G. Kuperberg. How hard is it to approximate the jones polynomial?, 2014.
- [38] C. Levaillant, B. Bauer, M. Freedman, Z. Wang, and P. Bonderson. Universal gates via fusion and measurement operations on $su(2)_4$ anyons. *Physical Review A*, 92(1), Jul 2015.
- [39] K. Meichanetzidis, S. Gogioso, G. D. Felice, N. Chiappori, A. Toumi, and B. Coecke. Quantum natural language processing on near-term quantum computers, 2020.
- [40] K. Meichanetzidis and S. Kourtis. Evaluating the jones polynomial with tensor networks. *Phys. Rev. E*, 100:033303, Sep 2019.
- [41] J. K. Pachos. *Introduction to Topological Quantum Computation*. Cambridge University Press, 2012.
- [42] V. V. Prasolov and A. B. Sosinskiĭ. *Knots, links, braids and 3-manifolds: an introduction to the new invariants in low-dimensional topology*. Number 154. American Mathematical Soc., 1997.
- [43] S.-J. Ran, E. Tirrito, C. Peng, X. Chen, G. Su, and M. Lewenstein. Review of tensor network contraction approaches. *arXiv preprint arXiv:1708.09213*, 2017.

- [44] M. Rédei. Why john von neumann did not like the hilbert space formalism of quantum mechanics (and what he liked instead). 1996.
- [45] E. C. Rowell and Z. Wang. Mathematics of topological quantum computing. *Bulletin of the American Mathematical Society*, 55(2):183–238, Jan 2018.
- [46] A. D. Sokal. Chromatic polynomials, potts models and all that. *Physica A: Statistical Mechanics and its Applications*, 279(1-4):324–332, May 2000.
- [47] A. Toumi, R. Yeung, and G. de Felice. Diagrammatic differentiation for quantum machine learning, 2021.
- [48] J. van de Wetering. Zx-calculus for the working quantum computer scientist, 2020.
- [49] J. Von Neumann. *Mathematische grundlagen der quantenmechanik*, volume 38. Springer-Verlag, 2013.
- [50] Q. Wang. *Completeness of the ZX-calculus*. PhD thesis, 2018.
- [51] Q. Wang. Qutrit zx-calculus is complete for stabilizer quantum mechanics. *Electronic Proceedings in Theoretical Computer Science*, 266:58–70, Feb 2018.
- [52] Q. Wang. Completeness of algebraic zx-calculus over arbitrary commutative rings and semirings, 2020.
- [53] Z. Wang. *Topological quantum computation*. Number 112. American Mathematical Soc., 2010.
- [54] E. Witten. Quantum field theory and the jones polynomial. *Communications in Mathematical Physics*, 121(3):351–399, Sept. 1989.

A Local Complementation

In this appendix we give all of the proofs that are sufficiently complex as to detract from the main text. We try to prove results here as generally as possible. This first section is devoted to a partial proof of the qudit local complementation theorem. We warm up with a few lemmas:

Lemma A.1. The phases $\pm\vec{\tau}$ are their own reverse: that is, $\pm\tilde{\tau} = \pm\vec{\tau}$. In fact, this property holds for a family of stabilizer phases: any $\langle -x, x \rangle \in \mathcal{S}_d$ is its own reverse (where, importantly, we recall that the first argument of $\langle -, - \rangle$ is taken mod d).

Proof. In Lemma 5.15 we said the reverse of $\langle k, x \rangle$ is $\langle k, -(2k+x) \rangle$. So equating $-(2k+x)$ with x mod $2d$ forces $k = -x$ mod d . Then just recall $\pm\vec{\tau} = \pm\langle -1, 1-d \rangle$. \square

Now some diagrammatic equalities:

Lemma A.2. The following equations all hold in the ZX-calculus:

$$\begin{array}{c} \text{cup with } \vec{\alpha} \text{ and red dot} \end{array} = \begin{array}{c} \text{cup with } \vec{\alpha} \text{ and red dot} \end{array}, \quad \begin{array}{c} \text{red circle } \vec{\tau} \end{array} = \begin{array}{c} \text{stack of } -\vec{\tau} \text{ and } -\vec{\tau} \end{array}, \quad \begin{array}{c} \text{red circle } \pm\vec{\tau} \end{array} = \begin{array}{c} \text{green circle } \mp\vec{\tau} \end{array} \quad (98)$$

Proof. For the first equation:

$$\begin{array}{c} \text{cup with } \vec{\alpha} \text{ and red dot} \end{array} \stackrel{(23)}{=} \begin{array}{c} \text{cup with } \vec{\alpha} \text{ and red dot, with } -1 \text{ box} \end{array} \stackrel{(\text{id})}{=} \begin{array}{c} \text{cup with } \vec{\alpha} \text{ and red dot, with } -1 \text{ box} \end{array} \stackrel{(23)}{=} \begin{array}{c} \text{cup with } \vec{\alpha} \text{ and red dot, with } -1 \text{ box} \end{array} \stackrel{(25)}{=} \begin{array}{c} \text{cup with } \vec{\alpha} \text{ and red dot, with } -1 \text{ box} \end{array} \stackrel{(\text{H})}{=} \begin{array}{c} \text{cup with } \vec{\alpha} \text{ and red dot} \end{array}$$

For the second:

$$\begin{array}{c} \text{red circle } \vec{\tau} \end{array} \stackrel{(\text{s})}{=} \begin{array}{c} \text{stack of } -1 \text{ and } \vec{\tau} \end{array} \stackrel{(\text{cc})}{=} \begin{array}{c} \text{stack of } -1 \text{ and } \vec{\tau} \end{array} \stackrel{(\text{E})}{=} \begin{array}{c} \text{stack of } -\vec{\tau} \text{ and } -\vec{\tau} \end{array} \stackrel{(\text{f})}{=} \begin{array}{c} \text{stack of } -\vec{\tau} \text{ and } -\vec{\tau} \end{array} \stackrel{(\text{cc})}{=} \begin{array}{c} \text{stack of } -\vec{\tau} \text{ and } -\vec{\tau} \end{array}$$

The third equation is just a special case of Lemma 5.16 with $x = 0$ mod $2d$. \square

We use all four of the above in the following *triangle lemma*:

Lemma A.3.

$$\begin{array}{c} \text{triangle of nodes with } \bar{\tau} \end{array} = \begin{array}{c} \text{two } -2\bar{\tau} \text{ nodes and } \bar{\tau} \text{ node} \end{array} \quad (99)$$

Proof.

$$\begin{array}{l}
 \begin{array}{c} \text{triangle} \end{array} \stackrel{(23)}{=} \begin{array}{c} \text{crossed lines with } \bar{\tau} \end{array} \stackrel{(f)}{=} \begin{array}{c} \text{crossed lines with } \bar{\tau} \text{ and } 1 \end{array} \\
 \stackrel{A.2}{=} \begin{array}{c} \text{crossed lines with } \bar{\tau} \text{ and } 1 \end{array} \stackrel{A.1}{=} \begin{array}{c} \text{crossed lines with } \bar{\tau} \text{ and } 1 \end{array} \stackrel{(f)}{=} \begin{array}{c} \text{crossed lines with } \bar{\tau} \text{ and } 1 \end{array} \\
 \stackrel{(E)}{=} \begin{array}{c} \text{crossed lines with } \bar{\tau} \text{ and } 1 \end{array} \stackrel{A.1}{=} \begin{array}{c} \text{crossed lines with } \bar{\tau} \text{ and } 1 \end{array} \stackrel{A.2}{=} \begin{array}{c} \text{crossed lines with } \bar{\tau} \text{ and } 1 \end{array} \\
 \stackrel{(f)}{=} \begin{array}{c} \text{crossed lines with } \bar{\tau} \text{ and } 1 \end{array} \stackrel{(b)}{=} \begin{array}{c} \text{crossed lines with } \bar{\tau} \text{ and } 1 \end{array} \\
 \stackrel{A.2}{=} \begin{array}{c} \text{crossed lines with } \bar{\tau} \text{ and } 1 \end{array} \stackrel{(f)}{=} \begin{array}{c} \text{crossed lines with } \bar{\tau} \text{ and } 1 \end{array}
 \end{array}$$

□

Now, let K_n denote the complete graph on K vertices, wherein every pair of distinct vertices have an edge of weight 1 between them. The following lemma tells us the cost of removing an edge in the K_3 graph state:

Lemma A.4.

$$\begin{array}{c} \text{Diagram 1} \end{array} = \begin{array}{c} \text{Diagram 2} \end{array} \quad (100)$$

Diagram 1: A triangle graph with three vertices. The bottom-left vertex is connected to the other two by dashed blue lines. The top vertex is connected to the other two by solid black lines. The bottom-right vertex is connected to the other two by solid black lines.

Diagram 2: A triangle graph with three vertices. The bottom-left vertex is connected to the other two by dashed blue lines. The top-left vertex is connected to the other two by solid black lines and has a red circle labeled $\bar{\tau}$ above it. The top-right vertex is connected to the other two by solid black lines and has a green circle labeled $-\bar{\tau}$ above it. The bottom-right vertex is connected to the other two by solid black lines and has a green circle labeled $-\bar{\tau}$ above it.

Proof.

$$\begin{array}{c} \text{Diagram 1} \end{array} \stackrel{(f)}{=} \begin{array}{c} \text{Diagram 2} \end{array} \stackrel{(cc)}{=} \begin{array}{c} \text{Diagram 3} \end{array} \stackrel{(E)}{=} \begin{array}{c} \text{Diagram 4} \end{array} \stackrel{(f)}{=} \begin{array}{c} \text{Diagram 5} \end{array} \stackrel{A.3}{=} \begin{array}{c} \text{Diagram 6} \end{array} \stackrel{(cc)}{=} \begin{array}{c} \text{Diagram 7} \end{array} \stackrel{A.2}{=} \begin{array}{c} \text{Diagram 8} \end{array} \stackrel{(cc)}{=} \begin{array}{c} \text{Diagram 9} \end{array}$$

Diagram 1: A triangle graph with three vertices. The bottom-left vertex is connected to the other two by solid black lines. The top vertex is connected to the other two by solid black lines. The bottom-right vertex is connected to the other two by solid black lines. Each edge has a yellow box labeled 1.

Diagram 2: A triangle graph with three vertices. The bottom-left vertex is connected to the other two by solid black lines. The top vertex is connected to the other two by solid black lines. The bottom-right vertex is connected to the other two by solid black lines. Each edge has a yellow box labeled 1.

Diagram 3: A triangle graph with three vertices. The bottom-left vertex is connected to the other two by solid black lines. The top vertex is connected to the other two by solid black lines. The bottom-right vertex is connected to the other two by solid black lines. Each edge has a yellow box labeled 1.

Diagram 4: A triangle graph with three vertices. The bottom-left vertex is connected to the other two by solid black lines. The top vertex is connected to the other two by solid black lines. The bottom-right vertex is connected to the other two by solid black lines. Each edge has a yellow box labeled 1. The top-left vertex has a red circle labeled $\bar{\tau}$ above it. The top-right vertex has a red circle labeled $\bar{\tau}$ above it. The bottom-right vertex has a red circle labeled $\bar{\tau}$ above it.

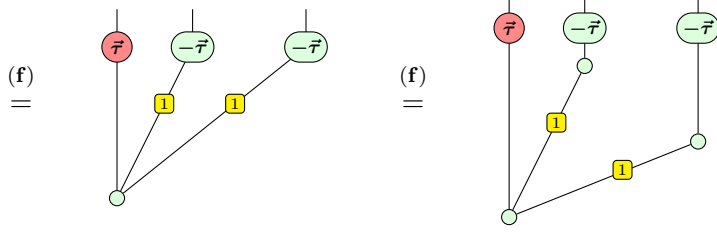
Diagram 5: A triangle graph with three vertices. The bottom-left vertex is connected to the other two by solid black lines. The top vertex is connected to the other two by solid black lines. The bottom-right vertex is connected to the other two by solid black lines. Each edge has a yellow box labeled 1. The top-left vertex has a green circle labeled $\bar{\tau}$ above it. The top-right vertex has a green circle labeled $\bar{\tau}$ above it. The bottom-right vertex has a red circle labeled $\bar{\tau}$ above it.

Diagram 6: A triangle graph with three vertices. The bottom-left vertex is connected to the other two by solid black lines. The top vertex is connected to the other two by solid black lines. The bottom-right vertex is connected to the other two by solid black lines. Each edge has a yellow box labeled 1. The top-left vertex has a green circle labeled $\bar{\tau}$ above it. The top-right vertex has a green circle labeled $\bar{\tau}$ above it. The bottom-right vertex has a red circle labeled $\bar{\tau}$ above it.

Diagram 7: A triangle graph with three vertices. The bottom-left vertex is connected to the other two by solid black lines. The top vertex is connected to the other two by solid black lines. The bottom-right vertex is connected to the other two by solid black lines. Each edge has a yellow box labeled 1. The top-left vertex has a green circle labeled $\bar{\tau}$ above it. The top-right vertex has a green circle labeled $\bar{\tau}$ above it. The bottom-right vertex has a red circle labeled $\bar{\tau}$ above it.

Diagram 8: A triangle graph with three vertices. The bottom-left vertex is connected to the other two by solid black lines. The top vertex is connected to the other two by solid black lines. The bottom-right vertex is connected to the other two by solid black lines. Each edge has a yellow box labeled 1. The top-left vertex has a green circle labeled $\bar{\tau}$ above it. The top-right vertex has a green circle labeled $\bar{\tau}$ above it. The bottom-right vertex has a red circle labeled $\bar{\tau}$ above it.

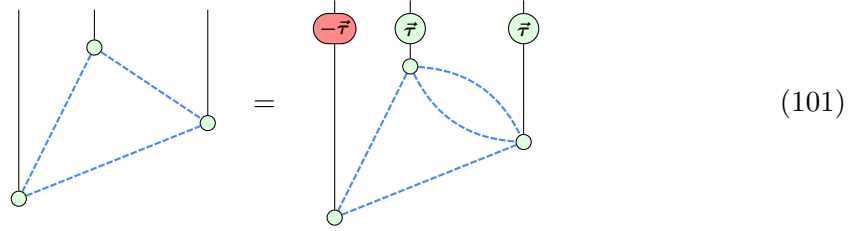
Diagram 9: A triangle graph with three vertices. The bottom-left vertex is connected to the other two by solid black lines. The top vertex is connected to the other two by solid black lines. The bottom-right vertex is connected to the other two by solid black lines. Each edge has a yellow box labeled 1. The top-left vertex has a green circle labeled $\bar{\tau}$ above it. The top-right vertex has a green circle labeled $\bar{\tau}$ above it. The bottom-right vertex has a red circle labeled $\bar{\tau}$ above it.



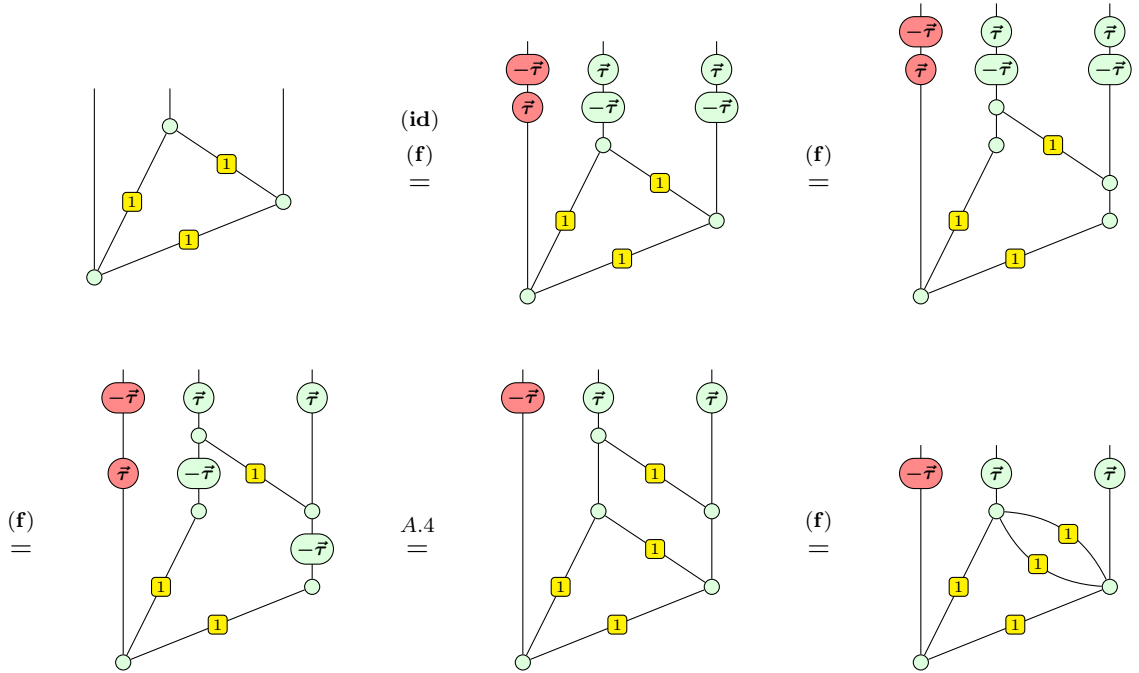
□

And finally this gives our first local complementation result: the complementation by 1 for the case K_3 . Recall that for an edge ij of weight w_{ij} , the local complementation by 1 at x increases w_{ij} by $w_{ix}w_{jx}$:

Lemma A.5.



Proof.



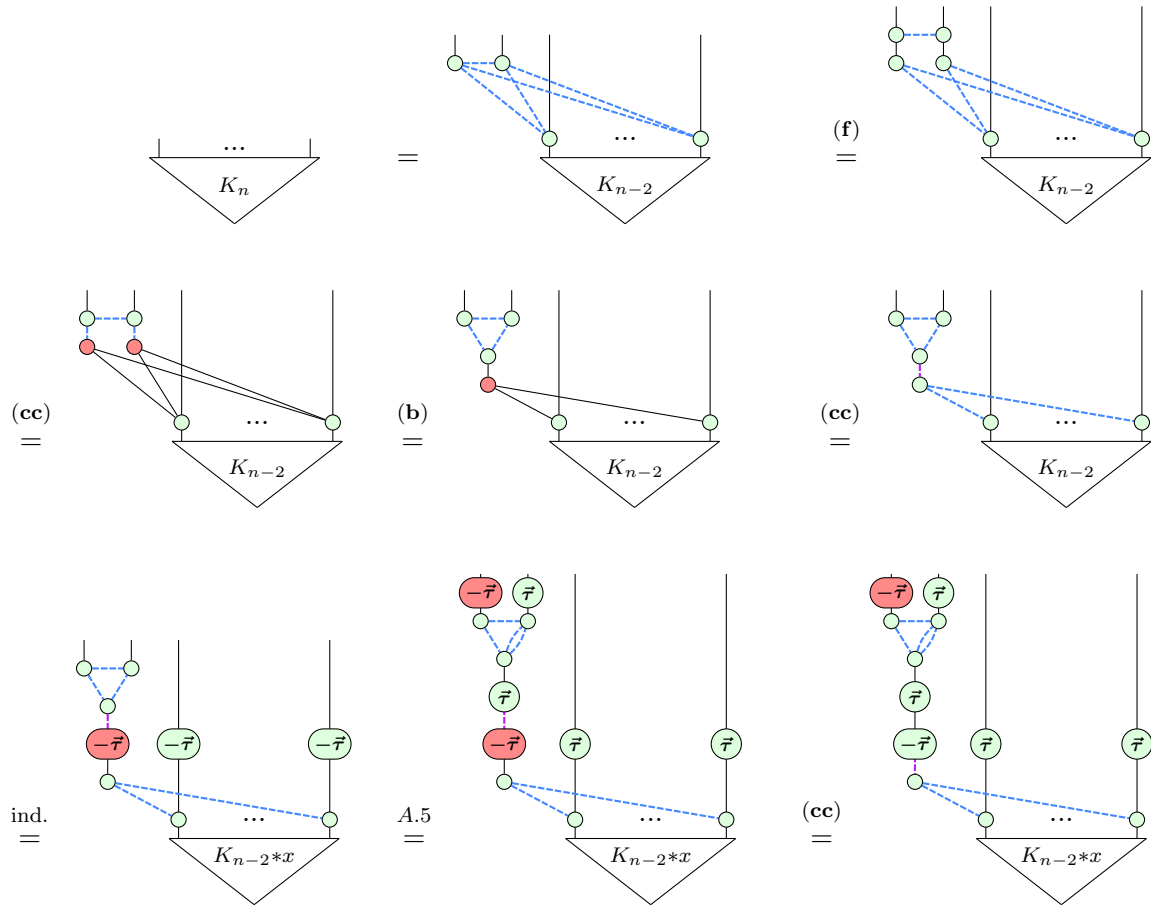
□

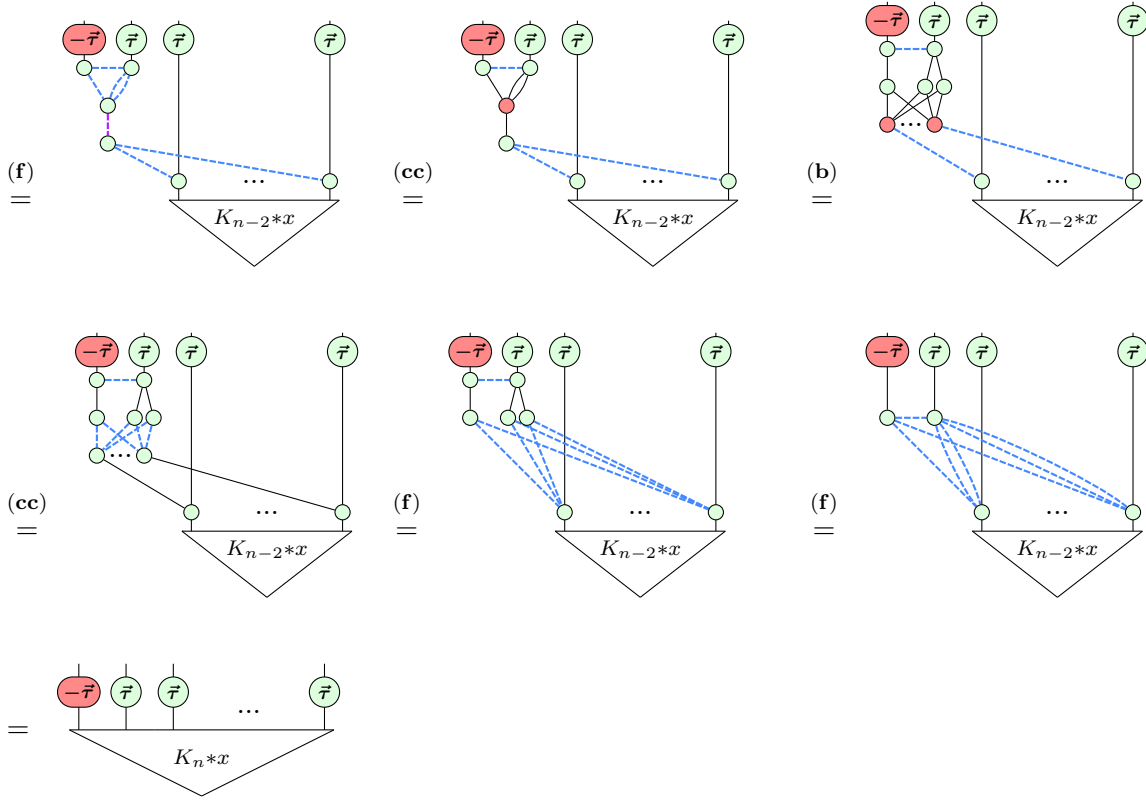
The local complementation by 1 for the smaller complete graphs K_1 and K_2 are easy to prove. For $n > 3$ we proceed inductively:

Lemma A.6.

$$\begin{array}{c} \dots \\ \triangle \\ K_n \end{array} = \begin{array}{c} \begin{array}{c} \textcolor{red}{-\bar{\tau}} \quad \bar{\tau} \quad \bar{\tau} \quad \dots \quad \bar{\tau} \end{array} \\ \triangle \\ K_{n-2} * x \end{array} \quad (102)$$

Proof.

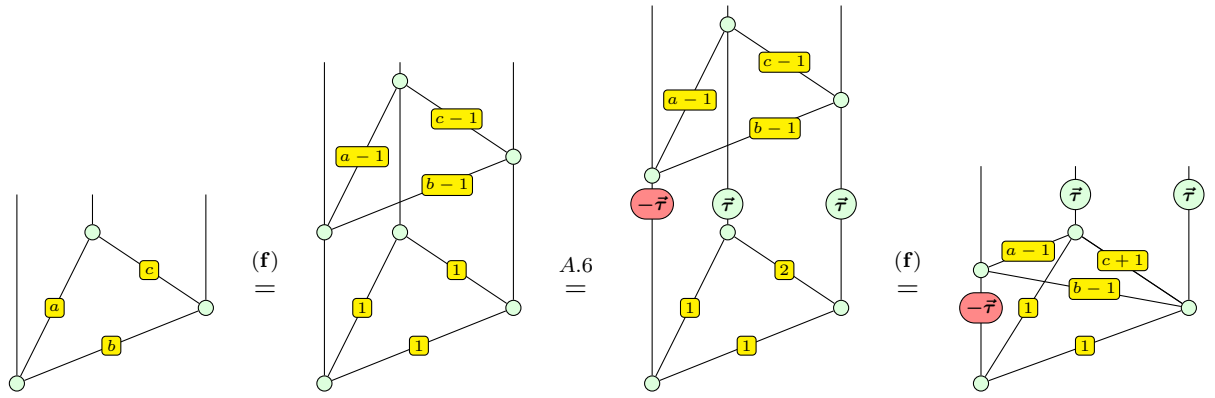




□

For local complementation by $a > 1$ on the complete graph K_n , we just note that this means a local complementations by 1: $K_n * a x = (((K_n * 1 x) * 1 x) \dots) * 1 x$. Finally, only the non-complete case remains. The proofs for the qubit and qutrit case in Refs. [21] and [27] respectively state that this can be deduced from the complete case. Roughly, they say that any graph state G on n vertices can be written as a complete graph K_n plus some additional weighted edges. By spider fusion, we can temporarily move these extra edges, use our result for the complete graph K_n , then use fusion and other rules to put the extra edges back in place. However, doing this for the qudit case has proved far less trivial than anticipated. Therefore, sadly, we are forced to leave the full result as a conjecture. We

give the first few steps of the proposed proof strategy below:



B Eliminating $\langle \pm 1, x \rangle$ -Spiders

Here we prove the elimination theorems for spiders with phases $\langle \pm 1, x \rangle$, assuming the local complementation result from the previous section holds. Again, it is perhaps easiest to first prove one of the qutrit cases as an example, before moving on to the general case. Let's write our qutrit spider phases in $\langle -, - \rangle$ notation:

$$\mathcal{M} = \left\{ \begin{array}{c} \cdots \\ \textcircled{\frac{0}{0}} \\ \cdots \end{array}, \begin{array}{c} \cdots \\ \textcircled{\frac{1}{2}} \\ \cdots \end{array}, \begin{array}{c} \cdots \\ \textcircled{\frac{2}{1}} \\ \cdots \end{array} \right\} = \left\{ \begin{array}{c} \cdots \\ \langle 0, 0 \rangle \\ \cdots \end{array}, \begin{array}{c} \cdots \\ \langle 0, -2 \rangle \\ \cdots \end{array}, \begin{array}{c} \cdots \\ \langle 0, 2 \rangle \\ \cdots \end{array} \right\} \quad (103)$$

$$\mathcal{N} = \left\{ \begin{array}{c} \cdots \\ \textcircled{\frac{0}{1}} \\ \cdots \end{array}, \begin{array}{c} \cdots \\ \textcircled{\frac{1}{0}} \\ \cdots \end{array}, \begin{array}{c} \cdots \\ \textcircled{\frac{0}{2}} \\ \cdots \end{array}, \begin{array}{c} \cdots \\ \textcircled{\frac{2}{0}} \\ \cdots \end{array} \right\} = \left\{ \begin{array}{c} \cdots \\ \langle -1, 2 \rangle \\ \cdots \end{array}, \begin{array}{c} \cdots \\ \langle -1, 0 \rangle \\ \cdots \end{array}, \begin{array}{c} \cdots \\ \langle 1, -2 \rangle \\ \cdots \end{array}, \begin{array}{c} \cdots \\ \langle 1, 0 \rangle \\ \cdots \end{array} \right\} \quad (104)$$

$$\mathcal{P} = \left\{ \begin{array}{c} \cdots \\ \textcircled{\frac{1}{1}} \\ \cdots \end{array}, \begin{array}{c} \cdots \\ \textcircled{\frac{2}{2}} \\ \cdots \end{array} \right\} = \left\{ \begin{array}{c} \cdots \\ \langle 1, 2 \rangle \\ \cdots \end{array}, \begin{array}{c} \cdots \\ \langle -1, -2 \rangle \\ \cdots \end{array} \right\} \quad (105)$$

Or a bit more visually, using the lattice on which we drew \mathcal{S}_d earlier:

$$\begin{array}{c} x \\ \uparrow \\ 5 \\ 4 \begin{array}{ccc} \textcircled{\frac{1}{2}} & \textcircled{\frac{0}{2}} & \textcircled{\frac{2}{2}} \end{array} \\ 3 \\ 2 \begin{array}{ccc} \textcircled{\frac{2}{1}} & \textcircled{\frac{1}{1}} & \textcircled{\frac{0}{1}} \end{array} \\ 1 \\ 0 \begin{array}{ccc} \textcircled{\frac{0}{0}} & \textcircled{\frac{2}{0}} & \textcircled{\frac{1}{0}} \end{array} \rightarrow k \\ 0 \quad 1 \quad 2 \end{array} \quad (106)$$

So the $\langle \pm 1, x \rangle$ -spiders are exactly the \mathcal{P} - and \mathcal{N} -spiders. Let's prove the case for an \mathcal{N} -spider with phase $\textcircled{\frac{0}{1}}$. Lemma 5.16 tells us that:

$$\textcircled{\frac{0}{1}} = \textcircled{\frac{2}{0}} \quad (107)$$

And one extra result we'll need, proved for qudits, is:

$$\begin{array}{c} \cdots \\ \textcircled{\tilde{\alpha}} \\ | \\ \textcircled{x\tilde{\kappa}} \end{array} \stackrel{(f)}{=} \begin{array}{c} \cdots \\ \textcircled{\tilde{\alpha}} \\ | \\ \textcircled{x\tilde{\kappa}} \\ \bullet \end{array} \stackrel{(\kappa)}{=} \begin{array}{c} \cdots \\ | \quad | \\ \textcircled{x\tilde{\kappa}} \quad \textcircled{x\tilde{\kappa}} \\ | \\ \textcircled{\tilde{\alpha}^x} \\ \bullet \end{array} \stackrel{(z)}{=} \begin{array}{c} \cdots \\ | \quad | \\ \textcircled{x\tilde{\kappa}} \quad \textcircled{x\tilde{\kappa}} \\ \bullet \quad \bullet \end{array} \stackrel{(f)}{=} \begin{array}{c} \cdots \\ | \quad | \\ \textcircled{x\tilde{\kappa}} \quad \textcircled{x\tilde{\kappa}} \end{array} \quad (108)$$

The theorem for this qutrit case is then:

$$\begin{array}{c}
 \begin{array}{|c|} \hline \dots \\ \hline \alpha_k \\ \hline \beta_k \\ \hline \end{array}
 \begin{array}{|c|} \hline \dots \\ \hline \gamma_k \\ \hline \delta_k \\ \hline \end{array}
 \end{array}
 \begin{array}{c}
 k \in [K_1] \quad k \in [K_2]
 \end{array}
 \begin{array}{c}
 \begin{array}{|c|} \hline 0 \\ \hline 1 \\ \hline \end{array}
 \end{array}
 =
 \begin{array}{c}
 \begin{array}{|c|} \hline \dots \\ \hline \alpha_k + 1 + 2 \\ \hline \beta_k + 1 + 1 \\ \hline \end{array}
 \begin{array}{|c|} \hline \dots \\ \hline \gamma_k + 1 + 1 \\ \hline \delta_k + 1 + 2 \\ \hline \end{array}
 \end{array}
 \begin{array}{c}
 k \in [K_1] \quad k \in [K_2]
 \end{array}
 \quad (109)$$

Proof.

$$\begin{array}{c}
 \begin{array}{|c|} \hline \dots \\ \hline \alpha_k \\ \hline \beta_k \\ \hline \end{array}
 \begin{array}{|c|} \hline \dots \\ \hline \gamma_k \\ \hline \delta_k \\ \hline \end{array}
 \end{array}
 \begin{array}{c}
 k \in [K_1] \quad k \in [K_2]
 \end{array}
 \begin{array}{c}
 \begin{array}{|c|} \hline 0 \\ \hline 1 \\ \hline \end{array}
 \end{array}
 \stackrel{(f)}{=}
 \begin{array}{c}
 \begin{array}{|c|} \hline \dots \\ \hline \alpha_k \\ \hline \beta_k \\ \hline \end{array}
 \begin{array}{|c|} \hline \dots \\ \hline \gamma_k \\ \hline \delta_k \\ \hline \end{array}
 \end{array}
 \begin{array}{c}
 k \in [K_1] \quad k \in [K_2]
 \end{array}
 \begin{array}{c}
 \begin{array}{|c|} \hline 0 \\ \hline 1 \\ \hline \end{array}
 \end{array}
 \stackrel{4.5}{=}
 \begin{array}{c}
 \begin{array}{|c|} \hline \dots \\ \hline \alpha_k \\ \hline \beta_k \\ \hline \end{array}
 \begin{array}{|c|} \hline \dots \\ \hline \gamma_k \\ \hline \delta_k \\ \hline \end{array}
 \end{array}
 \begin{array}{c}
 k \in [K_1] \quad k \in [K_2]
 \end{array}
 \begin{array}{c}
 \begin{array}{|c|} \hline 0 \\ \hline 1 \\ \hline \end{array}
 \end{array}
 \stackrel{(23)}{=}
 \begin{array}{c}
 \begin{array}{|c|} \hline \dots \\ \hline \alpha_k \\ \hline \beta_k \\ \hline \end{array}
 \begin{array}{|c|} \hline \dots \\ \hline \gamma_k \\ \hline \delta_k \\ \hline \end{array}
 \end{array}
 \begin{array}{c}
 k \in [K_1] \quad k \in [K_2]
 \end{array}
 \begin{array}{c}
 \begin{array}{|c|} \hline 1 \\ \hline 1 \\ \hline \end{array}
 \end{array}
 \stackrel{(25)}{=}
 \begin{array}{c}
 \begin{array}{|c|} \hline \dots \\ \hline \alpha_k \\ \hline \beta_k \\ \hline \end{array}
 \begin{array}{|c|} \hline \dots \\ \hline \gamma_k \\ \hline \delta_k \\ \hline \end{array}
 \end{array}
 \begin{array}{c}
 k \in [K_1] \quad k \in [K_2]
 \end{array}
 \begin{array}{c}
 \begin{array}{|c|} \hline 1 \\ \hline 1 \\ \hline \end{array}
 \end{array}
 \stackrel{(H)}{=}
 \begin{array}{c}
 \begin{array}{|c|} \hline \dots \\ \hline \alpha_k \\ \hline \beta_k \\ \hline \end{array}
 \begin{array}{|c|} \hline \dots \\ \hline \gamma_k \\ \hline \delta_k \\ \hline \end{array}
 \end{array}
 \begin{array}{c}
 k \in [K_1] \quad k \in [K_2]
 \end{array}
 \begin{array}{c}
 \begin{array}{|c|} \hline 1 \\ \hline 1 \\ \hline \end{array}
 \end{array}
 \end{array}$$

$$\begin{aligned}
(107) \quad & \begin{array}{c} \text{Diagram 1: Two boxes labeled } k \in [K_1] \text{ and } k \in [K_2]. \text{ Each box contains a spider with legs } \alpha_k, \beta_k, \gamma_k, \delta_k \text{ and a node } \frac{1}{1}. \text{ The boxes are connected by a blue dashed line. Below the boxes is a red node } \frac{-1}{-1} \text{ and a red node } \frac{2}{0}. \end{array} \\
& \stackrel{(f)}{=} \begin{array}{c} \text{Diagram 2: Similar to Diagram 1, but the red node is } \frac{1}{2}. \end{array} \\
& \stackrel{(108)}{=} \begin{array}{c} \text{Diagram 3: Similar to Diagram 1, but the red node is } \frac{1}{2}. \end{array} \\
& \stackrel{(cc)}{=} \begin{array}{c} \text{Diagram 4: Similar to Diagram 1, but the red node is } \frac{1}{2}. \end{array} \\
& \stackrel{(f)}{=} \begin{array}{c} \text{Diagram 5: Similar to Diagram 1, but the red node is } \frac{1}{2}. \end{array}
\end{aligned}$$

□

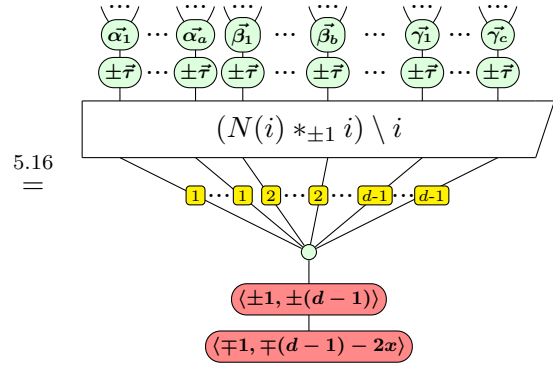
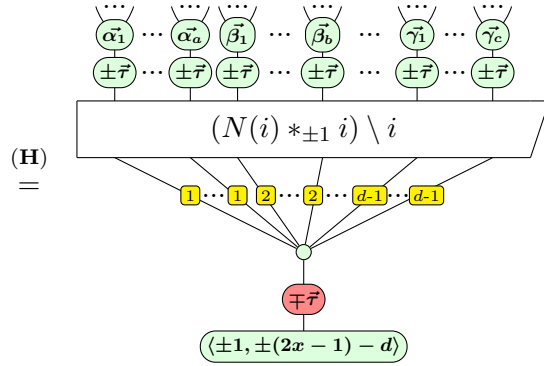
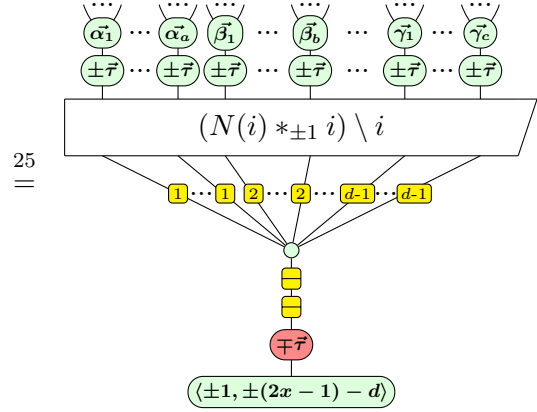
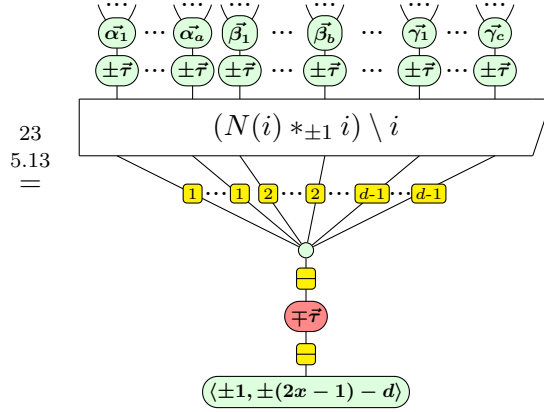
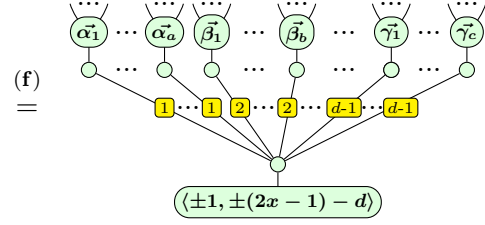
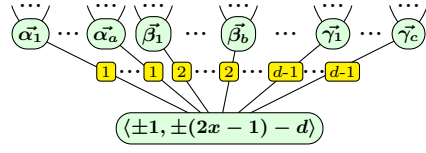
We now conditionally prove the theorem in its full generality. The following equation is required:

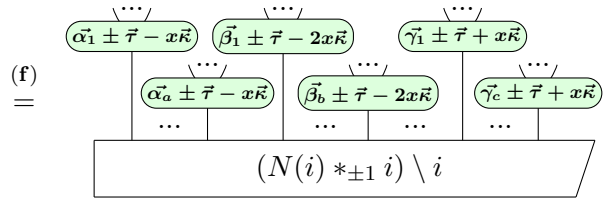
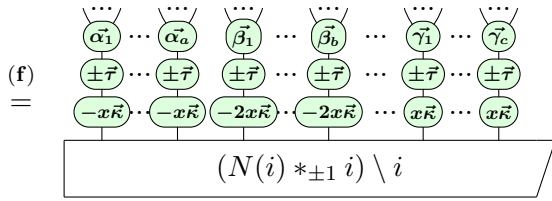
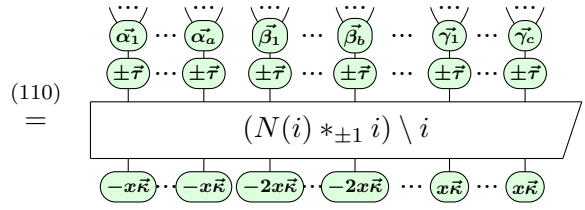
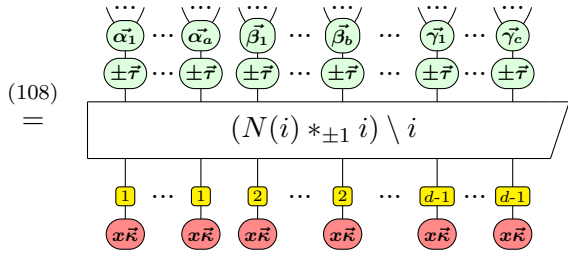
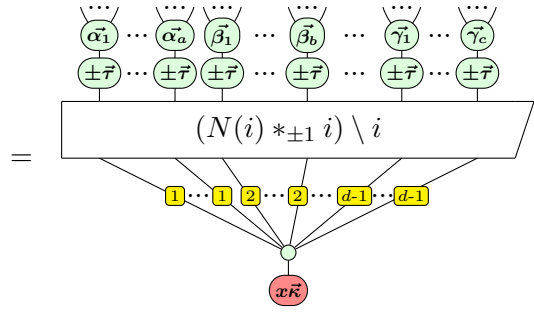
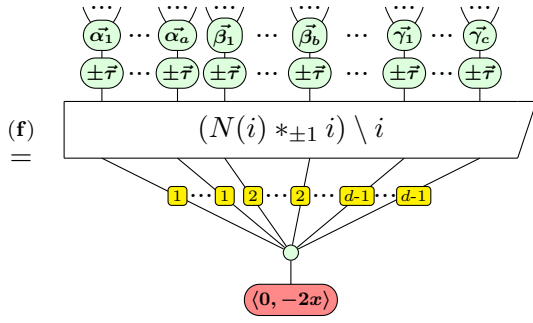
$$\begin{array}{c} \text{Diagram 1: A node } n \text{ with a red node } x\vec{\kappa} \text{ below it.} \end{array} \stackrel{(108)}{=} \begin{array}{c} \text{Diagram 2: A node } n \text{ with two red nodes } x\vec{\kappa} \text{ below it.} \end{array} \stackrel{(cc)}{=} \begin{array}{c} \text{Diagram 3: A node } n \text{ with two green nodes } -x\vec{\kappa} \text{ below it.} \end{array} \stackrel{(f)}{=} \begin{array}{c} \text{Diagram 4: A node } n \text{ with a green node } -nx\vec{\kappa} \text{ below it.} \end{array} \quad (110)$$

Theorem B.1. / Theorem 5.17 (Conditional). Suppose we have a graph-like diagram containing a node i with phase $\langle \pm 1, y \rangle \in \mathcal{S}_d$. Let $N(i)$ denote the neighbours of i (spiders connected to i by a Hadamard edge of non-zero weight). Then the following equation holds:

$$\begin{array}{c} \text{Diagram 1: A spider with legs } \alpha_1, \alpha_a, \beta_1, \beta_b, \gamma_1, \gamma_c \text{ and a node } (\pm 1, \pm(2x-1)-d). \end{array} \stackrel{(f)}{=} \begin{array}{c} \text{Diagram 2: A spider with legs } \alpha_1 \pm \vec{\tau} - x\vec{\kappa}, \beta_1 \pm \vec{\tau} - 2x\vec{\kappa}, \gamma_1 \pm \vec{\tau} + x\vec{\kappa} \text{ and a node } (N(i) *_{\pm 1} i) \setminus i. \end{array}$$

Proof.





□

C Pivoting in the Qutrit ZX-Calculus

Here we prove the qudit pivot equality from Theorem 5.14, assuming the qudit local complementation result holds. There is quite a lot going on in the diagrams in the proof, so we again start with the specific qutrit case from Theorem 4.7, which is a little more digestible. We recall the following: since **(E)** holds under taking adjoints, and swapping the roles of red and green, we have:

$$\begin{array}{c} | \\ \text{1} \end{array} = \begin{array}{c} \text{green} \frac{2}{2} \\ \text{red} \frac{2}{2} \\ \text{green} \frac{2}{2} \end{array} = \begin{array}{c} \text{red} \frac{2}{2} \\ \text{green} \frac{2}{2} \\ \text{red} \frac{2}{2} \end{array}, \quad \begin{array}{c} | \\ \text{2} \end{array} = \begin{array}{c} \text{red} \frac{1}{1} \\ \text{green} \frac{1}{1} \\ \text{red} \frac{1}{1} \end{array} = \begin{array}{c} \text{green} \frac{1}{1} \\ \text{red} \frac{1}{1} \\ \text{green} \frac{1}{1} \end{array} \quad (111)$$

Theorem C.1. / Theorem 4.7. Given $a \in \mathbb{Z}_3$ and a graph state (G, W) containing connected nodes i and j , define:

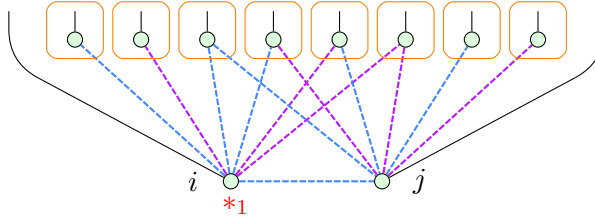
$$N_{=}(i, j) := \{x \in N(i) \cap N(j) \mid w_{x,i} = w_{x,j}\} \quad (112)$$

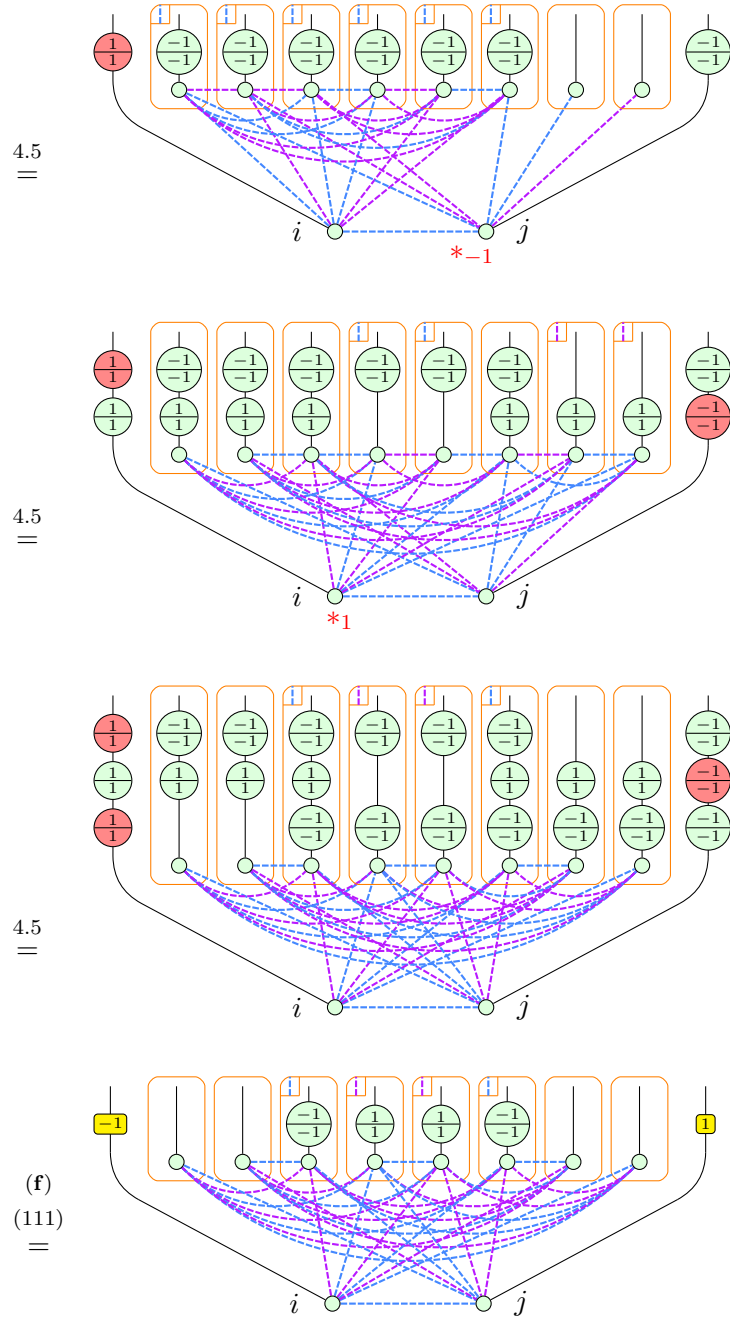
$$N_{\neq}(i, j) := \{x \in N(i) \cap N(j) \mid w_{x,i} \neq w_{x,j}\} \quad (113)$$

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

$$\begin{array}{c} \begin{array}{ccccccc} i & j & N_{\neq}(i,j) & N_{=}(i,j) & & & \\ \text{---} & \text{---} & \text{---} & \text{---} & \text{---} & \text{---} & \text{---} \\ & & \dots & \dots & & & \dots \end{array} \\ \text{---} \\ G \end{array} = \begin{array}{c} \begin{array}{ccccccc} i & j & N_{\neq}(i,j) & N_{=}(i,j) & & & \\ \text{---} & \text{---} & \text{---} & \text{---} & \text{---} & \text{---} & \text{---} \\ & & \dots & \dots & & & \dots \end{array} \\ \text{---} \\ G \wedge_a ij \end{array}$$

Proof. There are four cases $(a, w_{i,j} \in \{1, 2\})$, which split into two pairs of symmetric cases: $a = w_{i,j}$ and $a \neq w_{i,j}$. We show just one case – the 1-pivot along ij of weight 1 – the remaining cases being analogous. We again employ the !-notation. An asterisk $*_a$ next to a spider means that at the next step an a -local complementation will be performed at this spider.





□

For the qudit case, we abstract away the details about the changes to edge weights. We

define the following shorthands:

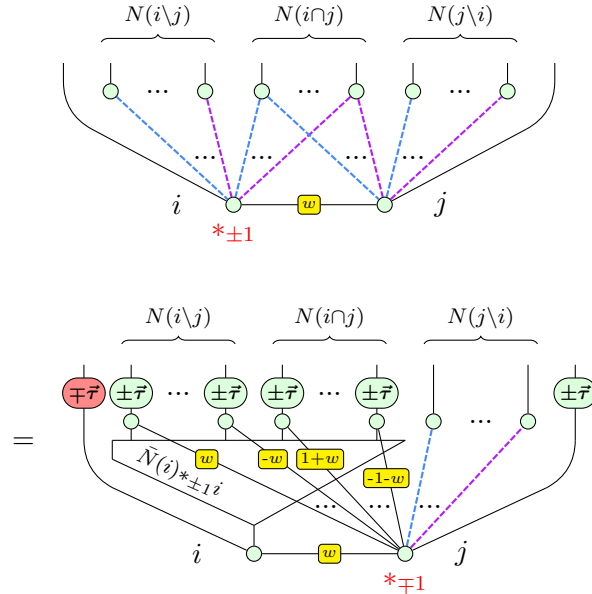
$$N(i \cap j) = N(i) \cap N(j), \quad N(i \cup j) = N(i) \cup N(j), \quad N(i \setminus j) = N(i) \setminus N(j) \quad (114)$$

If we write \bar{N} instead of N this means we include nodes i and j too, e.g. $\bar{N}(i \cap j) = \{i, j\} \cup N(i \cap j)$. Similarly we write $\bar{N}(i) = \{i\} \cup N(i)$.

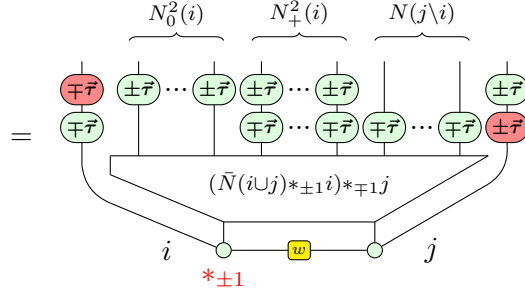
Theorem C.2. / Theorem 5.14 (Conditional). Suppose we have a graph state (G, W) containing connected nodes i and j . Let $\{N_{\pm 2}(i \cup j), N_{\pm 1}(i \cup j), N_{\mp 1}(i \cup j)\}$ be disjoint subsets of $N(i \cup j)$; the diagrams below shed light on their definitions, so we define them at the end of this proof. Then the following equation relates G and its proper ± 1 -pivot along ij :

$$\begin{array}{c} \text{...} \\ \diagdown \quad \diagup \\ G \end{array} = \begin{array}{c} \begin{array}{ccccccc} i & j & N_{\pm 2}(i \cup j) & N_{\pm 1}(i \cup j) & N_{\mp 1}(i \cup j) & & \\ \hline \boxed{\mp 1} & \boxed{\pm 1} & \boxed{\pm 2\tau} \dots \boxed{\pm 2\tau} & \boxed{\pm \tau} \dots \boxed{\pm \tau} & \boxed{\mp \tau} \dots \boxed{\mp \tau} & & \end{array} \\ \diagdown \quad \diagup \\ G \wedge_{\pm 1} ij \end{array} \quad (115)$$

Proof. We need only consider the nodes in $\bar{N}(i \cup j)$. Let w_{xy}^k denote the weight of the edge between nodes x and y after k local complementations. First we locally complement by ± 1 at i . For any node x , this leaves the weight w_{ix}^1 of its edge to i unchanged, but w_{jx}^1 is $w_{jx} + w_{ix}w$. We abstract away changes to weights of edges between neighbours of i . All neighbours of i pick up a green $\pm \tau$ gate.

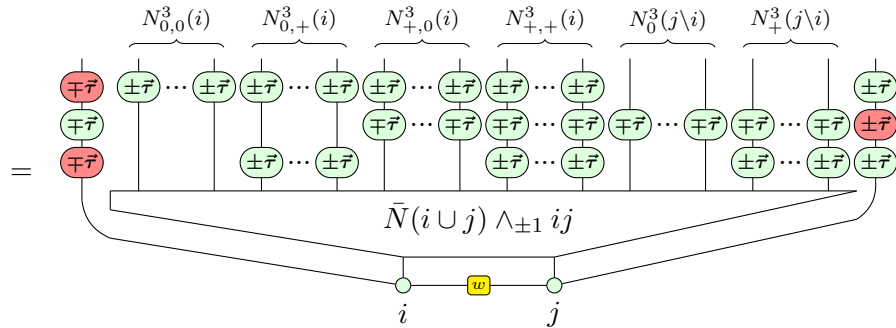


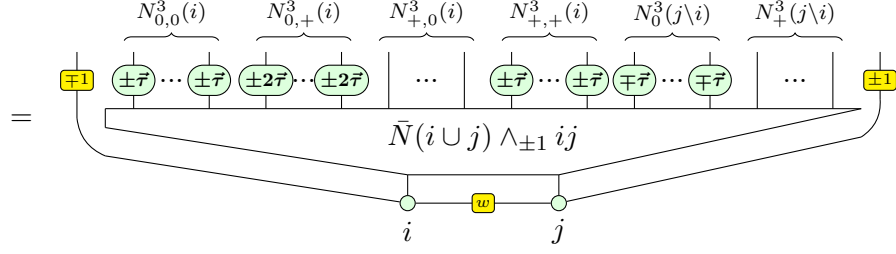
Next we locally complement by ∓ 1 at j . This leaves weights w_{jx}^2 unchanged, but w_{ix}^2 is $w_{ix}^1 + w_{jx}^1 w$. We abstract away other edge weight changes. Nodes $x \in N(i \cup j)$ with $w_{jx}^1 \neq 0$ pick up a green $\mp \tau$ gate. So let $N_0^2(i) := \{x \in N(i) : w_{jx}^1 = 0\}$, and similarly let $N_+^2(i) := \{x \in N(i) : w_{jx}^1 \neq 0\}$.



Finally we again locally complement by ± 1 at i . This leaves weights w_{ix}^3 unchanged, but w_{jx}^3 is $w_{jx}^2 + w_{ix}^2 w$. Nodes $x \in N(i \cup j)$ with $w_{ix}^2 \neq 0$ pick up a green $\pm \tau$ gate. So let:

$$\begin{aligned} N_{0,0}^3(i) &:= \{x \in N_0^2(i) : w_{ix}^2 = 0\} \\ N_{0,+}^3(i) &:= \{x \in N_0^2(i) : w_{ix}^2 \neq 0\} \\ N_{+,0}^3(i) &:= \{x \in N_+^2(i) : w_{ix}^2 = 0\} \\ N_{+,+}^3(i) &:= \{x \in N_+^2(i) : w_{ix}^2 \neq 0\} \\ N_0^3(j \setminus i) &:= \{x \in N(j \setminus i) : w_{ix}^2 = 0\} \\ N_+^3(j \setminus i) &:= \{x \in N(j \setminus i) : w_{ix}^2 \neq 0\} \end{aligned}$$



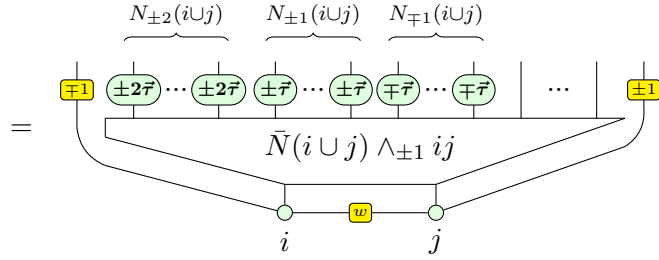


Now we group together these sets of nodes under the following new names:

$$N_{\pm 2}(i \cup j) := N_{0,+}^3(i)$$

$$N_{\pm 1}(i \cup j) := N_{0,0}^3(i) \cup N_{+,+}^3(i)$$

$$N_{\mp 1}(i \cup j) := N_0^3(j \setminus i)$$

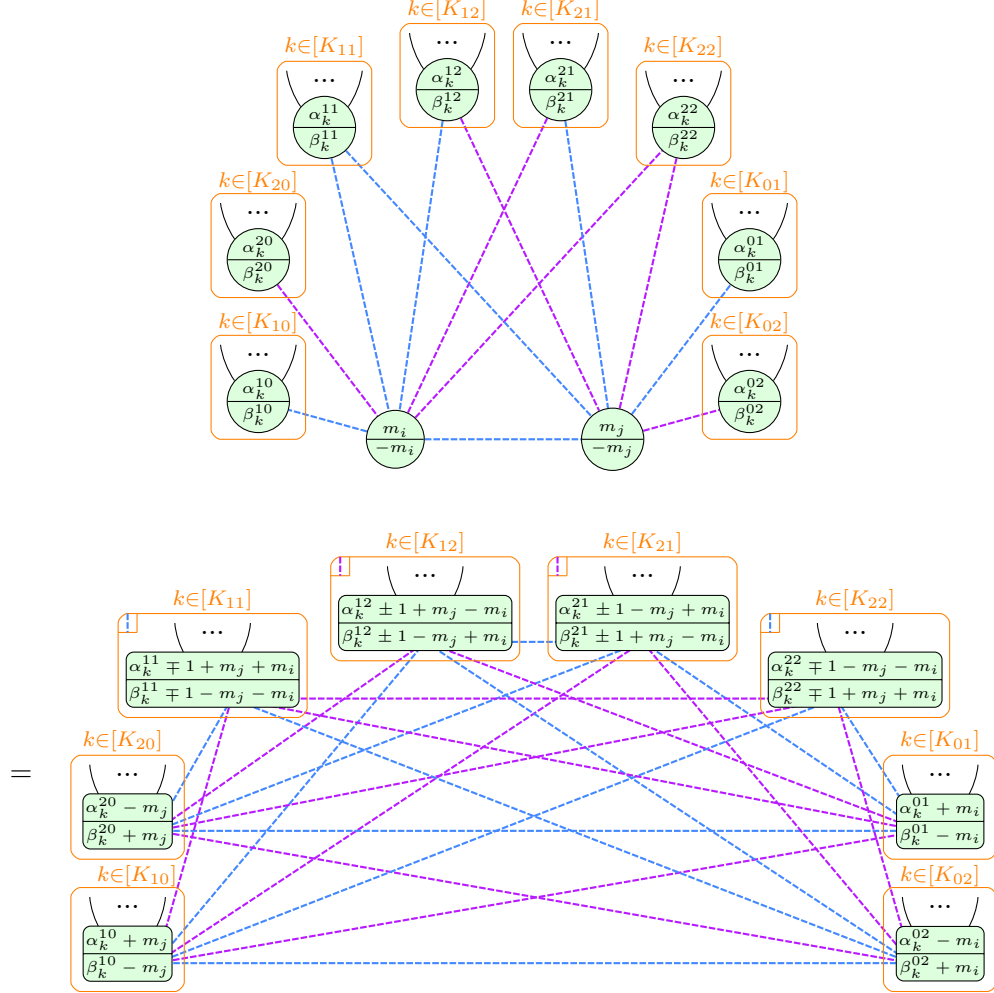


□

D Eliminating $\langle 0, 2x \rangle$ -Spiders

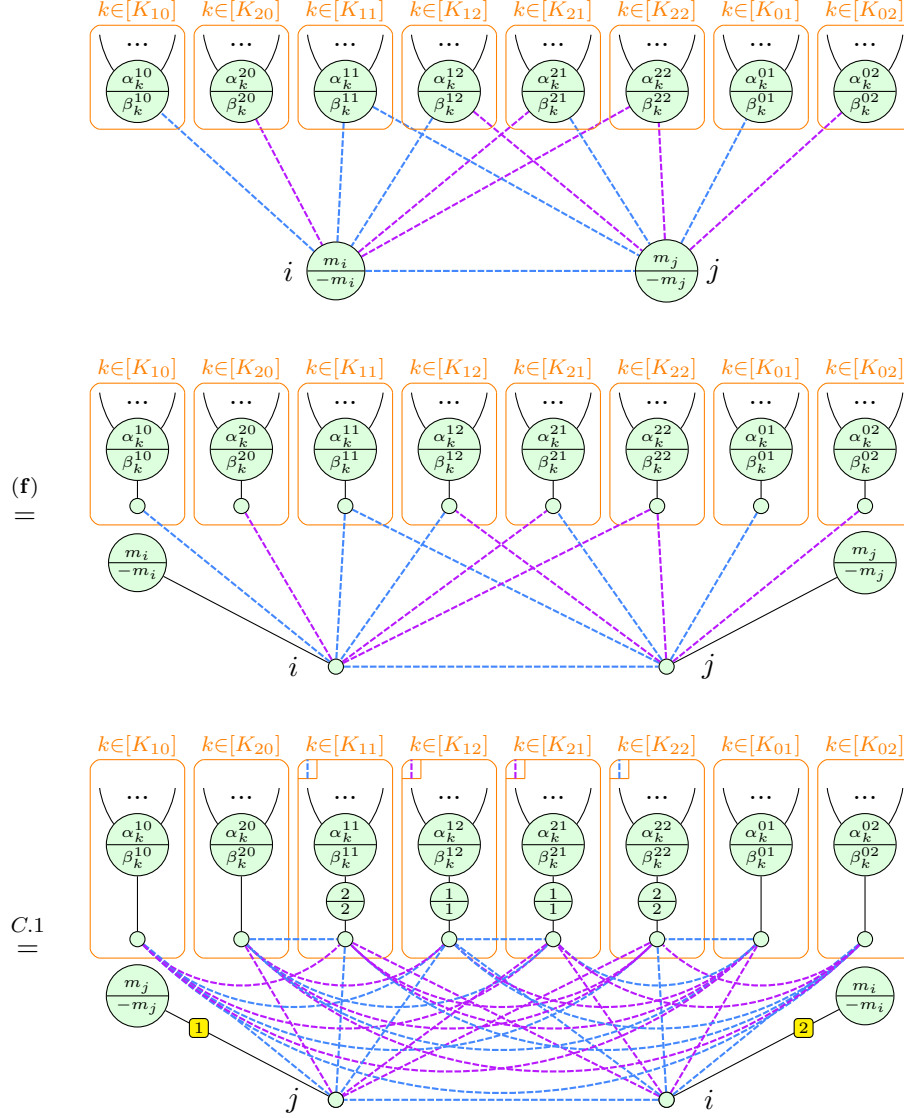
Unsurprisingly, we again warm up by looking at the simpler qutrit case.

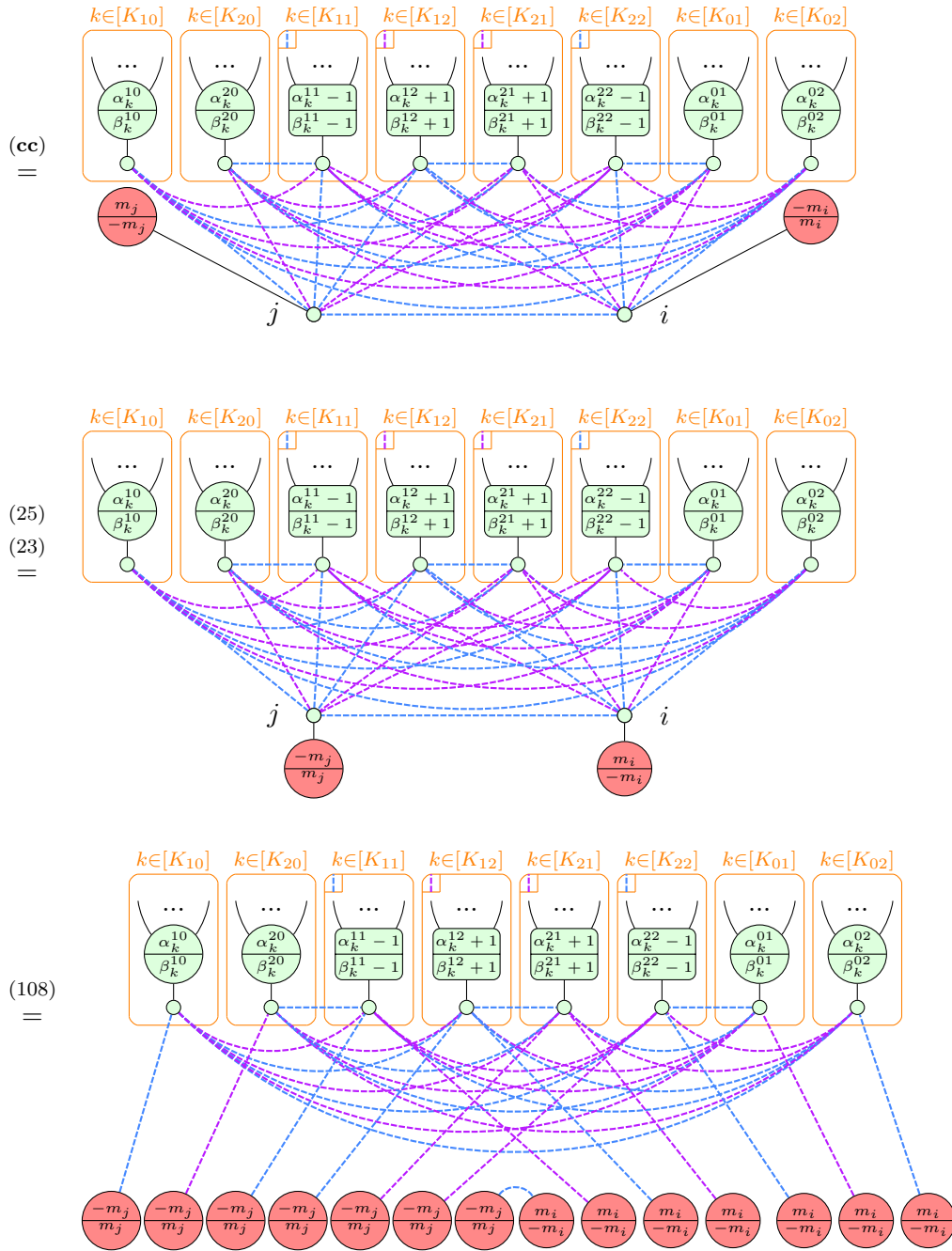
Theorem D.1. / Theorem 4.10. Given any graph-like ZX-diagram containing two interior \mathcal{M} -spiders i and j connected by edge ij of weight $w_{i,j} =: w \in \{1, 2\}$, suppose we perform a proper $\pm w$ -pivot along ij (both choices give the same result). For the case $w = 1$, the new ZX-diagram is related to the old one by the following equality:

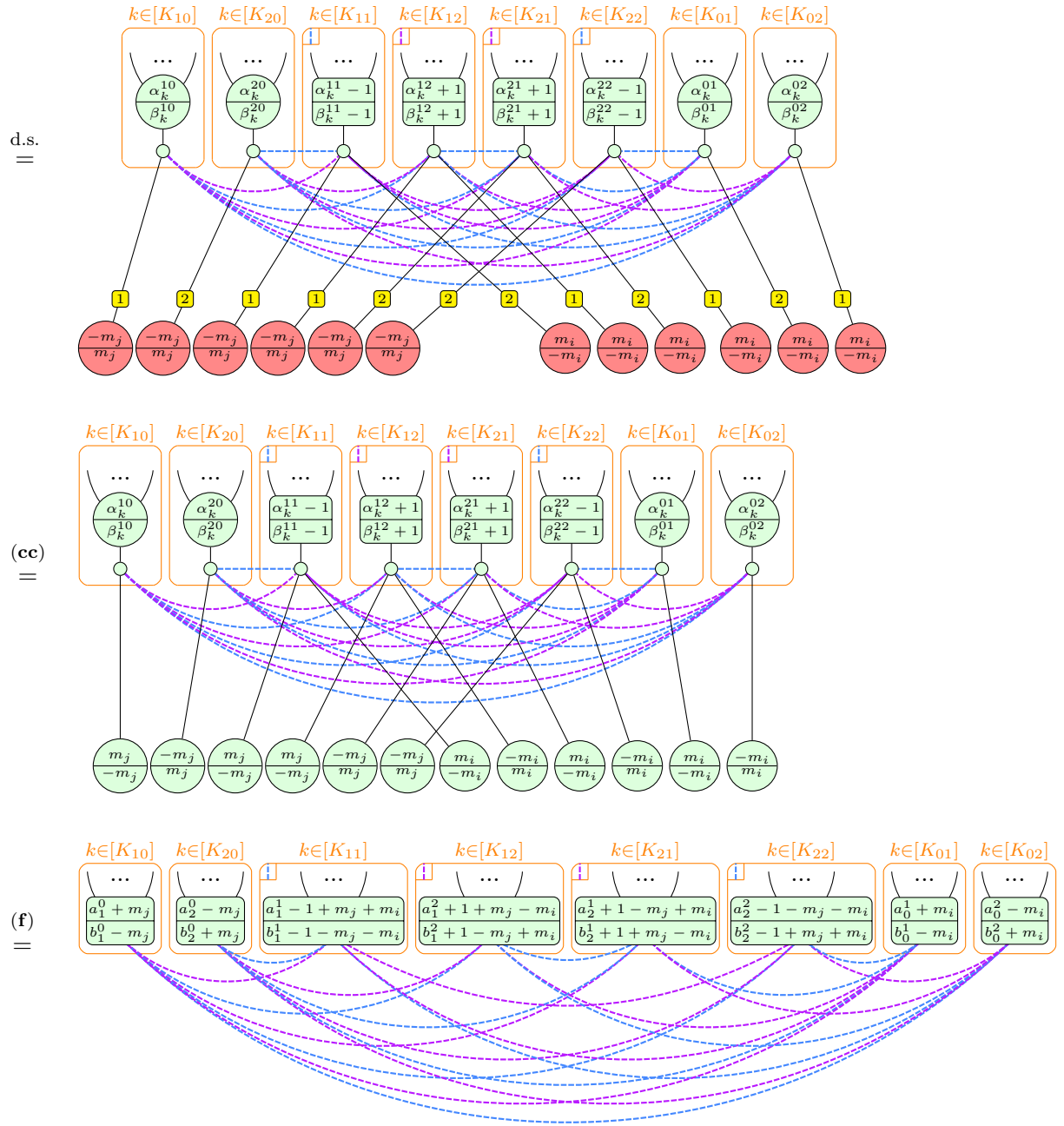


The case $w = 2$ differs only as follows: in the first diagram (the left hand side of the equation), the edge ij is purple (by definition), while in the lower diagram (the right hand side of the equation), a $\pm 2 = \mp 1$ will replace all occurrences of ± 1 , and the roles of purple and blue will be swapped throughout.

Proof. We show the case where $w_{ij} =: w = 1$, with the case $w = 2$ being completely analogous. We can choose either a proper 1-pivot or a proper 2-pivot; both give the same result. Here we only show the former:







□

Theorem D.2. / Theorem 5.18 (Conditional). Suppose we have a graph-like diagram containing connected nodes i and j with phases $\langle 0, 2x \rangle$ and $\langle 0, 2y \rangle \in \mathcal{S}_d$ respectively. Let $[a, b]$ denote the set of all nodes connected to i by an a -Hadamard edge and to j by a b -Hadamard edge. We subscript phases of such elements with a, b . In Theorem 5.14 we defined disjoint sets $\{N_{\pm 2}(i \cup j), N_{\pm 1}(i \cup j), N_{\mp 1}(i \cup j)\}$. Whether or not a node is a member of one of these sets is determined entirely by a and b . So the following is well-defined:

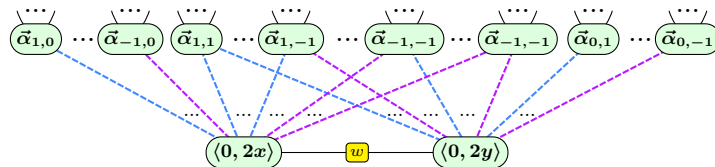
$$z_{a,b} = \begin{cases} z & \text{if } [a,b] \subseteq N_z(i \cup j) \\ 0 & \text{else} \end{cases} \quad (116)$$

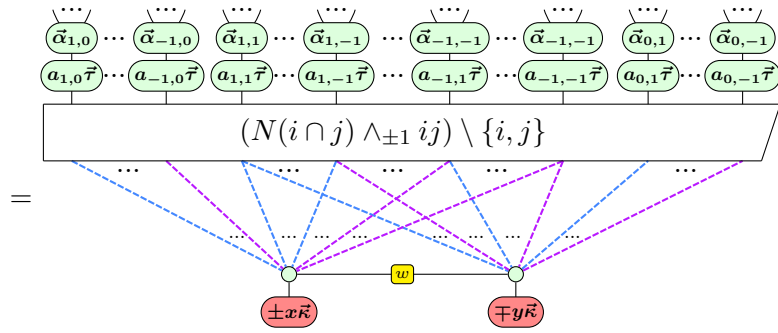
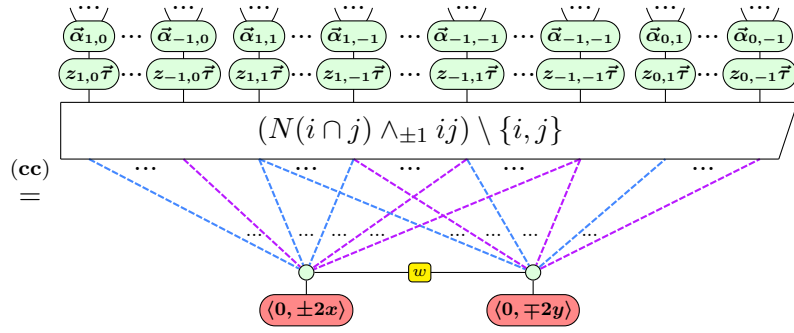
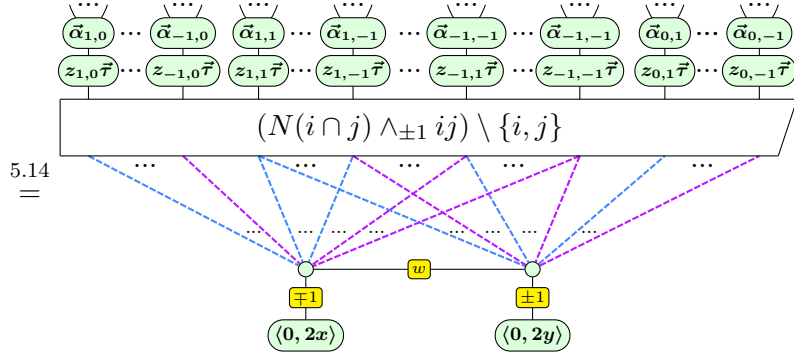
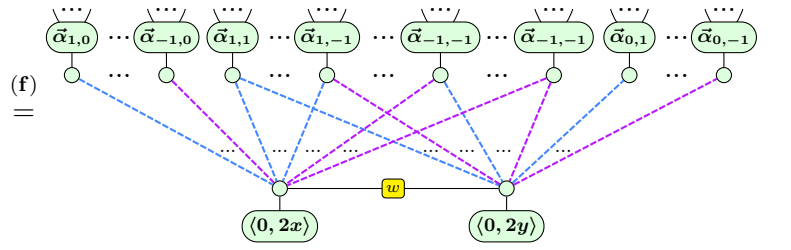
$$\vec{\beta}_{a,b} = (\pm ax \mp by)\vec{\kappa} + z_{a,b}\vec{\tau} \quad (117)$$

Then the following equation holds:

[illegible]

Proof.





(108)

=

(109)

=

(110)

=

(f)

$$\begin{array}{c}
\begin{array}{cccc}
\begin{array}{c} \vdots \\ \alpha_{1,0} + \beta_{1,0} \end{array} &
\begin{array}{c} \vdots \\ \alpha_{1,1} + \beta_{1,1} \end{array} &
\begin{array}{c} \vdots \\ \alpha_{-1,1} + \beta_{-1,1} \end{array} &
\begin{array}{c} \vdots \\ \alpha_{0,1} + \beta_{0,1} \end{array} \\
\vdots & \vdots & \vdots & \vdots \\
\begin{array}{c} \vdots \\ \alpha_{-1,0} + \beta_{-1,0} \end{array} &
\begin{array}{c} \vdots \\ \alpha_{1,-1} + \beta_{1,-1} \end{array} &
\begin{array}{c} \vdots \\ \alpha_{-1,-1} + \beta_{-1,-1} \end{array} &
\begin{array}{c} \vdots \\ \alpha_{0,-1} + \beta_{0,-1} \end{array}
\end{array} \\
= & & & \\
& \underbrace{\hspace{15em}}_{(N(i \cap j) \wedge_{\pm 1} ij) \setminus \{i, j\}}
\end{array}$$

□

E \pm -Boxes in the ZX-Calculus

We close by proving that the \pm -boxes in (94) correspond to stabilizer ZX-diagrams, while the qutrit X -box in (95) does not.

Proposition E.1. The following equalities hold up to a scalar under the standard interpretation:

$$\left[\begin{array}{c} \boxed{\pm} \\ | \end{array} \right]_{d=2} \simeq \left[\begin{array}{c} \textcircled{\pm \frac{\pi}{2}} \\ | \end{array} \right], \quad \left[\begin{array}{c} \boxed{\pm} \\ | \end{array} \right]_{d=3} \simeq \left[\begin{array}{c} \textcircled{\pm 1 \atop \pm 1} \\ | \end{array} \right], \quad \left[\begin{array}{c} \boxed{\pm} \\ | \end{array} \right]_{d=4} \simeq \left[\begin{array}{c} \textcircled{\pi} \text{---} \textcircled{\pi} \\ | \end{array} \right] \quad (119)$$

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

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

$$\left[\begin{array}{c} \textcircled{\alpha \atop \beta} \\ | \end{array} \right] = \frac{1}{3} \begin{pmatrix} 1 + e^{i\alpha} + e^{i\beta} & 1 + \omega^2 e^{i\alpha} + \omega e^{i\beta} & 1 + \omega e^{i\alpha} + \omega^2 e^{i\beta} \\ 1 + \omega e^{i\alpha} + \omega^2 e^{i\beta} & 1 + e^{i\alpha} + e^{i\beta} & 1 + \omega^2 e^{i\alpha} + \omega e^{i\beta} \\ 1 + \omega^2 e^{i\alpha} + \omega e^{i\beta} & 1 + \omega e^{i\alpha} + \omega^2 e^{i\beta} & 1 + e^{i\alpha} + e^{i\beta} \end{pmatrix}$$

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

$$\left[\begin{array}{c} \textcircled{\pm \frac{\pi}{2}} \\ | \end{array} \right] = \frac{1}{2} \begin{pmatrix} 1 \pm i & 1 \mp i \\ 1 \mp i & 1 \pm i \end{pmatrix} = \frac{\sqrt{2}}{2} e^{\mp i\frac{\pi}{4}} \begin{pmatrix} \pm i & 1 \\ 1 & \pm i \end{pmatrix} = \frac{\sqrt{2}}{2} e^{\mp i\frac{\pi}{4}} \left[\begin{array}{c} \boxed{\pm} \\ | \end{array} \right]_{d=2} \quad (120)$$

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

(121)

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

(122)

Then we decompose the diagram in such a way that we can apply the standard interpretation:

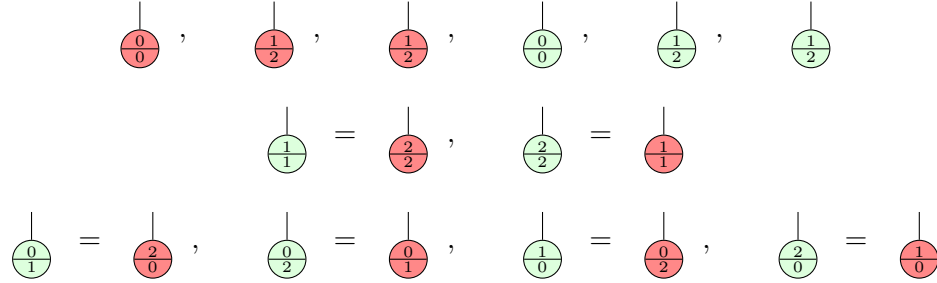
$$= \frac{\sqrt{2}}{8} \begin{pmatrix} -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \end{pmatrix}$$

$$= \frac{\sqrt{2}}{8} \left[\left[\begin{array}{c} | \\ \boxed{\pm} \\ | \end{array} \right] \right]_{d=4}$$

□

Now we prove the qutrit X -box is not a stabilizer diagram. The following lemma is required:

Lemma E.2. Every qutrit stabilizer state is equivalent to one of the following 12 diagrams:



Proof. We can use our qutrit elimination rules to prove this. A stabilizer state is a diagram with no inputs and one output, in which every phase component is a multiple of $\frac{2\pi}{3}$. After putting it in graph-like form (via Proposition 4.3), all but one spider will be interior. We'll call the non-interior one the *boundary* spider. Our elimination rules say we can remove all interior \mathcal{P} - and \mathcal{N} -spiders, plus all pairs of connected interior \mathcal{M} -spiders. So applying these rules until we can do so no more, we have two cases. The easiest case is where we end up with just the single boundary spider, which must have no inputs and one output. In the other case we get a single \mathcal{M} -spider connected to a boundary spider. The \mathcal{M} -spider can have no other legs, and the boundary spider must have exactly one other leg, which is the output of the overall diagram:

$$\begin{smallmatrix} | \\ \textcircled{a/b} \end{smallmatrix} \quad \text{or} \quad \begin{smallmatrix} | \\ \textcircled{a/b} \\ \text{---} \text{ } \boxed{\pm 1} \text{ ---} \\ \textcircled{m/-m} \end{smallmatrix} \quad (123)$$

In the first case, we're done. In the second case, we need only note:

$$\begin{smallmatrix} | \\ \textcircled{a/b} \\ \text{---} \text{ } \boxed{\pm 1} \text{ ---} \\ \textcircled{m/-m} \end{smallmatrix} \stackrel{(\text{cc})}{=} \begin{smallmatrix} | \\ \textcircled{a/b} \\ \text{---} \text{ } \textcircled{\pm m / \mp m} \end{smallmatrix} \stackrel{(110)}{=} \begin{smallmatrix} | \\ \textcircled{\pm m / \mp m} \end{smallmatrix}$$

□

Proposition E.3. The qutrit X -box defined below is not a stabilizer diagram:

$$\left[\begin{array}{c} | \\ \boxed{X} \\ | \end{array} \right] = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

Proof. Stabilizer diagrams are closed under composition (they form a group). Hence if the X -box is stabilizer, it must send the phaseless red spider state to a stabilizer state. This has matrix:

$$\left[\begin{array}{c} | \\ \boxed{X} \\ \bullet \end{array} \right] = \left[\begin{array}{c} | \\ \boxed{X} \\ | \end{array} \right] \left[\begin{array}{c} | \\ \bullet \end{array} \right] = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}$$

But, up to a scalar, this is not equal to any of the 12 stabilizer states above:

$$\left[\begin{array}{c} | \\ \bullet \end{array} \right] = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \quad \left[\begin{array}{c} | \\ \frac{1}{2} \end{array} \right] = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \quad \left[\begin{array}{c} | \\ \frac{2}{1} \end{array} \right] = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \quad \left[\begin{array}{c} | \\ \frac{a}{b} \end{array} \right] = \begin{pmatrix} 1 \\ \omega^a \\ \omega^b \end{pmatrix}$$

□