

Quantum Polynomials in the ZXW Calculus



Edwin Agnew

Wolfson College

University of Oxford

A thesis submitted for the degree of

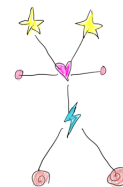
*Master of Science in Mathematics and the Foundations of Computer
Science*

August 2023

Abstract

Quantum pictorialism is an alternative approach to quantum information that replaces matrices with diagrams that are equipped with formal rewrite rules. Quantum pictorialism brings a new perspective to quantum information and emphasises *compositionality*, i.e. the importance of understanding how different processes compose with one another. One example of a new compositional perspective is the discovery in 2011 that two fundamental quantum states can be used to add and multiply numbers [\[1\]](#).

In this thesis, I generalise the arithmetic to addition and multiplication of *polynomials*. More specifically, working in a newly developed diagrammatic formalism called the ZXW calculus, I prove there is an isomorphism between a universal class of diagrams called controlled states and the ring of multilinear polynomials. Along the way, I prove a number of new results in the ZXW calculus, including the existence of two rings of controlled diagrams. Finally, I apply the isomorphism to yield a novel algorithm for entanglement detection and find a number of connections between quantum complexity theory and algebraic complexity theory.



Acknowledgements

To Bob, for proposing the connection between entanglement and ZXW arithmetic.

To Aleks, for putting me on to polynomials.

To Richie, for showing me the power of SPS.

To Razin, for pointing out the connection to controlled diagrams.

To Lia, for helping me tie it all together.

To Mary, for some inspiration.

Finally, to my friends from Wolfson and MFoCS for making this year such a wonderful one.

I thank you all immensely for your support.

Table of Contents

1	Introduction	1
1.1	The Way of the Diagram	2
1.2	Outline	4
2	Qubits	6
2.1	States	6
2.2	Dynamics	7
2.3	Composite Systems	8
2.4	Measurement	9
2.5	Quantum Circuits	10
2.6	Entanglement	11
2.7	Mixed States	14
3	Diagrams	16
3.1	Processes	16
3.1.1	Single Wires	16
3.1.2	Parallel Wires	18
3.1.3	Crossing Wires	20
3.1.4	Bending Wires	22
3.1.5	Process Theory	24
3.2	ZXW Calculus	26
3.2.1	Generators	27
3.2.2	Basic Rules	30
3.2.3	Arithmetical Rules	34
3.2.4	Complete Rule Set	37
3.2.5	Useful Lemmas	40
4	Controlled Diagrams	45

TABLE OF CONTENTS

4.1	Definitions	45
4.2	Copy coWs	50
4.3	Rings	54
5	Polynomials	61
5.1	Arithmetic in ZXW	61
5.2	Normal Forms	65
5.3	Substitution	75
5.4	Isomorphism	76
6	Applications	84
6.1	Entanglement Detection	84
6.1.1	Separability	84
6.1.2	Mixed States	87
6.2	Complexity Theory	89
6.2.1	Algebraic Complexity	91
6.2.2	Proof Complexity	96
7	Conclusion	102
7.1	Future Directions	103
	Bibliography	108
	Appendix A Algebra	114
A.1	Magmas	114
A.2	Monoids	115
A.3	Rings	115
A.3.1	Polynomials	117
A.4	Vector Spaces	117
	Appendix B Category Theory	120
B.1	Category	120

TABLE OF CONTENTS

B.2	Monoidal Category	121
B.2.1	Frobenius Algebra	121
B.3	Symmetric Monoidal Category	122
B.4	Compact Closed Category	123

1 | Introduction

“If I had asked people what they wanted, they would have said faster horses.”

–Henry Ford

We live at the apex of the age of information. Not only have digital technologies and communication come to dominate our everyday lives over the last 50 years, the study of information has also flourished in this time. Following Shannon’s rigorous definition of information in the 1940s [2], information theory has seeped from mathematics into diverse disciplines from physics to neuroscience [3] to geography [4]. Indeed, information has become so indispensable in physics that there are now some who claim it to be one of the most fundamental attributes of the universe [5]. However, taking information seriously in physics demands some modifications from Shannon’s conception in order to accommodate the more exotic phenomena of *quantum mechanics*.

While quantum information theory began as a branch of physics, it was eventually noticed that many of the weird bugs of nature it was wrestling with could be exploited for useful gain in computer science. One of the earliest discoveries was BB84 [6] - a key distribution protocol whose security depended only on the laws of physics. Since then, many more cryptographic developments have been made [7], as well as discoveries of exponential speed-ups in communication complexity [8] and several speed-ups in computational complexity [9], mostly notably a widely believed exponential speed-up in factoring [10]. Engineering progress has tried to keep pace with these theoretical developments and the past few years have seen a number of small-scale quantum computers being built, with the promise of genuinely useful devices being available in the coming decades.

Though there is still much conjecture and uncertainty, what is abundantly

clear is that quantum information provides more than just slightly faster Turing machines. I am confident that we have only begun to scratch the surface of what qubits have to offer. Unfortunately, dramatic breakthroughs have seemed to slow since the 90s. What may be hindering progress is how unintuitive quantum information is compared to its classical counter-part. The mathematical formalism is clunky¹, the philosophical foundations are a mess and the popular conception is that quantum = unintelligible magic.

1.1 The Way of the Diagram

At its core, quantum computation is simply matrix manipulation. Yet working directly with these matrices is unintuitive and about as tedious as working in assembly language. A notable symptom of this is that it took until 1992 - around 70 years after the birth of quantum mechanics - to discover the quantum teleportation protocol [11]. Even having done so, it is not at all clear from the circuit diagram (see fig 2.1) that it actually represents teleportation.

What is missing is not better theorists, but better notation. One potential source of higher abstraction for quantum information is **graphical calculi**. The main results of this dissertation are all proven in a graphical language called the ZXW-calculus. Other graphical calculi include the ZX, ZW and ZH calculus, among others. I will refer to them collectively as Z^* -calculi or simply Z^* . The ZXW-calculus is a very recent innovation but has its roots in applied category theory [12]. Already, it has found applications quantum chemistry [13] and quantum machine learning [14]. Other Z^* calculi have also found applications in circuit optimisation [15], error correction [16] and natural language processing [17], among much more.

¹Even von Neumann, one of the original proponents of Hilbert spaces, ended up denouncing his own formalism.

While neither the matrix formalism nor the diagrammatic formalism is strictly more useful than the other, I believe the main advantages of working with Z^* calculi are that they are more intuitive and logically rigorous. Though the diagrams are inspired by some quite heavy pure mathematics, working inside Z^* only requires learning a small number of rewrite rules. Moreover, these rewrite rules are complete [18] meaning that all proofs can be done “on the page”. Meanwhile, formally reasoning about a circuit diagram requires translating a picture to some matrices, then multiplying these matrices, then translating them back to a picture.

Nevertheless, one disadvantage of Z^* is that it is very easy to lose touch of what is physically realisable. Many of the generators are not unitary and so have no physical meaning. Moreover, quantum computers can only implement operations from a fixed gateset. While it is relatively trivial to translate from a circuit to a Z^* diagram, it turns out to be $\#\mathbf{P}$ -hard to extract a circuit from a generic ZX diagram [19]².

In summary, the distinction between circuit diagrams and string diagrams is a little like the distinction between Turing machines and the lambda calculus: the former has a little more physical meaning, while the latter is a little cleaner mathematically. However, both are computationally universal so it ultimately comes down to a matter of taste and application. Each approach provides a different perspective and the best insights always come from a synthesis of diverse perspectives. I favour the Z^* perspective in this work simply because being younger, it remains more unexplored.

²If you’re not familiar with complexity theory, think of $\#\mathbf{P}$ -hard as meaning very, very, hard.

1.2 Outline

I view this thesis as an attempt at diversifying our mathematical perspectives on quantum information. In the first half, I give a relatively self-contained introduction to quantum information. In chapter 1, I motivate this work and give an outline. In chapter 2, I summarise the standard Dirac formalism for qubits. In chapter 3, I lay the foundations for graphical reasoning and introduce a diagrammatic formalism called the ZXW calculus. This is the most extensive exposition of the ZXW calculus that I am aware of to date and includes a number of new lemmas.

In the second half of the thesis, I focus on a beautiful connection between ZXW diagrams and polynomials. In chapter 4, I introduce controlled diagrams and prove a number of results, including that controlled states form a ring. In chapter 5, I build up to the main result of this thesis which is an isomorphism between controlled states and the ring of multilinear polynomials. This is an exciting link because polynomials are a much older and better understood mathematical object than qubits so an isomorphism between them opens up a range of new techniques and perspectives. Though I discovered this connection independently, I was informed after completing an early draft of this thesis that it is in fact a known consequence of an folk isomorphism between tensors and multilinear maps. While the connection is used implicitly in works like [20], I have been unable to find an explicit proof for the result, despite contacting specialists in the field. Moreover, I am unaware of this isomorphism ever being deployed in quantum information and so chapter 6 discusses some speculative applications of the qubit-polynomial isomorphism. These include a novel algorithm for entanglement detection and some exciting connections with algebraic complexity theory. I end with some concluding remarks in chapter 7 and offer thoughts on a number of possible future directions.

1.2. *OUTLINE*

An anonymised code supplement for this thesis can be found at https://anonymous.4open.science/r/thesis_files-7COB/.

2 | Qubits

This chapter gives a very brief introduction to quantum information theory. For more details, see the textbook [21]. For the relevant mathematical background see appendix A.

A **bit** is an abstract unit of information which intuitively corresponds to the possibility of being in exactly two states: on or off. Bits can be physically realised, for example in light switches, transistors or clicky biros. But from an information-theoretic standpoint, the details of such physical implementations are irrelevant, so long as we have answered the question “can I flip it?”

Briefly remembering physical details, it turns out that very small things are not quite so simple. A quantum system contains information far more exotic than bits. A **qubit** is the most basic unit of quantum information. Naively, it corresponds to the possibility of being on, off, or both simultaneously. What exactly this means and whether it is true is the source of much controversy and confusion. I will mainly focus on the mathematical formalism and ignore the question of what is actually happening inside a qubit. But it is worth emphasising that qubits exist in nature and that scientists are getting better at constructing and manipulating them, most excitingly in the form of quantum computers.

2.1 States

As you probably know, bits live in $\mathbb{B} = \{0, 1\}$. Qubits (**quantum bits**), on the other hand, live in \mathbb{C}^2 (\mathbb{C} is the set of complex numbers). It is standard notation to write state vectors ψ in **kets** $|\psi\rangle$ and their complex conjugates

in **bras** $\langle\psi| = (|\psi\rangle)^\dagger$. Some particularly special states are:

$$|0\rangle := \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad |1\rangle := \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

These states are the analogues of classical bits. Since they form a basis (affectionately known as the **computational basis**), we can therefore write an arbitrary single qubit state $|\psi\rangle \in \mathbb{C}^2$ as:

$$\alpha|0\rangle + \beta|1\rangle = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}, \quad \alpha, \beta \in \mathbb{C}$$

Such a linear combination of basis states is usually called a **superposition** of $|0\rangle$ and $|1\rangle$. The coefficients α, β are called **amplitudes** and cannot be thought of as probabilities since they may be negative or complex in general. Instead, the squared modulus of an amplitude resembles a probability. So a state is said to be **normalised** if $\|\psi\|^2 = \langle\psi|\psi\rangle = |\alpha|^2 + |\beta|^2 = 1$. Normalised states correspond to those that are physically realistic, i.e. that could in theory be constructed with a quantum computer. Although many of the states in this thesis will not be normalised, one can always imagine rescaling them to be so, making no significant difference to their behaviour.

2.2 Dynamics

Having defined a qubit, we'd like to know what we can do with it. Naturally, we manipulate vectors with linear maps, i.e. matrices. A matrix U is physically possible if it preserves normalisation, i.e. $\|U\psi\| = \|\psi\|, \forall\psi$. If the input and output dimensions are the same, this is equivalent to requiring it be unitary, i.e. $UU^\dagger = U^\dagger U = I$, which at the very least means it must be

invertible. Some very important operations are

$$X := \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad Z := \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \quad H := \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

Not only are these all unitary, they are also self-adjoint and therefore involutive. The H gate sends the computational basis to the **X basis**, defined as:

$$\begin{aligned} H|0\rangle &= |+\rangle := \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \\ H|1\rangle &= |-\rangle := \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \end{aligned}$$

2.3 Composite Systems

Not a lot can be done with only a single qubit. We'd like to have lots. First, we at least need to be able to *describe* a system of several qubits. Given two qubits $|\psi_A\rangle, |\psi_B\rangle$, we can describe their **composite** state as $|\psi_A\psi_B\rangle := |\psi_A\rangle \otimes |\psi_B\rangle$, where \otimes is the Kronecker tensor product. For example,

$$|01\rangle := |0\rangle \otimes |1\rangle = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

It's worth emphasising that an n qubit state therefore has 2^n amplitudes - an awful lot! This should offer some intuition for the difficulty of classically simulating quantum computation [22] and the conjectured speed-up for quantum algorithms [9].

As before, unitary matrices are the allowed n -qubit operations. One of the

most important 2-qubit operations is the controlled-NOT:

$$CNOT := \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (2.1)$$

This gate gets its name because when the first qubit is $|0\rangle$, it does nothing and when the first qubit is $|1\rangle$, it flips the second, e.g. $CX|10\rangle = |11\rangle$. The CNOT gate is significant because it forms a universal gate set along with H and $T := \sqrt[4]{Z}$, i.e. any unitary can be approximated with some combination of $CNOT, H, T$ gates [21].

An important consequence of unitarity is that arbitrary qubits **cannot be cloned**. More formally:

Theorem 2.3.1: (No Cloning [23])

There is no unitary operation C that copies all states, i.e. maps $|\psi\rangle \otimes |e\rangle \mapsto |\psi\rangle \otimes |\psi\rangle$, for an arbitrary state $|\psi\rangle$ and some fixed initialisation state $|e\rangle$.

While copying specific states in a known basis is possible, the point is there is no single unitary that can copy all states.

2.4 Measurement

Though perhaps a little unintuitive, the story so far is quite clean: states come from $(\mathbb{C}^2)^{\otimes n}$ and evolve unitarily to preserve normalisation. Unfortunately, when clumsy macroscopic scientists start prodding a quantum system too much, a problematic process called **measurement** occurs. Measurement irreversibly collapses a quantum state into a new one, depending on the way

in which it is measured. More formally, given a set of orthogonal projectors $\{P_m\}$ such that $\sum_m P_m = I$ then a measurement on a state $|\psi\rangle$ gives outcome m with probability:

$$p(m) = \langle\psi|P_m|\psi\rangle$$

and leaves the system in the state:

$$\frac{P_m|\psi\rangle}{\sqrt{p(m)}}$$

The two most frequent examples are:

1. Measuring in the Z (computational) basis: $\{|0\rangle\langle 0|, |1\rangle\langle 1|\}$ which collapses $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ to $|0\rangle$ with probability $|\alpha|^2$ and to $|1\rangle$ with probability $|\beta|^2$.
2. Measuring in the X basis: $\{|+\rangle\langle +|, |-\rangle\langle -|\}$ which collapses $|\psi\rangle$ to $|+\rangle$ with probability $|\alpha + \beta|^2/2$ and to $|-\rangle$ with probability $|\alpha - \beta|^2/2$.

Measurements are a nuisance but are the only way of extracting classical information from a quantum system. The only situation in which a measurement does not change a state is when it is already equal to one of the projectors, for example measuring $|0\rangle$ in the Z basis gives back $|0\rangle$ deterministically.

2.5 Quantum Circuits

To summarise so far:

1. A qubit lives in \mathbb{C}^2 . A qubit $|\psi\rangle$ is normalised iff $\langle\psi|\psi\rangle = 1$.
2. Qubits evolve according to unitary matrices.
3. The composite system of multiple qubits is described with the Kro-

necker tensor product \otimes .

4. Measurement probabilistically collapses a state in order to extract classical information.

Quantum computation essentially boils down to preparing large states and making measurements on them. In the quantum circuit model, all qubits are initialised to some state (usually $|0\rangle$), a number of gates are applied and then measurements (typically in the Z basis) are applied at the end. Occasionally, further operations are applied that are conditioned on the classical outcomes of previous measurements. Since parsing long strings of matrix multiplications is quite uninformative, this is often presented as a graphical circuit. For example, the circuit for the quantum teleportation protocol is depicted in figure 2.1.

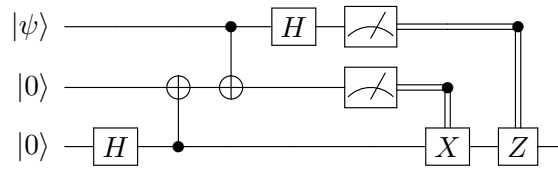


Figure 2.1: Quantum Teleportation Protocol [11]. Two qubit gates are CNOTs and double lines correspond to classical communication/control

2.6 Entanglement

Although we have established that arbitrary qubits cannot be cloned (theorem 2.3.1), the CNOT gate seems to come quite close. Taking the first qubit to be the control, one can see from (2.1) that

$$CNOT|00\rangle = |00\rangle, \quad CNOT|10\rangle = |11\rangle$$

So when the second qubit is initialised to $|0\rangle$, CNOT copies computational

basis states. To see that $CNOT| \cdot 0 \rangle$ does not copy every state consider

$$|Bell\rangle := CNOT|+0\rangle = \frac{CNOT(|00\rangle + |10\rangle)}{\sqrt{2}} = \frac{|00\rangle + |11\rangle}{\sqrt{2}} \quad (2.2)$$

This is not a copy of $|+\rangle$ since $|+\rangle \otimes |+\rangle = \frac{1}{4}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$. In fact $|Bell\rangle$ is a very interesting state because it cannot be rewritten as the tensor product of *any* states $|\psi_1\rangle \otimes |\psi_2\rangle$. Such a state is called inseparable or **entangled**. Otherwise, a state that can be written as a product of states is called **separable**.

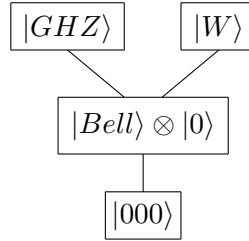
Entanglement is a very important feature of quantum information. It enables correlations between arbitrarily separated qubits that cannot be reproduced classically, a phenomenon known as **non-locality**. Famously, Einstein found this notion sufficiently spooky to consider it a bug of quantum mechanics and sought locally realistic explanations [24]. However, local realism has since been experimentally falsified [25] and entanglement is now considered a fundamental feature of quantum information. Computationally, there exists an alternative to the circuit model where universality is achieved by preparing a large entangled state then making a series of single qubit measurements on it [26]. Thus entanglement is not only real, but an extremely powerful resource.

Like electricity, money or patience, we treat entanglement as a **resource** because it can be used up to perform useful tasks. For example, the teleportation protocol in fig 2.1 uses a $|Bell\rangle$ state to transmit a qubit from Alice to Bob. The protocol also requires Alice to perform some local operations on her qubits and to communicate two classical bits to Bob but these are considered less interesting resources than the Bell state. More generally, in the resource theory of entanglement, local operations and classical communication (LOCC) are considered “free” operations because they cannot be used to generate or increase entanglement. This definition is often extended to

stochastic local operations and classical communication (SLOCC), i.e. where local operations need only succeed with some nonzero probability.

If a state $|\psi_1\rangle$ can be transformed into a state $|\psi_2\rangle$ using SLOCC, then we write $|\psi_1\rangle \geq_{SLOCC} |\psi_2\rangle$. Intuitively, this means that $|\psi_1\rangle$ is at least as useful as $|\psi_2\rangle$ because any useful task performed by $|\psi_2\rangle$ can also be performed by $|\psi_1\rangle$ (by first using SLOCC to turn it to $|\psi_2\rangle$). If both $|\psi_1\rangle \geq_{SLOCC} |\psi_2\rangle$ and $|\psi_2\rangle \geq_{SLOCC} |\psi_1\rangle$, then we say the two states are SLOCC-equivalent. This defines an equivalence class and is typically used to classify states by their operational behaviour.

For 2 qubits, there are only two SLOCC classes: one containing separable states and the other containing $|Bell\rangle$ from before. For 3 qubits, there are 4 SLOCC classes (up to permutations of qubits). Representatives of each class are depicted in the Hasse Diagram below.



The SLOCC-maximal representatives, $|GHZ\rangle$ and $|W\rangle$, are defined as follows:

$$|GHZ\rangle := \frac{1}{\sqrt{2}}(|000\rangle + |111\rangle)$$

$$|W\rangle := \frac{1}{\sqrt{3}}(|001\rangle + |010\rangle + |100\rangle)$$

These states both generalise to n qubits: $|GHZ_n\rangle$ producing a superposition of all $|0\rangle$'s and all $|1\rangle$'s and $|W_n\rangle$ producing a superposition of a single $|1\rangle$ on all possible qubits. Unfortunately, SLOCC classification breaks down for

$n > 3$ where there becomes an infinite number of SLOCC classes [27]. More generally, our understanding of multipartite entanglement breaks down soon after $n = 4$.

2.7 Mixed States

Suppose you measured a state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ but forgot to look at the result. Then rather being in a superposition, your state has “collapsed” to a probabilistic mixture of $|0\rangle$ and $|1\rangle$.

The help distinguish between superpositions and ignorance, theorists make use of the density matrix formalism, outlined as follows. Suppose a system is in some possible state $|\psi_i\rangle$ with probability p_i . Then the system is described by the **density matrix**:

$$\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|$$

Now applying some operation U that turns ρ to ρ' is described by conjugation:

$$\rho' = U\rho U^\dagger$$

The tensor product and measurement postulates generalise naturally. When $\rho = |\psi\rangle\langle\psi|$, then it is called a **pure state**. Otherwise, it is called a **mixed state**. Given some matrix ρ , it might not be obvious whether it can be written as a probabilistic mixture of pure states. Fortunately, we have the following characterisation: ρ is a density matrix corresponding to some collection $\{p_i, |\psi_i\rangle\}$ iff:

1. *trace*: $\text{tr}(\rho) = \sum_i \langle i|\rho|i\rangle = 1$.
2. *positivity*: $\forall|\phi\rangle, \langle\phi|\rho|\phi\rangle \geq 0$

Suppose we have a system ρ^{AB} that is shared between two parties - Alice

and Bob. Then all measurement statistics from Alice's perspective can be described by the **partial density operator**: $\rho^A = tr_B(\rho^{AB})$, where tr_B is the partial trace operation defined by:

$$tr_B(\rho^{AB}) = \sum_i (I^A \otimes \langle i|) \rho^{AB} (I^A \otimes |i\rangle)$$

Where I^A is the identity matrix on Alice's system and the sum is over all (computational) basis states. The partial trace can be thought of as forgetting about Bob. An interesting property of the entangled state $|Bell\rangle$ is that $tr_B(|Bell\rangle\langle Bell|) = I/2$, which is the known as the maximally mixed state. Thus, forgetting about one half of a maximally entangled state leads to complete ignorance.

3 | Diagrams

“It’s not the destination, it’s the journey”

– Ralph Waldo Emerson

The purpose of this chapter is to formalise the notion of plugging boxes together in order to properly introduce the ZXW calculus. This chapter is self-contained, though may be cross referenced with the corresponding definitions from category theory in Appendix B. The equation numbers intentionally match up so that, for example, (3.2) is a diagrammatic representation of (B.2). While the categorical content in the appendix is usually taught only to graduate maths students, I believe the diagrammatic counterpart presented in section 3.1 is sufficiently intuitive it could be taught to anyone. This chapter is heavily inspired by the exposition in [28].

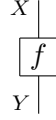
3.1 Processes

All diagrammatic reasoning in this thesis will be concerning processes - things that do things. Processes are a manifestation of the spirit of **compositionality**. Informally, compositionality is the idea that “things get more interesting when they are plugged together”. We are all familiar with plugging things together in the real world - this is something that hardly needs formalising. Instead, the purpose of formalising processes is so that we can bring our everyday intuitions about plugging wires to help solve abstract mathematical problems. First, we have to begin with the very basics.

3.1.1 Single Wires

A process is something that converts inputs to outputs. To aid abstraction, we won’t be particularly interested in how this conversion is done, only in the type of inputs a process expects to receive and the type of outputs it

promises to produce. A process $f : X \rightarrow Y$ is depicted as a box with inputs of type X above and outputs of type Y below:



f may be doing almost anything: converting numbers to numbers, electricity to light, coffee to writing or much more! By only imposing the *type* of input/output, we leave a process waiting to do something. It is necessary to limit inputs/outputs to a specific type, however, to prevent undefined situations like multiplying by cheese or adding 7 to an omelette.

Since we have abstracted away the details of what happens inside f , all we can talk about is how f relates to other boxes. Imagine we had some other process $g : Y \rightarrow Z$. Since the output of f is compatible with the input of g , we might like to feed f into g . This produces the **composite** process $g \circ f : X \rightarrow Z$ (read “ g after f ”), which may be considered a box in its own right. We depict $g \circ f$ by vertically stacking the individual boxes:

$$\begin{array}{c} X \\ | \\ \boxed{g \circ f} \\ | \\ Z \end{array} = \begin{array}{c} X \\ | \\ \boxed{f} \\ | \\ Y \\ | \\ \boxed{g} \\ | \\ Z \end{array} \quad (3.1)$$

When composing three different compatible processes f, g, h , there are actually two different ways to do so: stacking $(g \circ f)$ on top of h or stacking f on top of $(h \circ g)$. For all processes we consider, these compositions shall be equivalent which means we can unambiguously stack all three together at

once:

$$\begin{array}{c} X \\ | \\ \boxed{g \circ f} \\ | \\ Z \\ | \\ \boxed{h} \\ | \\ W \end{array} = \begin{array}{c} X \\ | \\ \boxed{f} \\ | \\ Y \\ | \\ \boxed{g} \\ | \\ Z \\ | \\ \boxed{h} \\ | \\ W \end{array} = \begin{array}{c} X \\ | \\ \boxed{f} \\ | \\ Y \\ | \\ \boxed{h \circ g} \\ | \\ W \end{array} \quad (3.2)$$

Finally, as is typical whenever we want to do something in mathematics, we require it be possible to **do nothing**. This “do nothing” operation is represented by a plain wire $id_X := |_X$ and satisfies:

$$\begin{array}{c} X \\ | \\ \boxed{f} \\ | \\ Y \end{array} = \begin{array}{c} X \\ | \\ \boxed{f} \\ | \\ Y \end{array} = \begin{array}{c} X \\ | \\ \boxed{f} \\ | \\ Y \end{array} \quad (3.3)$$

In summary, boxes can be plugged together and wires can be stretched.

3.1.2 Parallel Wires

So far, we can only plug boxes together along a single wire. To allow multiple wires in parallel (which is where things get interesting) we require some extra structure. A process which takes two inputs X and Y is defined as taking a single input of a new type $X \otimes Y$. Diagrammatically, $|_{X \otimes Y}$ looks like:

$$\begin{array}{c} X \otimes Y \\ | \\ X \otimes Y \end{array} = \begin{array}{c} X \quad Y \\ | \quad | \\ X \quad Y \end{array} \quad (3.4)$$

The horizontal composition of processes $f \otimes g$ can be thought of as doing “ f while g ”. So (3.4) says that doing nothing to $X \otimes Y$ is the same as

doing nothing to X while doing nothing to Y . As with vertical composition, horizontal composition is also associative:

$$\begin{array}{c} X_1 \otimes X_2 \\ \boxed{h \otimes g} \\ Y_1 \otimes Y_2 \end{array} \begin{array}{c} X_3 \\ \boxed{f} \\ Y_3 \end{array} = \begin{array}{c} X_1 \\ \boxed{h} \\ Y_1 \end{array} \begin{array}{c} X_2 \\ \boxed{g} \\ Y_2 \end{array} \begin{array}{c} X_3 \\ \boxed{f} \\ Y_3 \end{array} = \begin{array}{c} X_1 \\ \boxed{h} \\ Y_1 \end{array} \begin{array}{c} X_2 \otimes X_3 \\ \boxed{g \otimes f} \\ Y_2 \otimes Y_3 \end{array} \quad (3.5)$$

We further require that the order of vertical and horizontal composition does not matter:

$$\begin{array}{c} X_1 \\ \boxed{g_1 \circ f_1} \\ Z_1 \end{array} \begin{array}{c} X_2 \\ \boxed{g_2 \circ f_2} \\ Z_2 \end{array} = \begin{array}{c} X_1 \\ \boxed{f_1} \\ \boxed{g_1} \\ Z_1 \end{array} \begin{array}{c} X_2 \\ \boxed{f_2} \\ \boxed{g_2} \\ Z_2 \end{array} = \begin{array}{c} X_1 \otimes X_2 \\ \boxed{f_1 \otimes f_2} \\ \boxed{g_1 \otimes g_2} \\ Z_1 \otimes Z_2 \end{array} \quad (3.6)$$

(3.5) simply means there is no ambiguity when doing multiple things in parallel while (3.6) means vertical and horizontal composition work nicely together. Finally, now that horizontal composition gives us a new way to do things, we need a corresponding way to do nothing. We define an empty type called $\mathbf{1}$ and represent $\mid_{\mathbf{1}}$ (doing nothing to nothing) as empty space $\begin{smallmatrix} \cdots \\ \cdots \end{smallmatrix}$. That this does nothing means:

$$\begin{array}{c} X \\ \boxed{f} \\ Y \end{array} \begin{smallmatrix} \cdots \\ \cdots \end{smallmatrix} = \begin{array}{c} X \\ \boxed{f} \\ Y \end{array} = \begin{smallmatrix} \cdots \\ \cdots \end{smallmatrix} \begin{array}{c} X \\ \boxed{f} \\ Y \end{array} \quad (3.7)$$

$id_{\mathbf{1}} : \mathbf{1} \rightarrow \mathbf{1}$ isn't the only special process involving $\mathbf{1}$. Since $\mathbf{1}$ represents the empty type, then a process $\psi : \mathbf{1} \rightarrow X$ is considered to have no inputs. This is called a **state** and drawn as a triangle. Similarly, a process $\pi : X \rightarrow \mathbf{1}$ is

considered to have no outputs and is known as an **effect**.

$$\begin{array}{c} \mathbf{1} \mid \\ \boxed{\psi} \\ \mid \\ X \end{array} = \begin{array}{c} \triangle \psi \\ \mid \\ X \end{array} \qquad \begin{array}{c} X \mid \\ \boxed{\pi} \\ \mid \\ \mathbf{1} \end{array} = \begin{array}{c} X \mid \\ \nabla \pi \end{array}$$

Operationally, a state can be thought of as putting something into a system and an effect as a test for whether something came out. So plugging a state into an effect roughly corresponds to the probability the test succeeds on the given state. Such a process with neither inputs nor outputs is called a **number**.

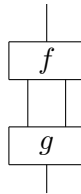
$$\begin{array}{c} \triangle \psi \\ \mid \\ X \\ \mid \\ \nabla \pi \end{array} = \diamond \lambda$$

An important number is **0** which corresponds to an impossible process. Horizontally composing anything with **0** will always give back **0** since you cannot make the impossible possible by doing other things elsewhere.

Since the wire labellings add clutter, they will henceforth be removed unless necessary.

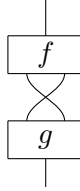
3.1.3 Crossing Wires

Consider the following situation:



Imagine that we'd prefer to connect the first output of f to the second input

g and vice-versa. Then diagrammatically this would look like:



This implicitly relies on a process \times that swaps its inputs. There are some important observations to be made about the swap. Clearly, swapping twice should do nothing (twice):

$$\times = \parallel \parallel \quad (3.8)$$

While swapping with the empty type should also do nothing:

$$x \times \mathbf{1} = x \parallel \quad (3.9)$$

Finally, we'd like to be able to slide boxes across swaps:

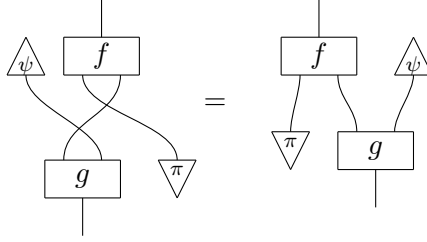
$$\begin{array}{c} \times \\ \boxed{f} \quad \boxed{g} \end{array} = \begin{array}{c} \parallel \quad \parallel \\ \boxed{g} \quad \boxed{f} \\ \times \end{array} \quad (3.10)$$

And slide swaps over identity wires:

$$\begin{array}{c} \times \\ \parallel \end{array} = \begin{array}{c} \parallel \\ \times \end{array} \quad (3.11)$$

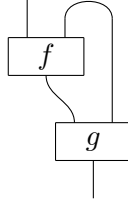
With swaps, we can start to see how flexible these diagrams become. We

can plug different outputs to different inputs in many different ways and untangle as much we like. For example, one can imagine literally dragging the triangles around in order to satisfy the equation:



3.1.4 Bending Wires

With swaps, we can connect any output to any input, but what if we wanted to connect two inputs? This could produce diagrams like the following:



Though doing so appears to mess with the flow of time, there are situations where it is possible - including qubits! Implicitly, such a wiring relies on a state \cap (called a cap) that transforms inputs to outputs. If we also allow an effect \cup (called a cup) that turns outputs to inputs, then we should expect that turning an input to an output and back again (or vice versa) does nothing:

$$\cup = | = \cap \quad (3.12)$$

If all a cap does is turn inputs to outputs, one would expect that both of its

wires should carry the same information. This can be expressed as:

$$\text{loop} = \text{cup} \quad (3.13)$$

An important consequence of introducing cups and caps is that states and processes become interchangeable. The idea is that we can take an arbitrary process f and turn it into a state by bending its input into an output:

$$\text{process } f \mapsto \text{state } f$$

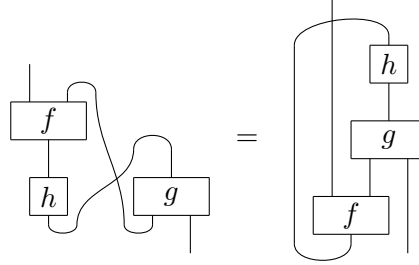
Similarly, we can take a bipartite state ψ and turn it into a process by bending one of its outputs to an input:

$$\text{state } \psi \mapsto \text{process } \psi$$

By (3.12), turning a process to a state and back again would give the original process and so this gives a bijection between states and processes. In quantum information theory, this is known as the Choi-Jamiołkowski isomorphism. Diagrammatically, this is known as **map-state duality**. Informally, I will refer to it as bending. This equation is also the crux of the quantum teleportation protocol from figure 2.1.

Another consequence of introducing caps and cups is that it gives us even more flexibility to re-arrange equations. While swaps allowed us to freely move boxes horizontally, caps and cups allow us to freely move them verti-

cally. For example,



Though this looks like quite a mess, the way to parse diagrams like these is to focus on which box is connected to which and in which order. So to see the equality holds, notice that the output of h is still connected to the input of g , that the first output of g is still connected to the second input of f , and so on. No matter where boxes are arranged on a page, diagrams will be equal if all of their connections are equal. This is perhaps the most important feature of diagrammatic reasoning. If you are to remember one thing, it should be the slogan:

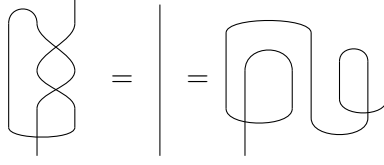
Only connectivity matters!

This is often abbreviated to OCM. Like associativity of addition, it is considered so natural that its usage will not be referenced explicitly in proofs or equations.

3.1.5 Process Theory

A **process theory** is any collection of types and processes that can be coherently plugged together. A specific process theory will interpret what happens inside a box and give meaning to composition. The more a process theory satisfies of equations (3.1) to (3.12), the more flexibility we have to plug boxes together. In other words, we want wires to be as wiggly as

possible:



Here are three examples (one fun, two serious) of a process theory:

1. **Lego** - the types are Tiny Tim's Lego blocks and the processes are the different ways Tim can fit the blocks together. For example, **stack** might take two square blocks and stick the first on top of the second. Vertical composition of processes (3.1) is Tim fitting some blocks together then sticking something else onto the resulting block. Clearly, Tim can do nothing (3.3) by not sticking anything together. This happens to be associative (3.2). Note that plugging boxes together therefore corresponds to combining different ways of plugging Lego blocks together. Horizontal composition of processes is Tim sticking blocks together while his clone, Tim₂, sticks some other blocks together. States correspond to Tim taking blocks out of their storage boxes and so effects correspond to Tim returning blocks to their boxes. The swap operation is Tim exchanging some blocks with Tim₂. Finally, unless Tim invents time travel, there are no cups or caps.
2. **Rel** - the process theory of relations. Types are sets X, Y, Z, \dots and processes are relations $R \subseteq X \times Y, S \subseteq Y \times Z, \dots$. xRy is used to denote $(x, y) \in R$. The composition of relations is defined as $x(S \circ R)z \iff \exists y, xRy \wedge ySz$. It can be easily checked that $S \circ R$ is again a relation and that this composition is associative and has unit $id_X = \{(x, x) : x \in X\}$. Horizontal composition of both types and processes is the Cartesian product \times , with unit $\mathbf{1} := \{*\}$. States and effects are both subsets of X . The swap operation is $\times_{X,Y} := \{(x, y), (y, x) : x \in X, y \in Y\}$. **Rel** has caps: $\cap_X := \{*, (x, x) : x \in X\}$ and cups:

$\cup_X := \{(x, x), * : x \in X\}$, satisfying (3.12). There are two numbers: $id_{\{*\}}$ and \emptyset , which intuitively correspond to possible and impossible. It is convenient to think of **Rel** as a possibilistic version of quantum mechanics.

3. **FdHilb** - types are finite dimensional Hilbert spaces (e.g. \mathbb{C}^2), processes are linear transformations. Composition is the usual matrix multiplication which is clearly closed, associative and unital. Horizontal composition is the Kronecker tensor product \otimes , with unit $\mathbf{1} := \mathbb{C}$. States are kets, effects are bras and numbers are complex scalars. So plugging a state into an effect gives the inner product, as one would hope. Swap is the usual swap matrix, e.g.

$$\begin{array}{c} \diagup \quad \diagdown \\ \mathbb{C}^2, \mathbb{C}^2 \end{array} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Finally, there are caps and cups, e.g. $\cup_{\mathbb{C}^2} := \begin{bmatrix} 1 & 0 & 0 & 1 \end{bmatrix}$. All Z^* calculi live in **FdHilb**, so this process theory is of particular interest.

The purpose of these examples is to illustrate how flexible the definition of a process theory is. While the remainder of this thesis will focus on one specific process theory, the whole point is that we are doing so within an abstraction that cares only about how things are plugged together.

3.2 ZXW Calculus

We are now ready to properly introduce the ZXW calculus. As already mentioned, this means we are working in **FdHilb**. More specifically, since this thesis deals with qubits, all wires will be \mathbb{C}^2 . Scalars will largely be ignored

and so equality of diagrams will, strictly speaking, usually mean equality up to a non-zero scalar. Similarly, many diagrams will not be normalised but could be made so by multiplying by the relevant scalar. An important, somewhat undiagrammatic, feature of **FdHilb** is that every type has a basis and so every process can be expanded as a sum over its action on basis vectors. Since a matrix is determined by what it does to a basis, one can prove equality of diagrams by plugging in combinations of $|0\rangle$ and $|1\rangle$ to input or output wires. This is considered undiagrammatic because it can lead to superficial disconnections, for example decomposing $I = |0\rangle\langle 0| + |1\rangle\langle 1|$. Usually, one hopes to keep a diagram as connected as possible to help visualise the flow of information.

3.2.1 Generators

The two main generators of the ZXW calculus are based off the $|GHZ\rangle$ and $|W\rangle$ states, the two maximally entangled 3-partite states from section 2.6. They are depicted as a green circle and black triangle respectively.

$$|GHZ\rangle = \text{green circle with three wires}, \quad |W\rangle = \text{black triangle with three wires}$$

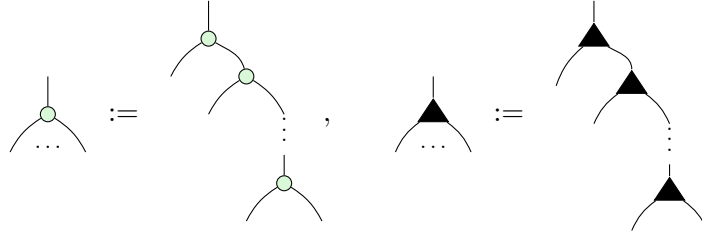
The problem with states is that they don't do anything, they simply are. Specifically, we cannot vertically compose states. Things get more interesting when we bend one of the wires to an input, producing the *Z node* and the *W node*.

$$\text{bent green circle} \rightsquigarrow \text{green circle} := |00\rangle\langle 0| + |11\rangle\langle 1| \quad (3.14)$$

$$\text{bent black triangle} \rightsquigarrow \text{black triangle} := |00\rangle\langle 0| + |01\rangle\langle 1| + |10\rangle\langle 1| \quad (3.15)$$

3.2. ZXW CALCULUS

Now there's inputs, we can plug nodes together. We define the generalised Z and W nodes accordingly:



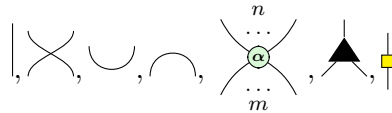
The Z node is further generalised to have multiple inputs and a multiplicative phase:

$$\begin{array}{c} n \\ \vdots \\ \text{---} \circ \alpha \text{---} \\ \vdots \\ m \end{array} := |0^m\rangle\langle 0^n| + e^{i\alpha}|1^m\rangle\langle 1^n|, \alpha \in \mathbb{C}$$

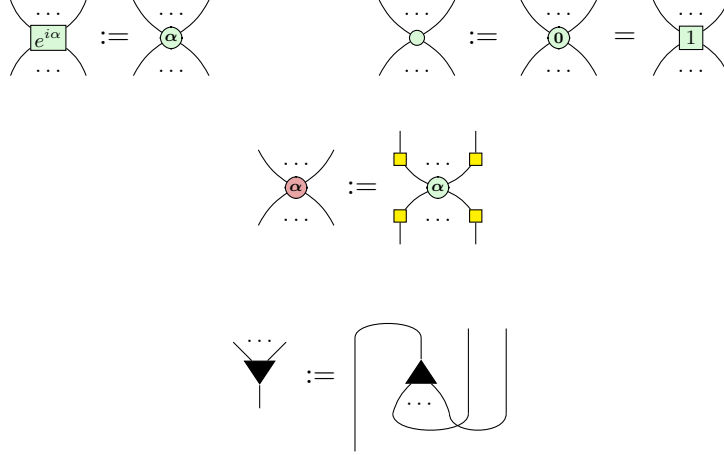
This generalised Z is known as a spider since it is generated by a special commutative Frobenius algebra (SCFA) and so is uniquely determined by the number of inputs, outputs and phase due to something known as the spider theorem (see appendix B.2.1). Finally, we draw the Hadamard gate as a yellow box:

$$\begin{array}{c} \text{---} \boxed{\text{yellow}} \text{---} \end{array} := \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

Then all ZXW diagrams are built out of the following generators:



For convenience, we will also use the following notation:



Before moving to the rules, it is worth noticing the following connections to definitions from chapter 2:

- The Z spider gets its name from the Z gate since $\text{Z spider} = \text{Z gate} = Z$.
- Similarly, the red spider is sometimes called the X spider since $\text{red spider} = X$.
- Computational basis states are red: $|0\rangle$, $|1\rangle$.
- Somewhat confusingly, X basis states are green: $|+\rangle$, $|-\rangle$.
- Red and green cups both equal the Bell state $|Bell\rangle = \text{cup} = \text{green cup} = \text{red cup}$.
- The CNOT gate is simply $\text{CNOT} = \text{green spider} - \text{red spider} = \text{green spider} - \text{red spider}$. Note that a horizontal wire can be interpreted as either an input or an output due to the principle of OCM.
- $T = \text{green circle with } \pi/4$. This suffices to show universality of ZXW diagrams since $\{H, T, CNOT\}$ is a universal gate set.

It is worth highlighting that the main generators, Z and W nodes, are built by bending and fusing the only maximally entangled 3-partite states. Of course, our generators should be inseparable (i.e. entangled) otherwise the

resulting diagrams would be disconnected. But the number 3 is significant for three reasons:

- Bending is what allows the generators to be composed in order to build arbitrarily large diagrams. Since bending a 2-partite state results in a straight wire, we would not get very far using generators with less than three wires.
- As mentioned in section 2.6, SLOCC classification breaks down for $n > 3$. So there would have been infinitely many options if we had opted for 4-partite generators.
- We shall soon get a glimpse of the connection between algebra and entanglement. Algebra is primarily occupied with binary operations (two inputs, one output). Typically, larger n -ary operations are handled by decomposing them into a series of associative binary operations. By analogy, the ZXW calculus decomposes all diagrams into a series of fusible generators.

So according to compositionality, three is the magic number!

3.2.2 Basic Rules

This section introduces some of the most important rules of the ZXW calculus. As rules they only need to be true when interpreted as matrices, however I also try to give some justification or commentary to prevent them all from feeling like meaningless shapes on a page.

As the definition (3.14) suggests, the Z-spider copies computational basis states:

$$\begin{array}{c} \textcircled{k\pi} \\ | \\ \textcircled{\pi} \\ \text{---} \end{array} = \begin{array}{c} \textcircled{k\pi} \\ | \end{array} \begin{array}{c} \textcircled{k\pi} \\ | \end{array}, k \in \{0, 1\} \quad (\text{K0})$$

Since the Z spider copies a basis, it corresponds to an SCFA [29] and therefore

satisfies the “fusion” equation:

$$\text{Diagram (S1)} \quad (S1)$$

Fusion is so fundamental that I will rarely reference its usage explicitly. The X spider also satisfies fusion (but with addition of phases) and copies green basis states (3.22, 3.21). In fact, any equality between diagrams containing only red and green spiders will still hold with all the colours flipped. They can also be vertically flipped meaning behaves like an upside down copy, i.e. goes to zero unless its inputs are identical.

Thanks to fusion, behaves like a monoid with unit . In fact, it computes XOR on red basis states , :

$$\text{Diagram} \quad j, k \in \{0, 1\}$$

Since XOR-ing then copying is the same as copying twice then XOR-ing twice, we have the following bialgebra rule:

$$\text{Diagram (B2)} \quad (B2)$$

By (3.15), the W node also copies , but behaves more like on :

$$\text{Diagram (3.16)} \quad (3.16)$$

The bending effect used in (3.15) is therefore equal to . Plugging $\langle 0|$

to the bottom of the W gives the identity:

$$\begin{array}{c} \text{---} \\ \diagup \quad \diagdown \\ \bullet \end{array} = \text{---} = \begin{array}{c} \text{---} \\ \diagdown \quad \diagup \\ \bullet \end{array} \quad (3.17)$$

So it has a co-unit. Plugging $\langle 1|$ disconnects:

$$\begin{array}{c} \text{---} \\ \diagup \quad \diagdown \\ \circ \end{array} = \begin{array}{c} \text{---} \\ \circ \\ \circ \\ \text{---} \end{array} \quad (3.18)$$

It turns out that the W node does not form an SCFA. In fact it forms something called an anti-special CFA [30]. Thus there is no W spider fusion so the W node is usually defined to have only one input. Nevertheless, it is still associative and commutative:

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

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

Together these allow for a restricted version fusion between Ws, but with only one input, which is what makes the definition of $\begin{array}{c} \text{---} \\ \diagup \quad \diagdown \\ \bullet \end{array}$ well defined:

$$\begin{array}{c} \text{---} \\ \diagup \quad \diagdown \\ \text{---} \\ \diagup \quad \diagdown \\ \text{---} \end{array} = \begin{array}{c} \text{---} \\ \diagup \quad \diagdown \\ \text{---} \end{array}$$


Things get most interesting when we relate two different generators. Perhaps the most important ZXW rule, interchanging Z and W nodes, is the following

bialgebra:



(BZW)

To verify this, imagine plugging in red basis states at the bottom. Suppose the bottom left wire is $\langle 0|$. Then


The same argument applies for the bottom right wire being $\langle 0|$. Finally, if both inputs are $\langle 1|$, then by (3.18), both sides go to zero.

Another way to interpret (BZW) is that  copies the Z spider and then entangles the Zs' outputs with a new W to preserve the total number of wires:

The diagrammatic equation (BZW) is shown as an equality between two Feynman diagrams. On the left, a vertex (green circle) is connected to a triangle loop (black triangle). On the right, a vertex (green circle) is connected to a self-energy insertion (black triangle) on a propagator line. The equation is labeled (BZW) with an equals sign below it.

Surprisingly, under conditions like these,  behaves sufficiently similarly to  that they become interchangeable:

(TA)

This can be seen by inspecting the matrices and noticing they only differ in the $\langle 11|$ column. This differing behaviour will never make a difference in a diagram like (TA) because (3.18) prevents the top left  being fed $\langle 11|$

and thus also the neighbouring  or .

$$\begin{array}{c} \text{black triangle} \\ \diagup \quad \diagdown \end{array} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \quad \begin{array}{c} \text{red dot} \\ \diagup \quad \diagdown \end{array} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ \mathbf{1} & \mathbf{0} \end{bmatrix}$$

The rules so far are largely based around plugging in red basis states, copying and fusion. When I wish to emphasise this particular flavour of ZXW in a proof, I will say something like “plugging red”.

3.2.3 Arithmetical Rules

Another flavour of ZXW stems from the generators’ ability to perform arithmetic [1]. If we represent a number $a \in \mathbb{C}$ as the effect $\boxed{a} = \begin{bmatrix} 1 & a \end{bmatrix}$, then it turns out:

$$\begin{array}{c} \text{black triangle} \\ \diagup \quad \diagdown \\ \boxed{a} \quad \boxed{b} \end{array} = \begin{array}{c} | \\ \boxed{a+b} \end{array} \quad (\text{ADD})$$

$$\begin{array}{c} \text{green dot} \\ \diagup \quad \diagdown \\ \boxed{a} \quad \boxed{b} \end{array} = \begin{array}{c} | \\ \boxed{a \times b} \end{array} \quad (\text{MUL})$$

Though (MUL) of course follows from (S1), it can be convenient to think of this as a rule in itself to emphasise the arithmetical perspective of ZXW, which I will signpost with phrases like “plugging green”.

Since $\langle 0| = \boxed{0}$, then another way of expressing (3.17) is:

$$\forall a \in \mathbb{C}, \quad \begin{array}{c} \text{black triangle} \\ \diagup \quad \diagdown \\ \boxed{a} \quad \text{red dot} \end{array} = \begin{array}{c} \text{black triangle} \\ \diagup \quad \diagdown \\ \boxed{a} \quad \boxed{0} \end{array} \stackrel{(\text{ADD})}{=} \begin{array}{c} | \\ \boxed{a+0} \end{array} = \begin{array}{c} | \\ \boxed{a} \end{array}$$

Note that π is the additive inverse since:

$$\begin{array}{c} | \\ \boxed{a} \end{array} \circ \begin{array}{c} | \\ \pi \end{array} = \begin{bmatrix} 1 & a \end{bmatrix} Z = \begin{bmatrix} 1 & -a \end{bmatrix} = \begin{array}{c} | \\ \boxed{-a} \end{array}$$

Meanwhile $X = \pi$ is the multiplicative inverse since

$$\begin{array}{c} | \\ \boxed{a} \end{array} \circ \begin{array}{c} | \\ \pi \end{array} = \begin{bmatrix} 1 & a \end{bmatrix} X = \begin{bmatrix} a & 1 \end{bmatrix} = a \begin{bmatrix} 1 & 1/a \end{bmatrix} \approx \begin{bmatrix} 1 & 1/a \end{bmatrix} = \begin{array}{c} | \\ \boxed{1/a} \end{array}$$

Due to the rescaling trick $\begin{bmatrix} \alpha & \beta \end{bmatrix} \approx \begin{bmatrix} 1 & \beta/\alpha \end{bmatrix}$, one can treat almost any single qubit effect as a number. The sole exception is $\beta\langle 1| = \begin{bmatrix} 0 & \beta \end{bmatrix}$ which is sort of like infinity.

The final thing to check to ensure $\left(\begin{array}{c} | \\ \boxed{a} \end{array}, \blacktriangle, \begin{array}{c} | \\ \pi \end{array} \right)$ can perform arithmetic is distributivity. This is enabled by the rule:

$$\begin{array}{c} | \\ \boxed{a} \end{array} \blacktriangle = \begin{array}{c} | \\ \blacktriangle \end{array} \begin{array}{c} | \\ \boxed{a} \end{array} \begin{array}{c} | \\ \boxed{a} \end{array} \quad (\text{Pcy})$$

This rule captures distributivity since:

$$\begin{array}{c} | \\ \boxed{a(b+c)} \end{array} = \begin{array}{c} | \\ \pi \end{array} \begin{array}{c} | \\ \boxed{a} \end{array} \blacktriangle \begin{array}{c} | \\ \boxed{b} \end{array} \begin{array}{c} | \\ \boxed{c} \end{array} = \begin{array}{c} | \\ \boxed{a} \end{array} \blacktriangle \begin{array}{c} | \\ \boxed{b} \end{array} \begin{array}{c} | \\ \boxed{c} \end{array} \stackrel{(\text{Pcy})}{=} \begin{array}{c} | \\ \blacktriangle \end{array} \begin{array}{c} | \\ \boxed{a} \end{array} \begin{array}{c} | \\ \boxed{a} \end{array} \begin{array}{c} | \\ \boxed{b} \end{array} \begin{array}{c} | \\ \boxed{c} \end{array} = \begin{array}{c} | \\ \boxed{ab+ac} \end{array} \quad (3.19)$$

We can use distributivity and (TA) to show that the W node effectively

copies number states in certain situations.

$$\begin{array}{c}
 \begin{array}{c} \text{Diagram 1} \\ \text{(Pcy)} \\ \text{Diagram 2} \end{array} = \begin{array}{c} \text{Diagram 3} \\ \text{(TA)} \\ \text{Diagram 4} \end{array} \quad (3.20) \\
 \begin{array}{c} \text{Diagram 5} \\ \text{(3.21)} \\ \text{Diagram 6} \end{array} = \begin{array}{c} \text{Diagram 7} \\ \text{Diagram 8} \end{array}
 \end{array}$$



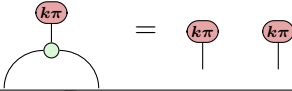
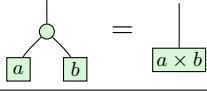
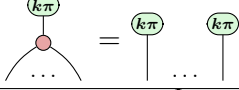

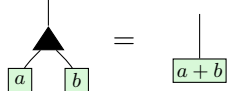
We shall explore this copying in more detail in section 4.2. For now, it suffices to say that whenever a W node is stuck onto the side of diagram with \circ 's, then any other \blacktriangledown will copy numbers. In such a situation, we might expect that adding then copying is the same as copying twice then adding twice. Indeed this is the case, forming a bialgebra similar to (B2) and (BZW), but with the extra W inserted to enable the copying behaviour:

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

Many ZXW diagrams can be interpreted as performing some combination of additions and multiplications on their inputs. In which case, it is natural to imagine plugging in \boxed{a} . If two diagrams can be seen to perform the same arithmetic function on *all* green inputs, then they must be equal because $\boxed{1}, \boxed{-1}$ are two particular number states which form a basis.

The ability to alternate between the red and green perspectives gives us a lot of freedom for how to interpret ZXW diagrams. Chapter 4 focuses on controlled diagrams which are mostly based around plugging red. Chapter 5 essentially generalises [1] and so is mostly based around plugging green. This informal distinction between the red and green perspectives is summarised

below.

Perspective	Red	Green
<i>Interpretation</i>	Copy and control	Arithmetic
<i>Time</i>	↓	↑
<i>Plugging states</i>		
<i>Z</i>		
<i>X</i>		N/A
<i>W</i>		
<i>Exemplary result</i>	Theorem 4.3.1	Proposition 5.2.5

Since the red perspective is reminiscent of the ZX calculus and the green perspective is reminiscent of the ZW calculus, the power of the ZXW calculus comes from its ability to unify these perspectives simultaneously. Theorem 5.4.1, the main result of this thesis, is a wonderful example of this synthesis.

3.2.4 Complete Rule Set

You have been presented a collection of rules for manipulating diagrams and a way of interpreting these diagrams as matrices. Technically, this means we have defined a logical system with diagrammatic rewrites as syntax and matrix equality as semantics. Two very important properties to consider about every logical system are:

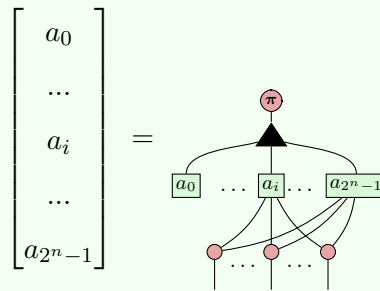
1. **Soundness:** if the rules prove something to be true, then it is indeed true.
2. **Completeness:** if something is true, then there is a way to prove it using the rules.

Soundness is quite a weak requirement and for ZXW can be easily verified by translating each of the rules into matrices and checking the two sides are equal. Completeness is usually more elusive and for ZXW means that any equality of matrices can be proven diagrammatically. Fortunately, completeness of the ZXW calculus was proven earlier this year [18]. This adds to a line of Z^* completeness results, dating back to qubit ZW completeness in 2015 [31].

The most recent result actually proves completeness for qudits of arbitrary dimension¹. However, since this thesis deals specifically with qubits, I will present only the relevant rules. Very briefly, the proof depends on finding a normal form such that any state can be rewritten into its normal form using ZXW rules. Then whenever two states are equal, they can be rewritten to the same normal form and equality can be shown by reversing half of the rewrites. The fact that equality of states is sufficient for completeness is due to a clever bending trick. The normal form used in the proof (which I call the completeness normal form) is presented below:

Definition 3.2.1: Completeness Normal Form (CoNF)

The CoNF of an arbitrary n qubit state is



The box a_i is wired according to the binary expansion of i .

For completeness, I include the complete list of rules used in the proof of

¹In qubits, there are two basis states: $|0\rangle, |1\rangle \in \mathbb{C}^2$. In qudits, there are d basis states: $|0\rangle, |1\rangle, |2\rangle, \dots, |d-1\rangle \in \mathbb{C}^d$

ZXW completeness below. The rules and their names have been taken directly from [18], then modified for the qubit case.

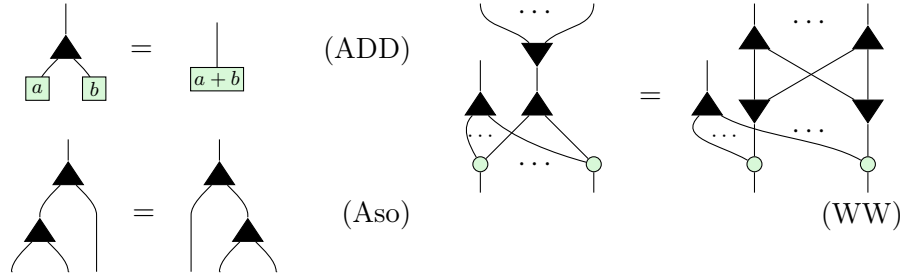
ZX Rules:

$$\begin{array}{lcl}
 \begin{array}{c} \text{Diagram 1} \end{array} & = & \begin{array}{c} \text{Diagram 2} \end{array} = \begin{array}{c} \text{Diagram 3} \end{array} \quad (S1) \\
 \begin{array}{c} \text{Diagram 4} \end{array} = \begin{array}{c} \text{Diagram 5} \end{array} = \begin{array}{c} \text{Diagram 6} \end{array} \quad (S2) & \begin{array}{c} \text{Diagram 7} \end{array} = \begin{array}{c} \text{Diagram 8} \end{array} \quad (K1) \\
 \begin{array}{c} \text{Diagram 9} \end{array} = \begin{array}{c} \text{Diagram 10} \end{array} \quad (Ept) & \begin{array}{c} \text{Diagram 11} \end{array} = \begin{array}{c} \text{Diagram 12} \end{array} \quad (K2) \\
 \begin{array}{c} \text{Diagram 13} \end{array} = \begin{array}{c} \text{Diagram 14} \end{array} \quad (B2) & \begin{array}{c} \text{Diagram 15} \end{array} = \begin{array}{c} \text{Diagram 16} \end{array} \quad (K2) \\
 \begin{array}{c} \text{Diagram 17} \end{array} = \begin{array}{c} \text{Diagram 18} \end{array} \quad (K0) & \begin{array}{c} \text{Diagram 19} \end{array} = \begin{array}{c} \text{Diagram 20} \end{array} \quad (Zer) \\
 \begin{array}{c} \text{Diagram 21} \end{array} = \begin{array}{c} \text{Diagram 22} \end{array} \quad (H)
 \end{array}$$

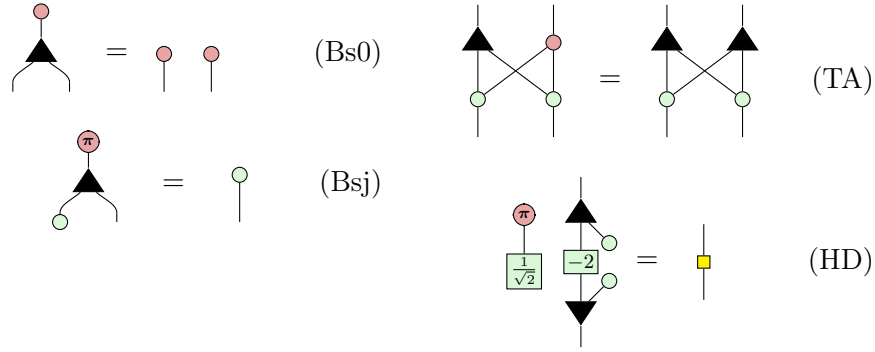
Where $k \in \{0, 1\}$.

ZW Rules:

$$\begin{array}{lcl}
 \begin{array}{c} \text{Diagram 1} \end{array} = \begin{array}{c} \text{Diagram 2} \end{array} \quad (Pcy) & \begin{array}{c} \text{Diagram 3} \end{array} = \begin{array}{c} \text{Diagram 4} \end{array} \quad (BZW) \\
 \begin{array}{c} \text{Diagram 5} \end{array} = \begin{array}{c} \text{Diagram 6} \end{array} \quad (Sym)
 \end{array}$$



ZXW Rules:



3.2.5 Useful Lemmas

To help get a better feel for diagrammatic reasoning within these rules, let's establish some lemmas that will be useful later.

Lemma 3.2.2

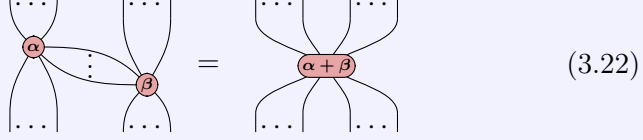
$$\text{Diagram with a red circle and a green circle labeled } k\pi \text{ as inputs} = \text{Diagram with two green circles labeled } k\pi \text{ as inputs} \quad (3.21)$$

Proof.

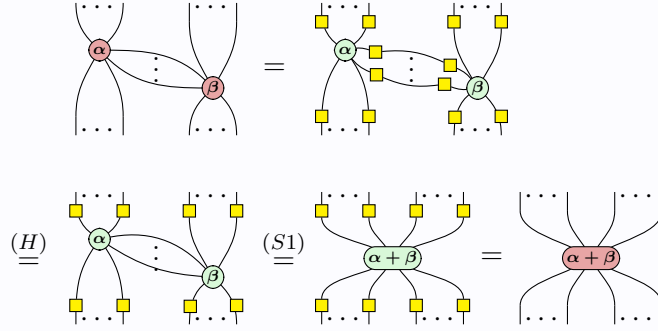
$$\text{Diagram with a red circle and a green circle labeled } k\pi \text{ as inputs} = \text{Diagram with a red circle and a green circle labeled } k\pi \text{ as inputs, with a yellow square as input} \stackrel{(H)}{=} \text{Diagram with a red circle and a green circle labeled } k\pi \text{ as inputs, with a yellow square as input} \stackrel{(K0)}{=} \text{Diagram with two green circles labeled } k\pi \text{ as inputs}$$

□

Lemma 3.2.3: X-spider fusion



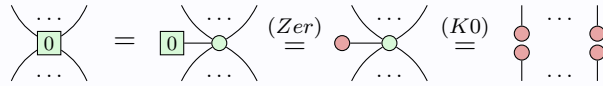
Proof.



Lemma 3.2.4



Proof.



Lemma 3.2.5



Proof.

$$\begin{array}{c} | \\ \boxed{a} \\ \bullet \end{array} = \begin{array}{c} | \\ \bullet \\ | \\ \boxed{a} \end{array} \begin{array}{c} | \\ \bullet \\ | \\ \bullet \end{array} \stackrel{(Zer)}{=} \begin{array}{c} | \\ \bullet \\ | \\ \boxed{a} \end{array} \begin{array}{c} | \\ \bullet \\ | \\ \boxed{0} \end{array} \stackrel{(MUL)}{=} \begin{array}{c} | \\ \bullet \\ | \\ \boxed{0} \end{array} \begin{array}{c} | \\ \bullet \\ | \\ \bullet \end{array} \stackrel{(Zer)}{=} \begin{array}{c} | \\ \bullet \\ | \\ \bullet \end{array}$$

□

Lemma 3.2.6

$$\begin{array}{c} \blacktriangle \\ | \\ \boxed{a} \end{array} \begin{array}{c} \blacktriangle \\ | \\ \boxed{b} \end{array} = \begin{array}{c} | \\ \boxed{a+b} \\ | \end{array} \quad (3.25)$$

Proof. This is a simple case of (BZW) in disguise:

$$\begin{array}{c} \blacktriangle \\ | \\ \boxed{a} \end{array} \begin{array}{c} \blacktriangle \\ | \\ \boxed{b} \end{array} = \begin{array}{c} | \\ \bullet \\ | \\ \boxed{a} \end{array} \begin{array}{c} | \\ \bullet \\ | \\ \boxed{b} \end{array} = \begin{array}{c} \curvearrowright \\ | \\ \bullet \\ | \\ \boxed{a} \end{array} \begin{array}{c} \curvearrowright \\ | \\ \bullet \\ | \\ \boxed{b} \end{array} \stackrel{(BZW)}{=} \begin{array}{c} | \\ \bullet \\ | \\ \bullet \end{array} \begin{array}{c} | \\ \bullet \\ | \\ \bullet \end{array} \stackrel{(ADD)}{=} \begin{array}{c} | \\ \boxed{a+b} \\ | \end{array}$$

□

In particular, this implies:

$$\begin{array}{c} \blacktriangle \\ | \\ \bullet \end{array} \begin{array}{c} \blacktriangle \\ | \\ \bullet \end{array} = \begin{array}{c} \blacktriangle \\ | \\ \boxed{1} \end{array} \begin{array}{c} \blacktriangle \\ | \\ \boxed{1} \end{array} \stackrel{(3.25)}{=} \begin{array}{c} | \\ \boxed{2} \\ | \end{array} \quad (3.26)$$

And similarly:

$$\begin{array}{c} \blacktriangle \\ | \\ \bullet \end{array} \begin{array}{c} \bullet \\ | \\ \blacktriangle \end{array} = \begin{array}{c} \blacktriangle \\ | \\ \boxed{1} \end{array} \begin{array}{c} \bullet \\ | \\ \boxed{-1} \end{array} \stackrel{(3.25)}{=} \begin{array}{c} | \\ \boxed{0} \\ | \end{array} \stackrel{(3.23)}{=} \begin{array}{c} \bullet \\ | \\ \bullet \end{array} \quad (3.27)$$

Moreover, (3.25) generalises straightforwardly to n coefficients:

$$\begin{array}{c}
 \begin{array}{ccc}
 \begin{array}{c} \text{Diagram 1: } a_1, a_2, \dots, a_n \text{ in boxes, top and bottom nodes, arcs } a_1 \rightarrow a_2 \rightarrow \dots \rightarrow a_n \text{ and } a_n \rightarrow a_1 \end{array} & \stackrel{(Aso)}{=} & \begin{array}{c} \text{Diagram 2: } a_1, a_2, \dots, a_n \text{ in boxes, top and bottom nodes, arcs } a_1 \rightarrow a_2 \rightarrow \dots \rightarrow a_n \text{ and } a_n \rightarrow a_1 \end{array} \\
 & & \stackrel{(3.25)}{=} \begin{array}{c} \text{Diagram 3: } a_1 + a_2, \dots, a_n \text{ in boxes, top and bottom nodes, arcs } a_1 + a_2 \rightarrow \dots \rightarrow a_n \text{ and } a_n \rightarrow a_1 + a_2 \end{array}
 \end{array} \\
 \\
 \begin{array}{c}
 \begin{array}{c} \text{Diagram 4: } \sum_{i=1}^{n-1} a_i, a_n \text{ in boxes, top and bottom nodes, arcs } \sum_{i=1}^{n-1} a_i \rightarrow a_n \text{ and } a_n \rightarrow \sum_{i=1}^{n-1} a_i \end{array} & \stackrel{(3.25)}{=} & \begin{array}{c} \text{Diagram 5: } \sum_{i=1}^n a_i \text{ in a box, top and bottom nodes, arcs } \sum_{i=1}^n a_i \rightarrow \sum_{i=1}^n a_i \text{ and } \sum_{i=1}^n a_i \rightarrow \sum_{i=1}^n a_i \end{array}
 \end{array}
 \end{array} \tag{3.28}$$

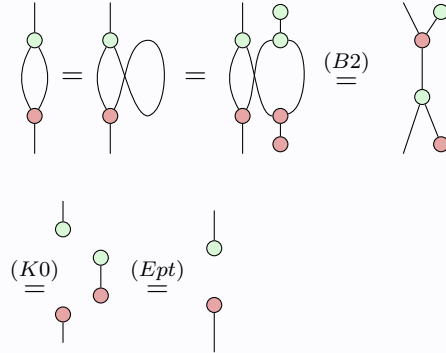
With multiple outputs this becomes:

$$\begin{array}{c}
 \begin{array}{ccc}
 \begin{array}{c} \text{Diagram 1: } a_1, a_2, \dots, a_n \text{ in boxes, top and bottom nodes, arcs } a_1 \rightarrow a_2 \rightarrow \dots \rightarrow a_n \text{ and } a_n \rightarrow a_1 \end{array} & = & \begin{array}{c} \text{Diagram 2: } a_1, a_2, \dots, a_n \text{ in boxes, top and bottom nodes, arcs } a_1 \rightarrow a_2 \rightarrow \dots \rightarrow a_n \text{ and } a_n \rightarrow a_1 \end{array} \\
 & & \stackrel{(BZW)}{=} \begin{array}{c} \text{Diagram 3: } a_1, a_2, \dots, a_n \text{ in boxes, top and bottom nodes, arcs } a_1 \rightarrow a_2 \rightarrow \dots \rightarrow a_n \text{ and } a_n \rightarrow a_1 \end{array} \\
 & & \stackrel{(3.28)}{=} \begin{array}{c} \text{Diagram 4: } \sum_{i=1}^n a_i \text{ in a box, top and bottom nodes, arcs } \sum_{i=1}^n a_i \rightarrow \sum_{i=1}^n a_i \text{ and } \sum_{i=1}^n a_i \rightarrow \sum_{i=1}^n a_i \end{array}
 \end{array}
 \end{array} \tag{3.29}$$

Lemma 3.2.7: Hopf Law

$$\begin{array}{c}
 \begin{array}{c} \text{Diagram 1: } \text{A box with } a_1 \text{ and } a_2 \text{ inside, top and bottom nodes, arcs } a_1 \rightarrow a_2 \text{ and } a_2 \rightarrow a_1 \end{array} = \begin{array}{c} \text{Diagram 2: } \text{A box with } a_1 \text{ and } a_2 \text{ inside, top and bottom nodes, arcs } a_1 \rightarrow a_2 \text{ and } a_2 \rightarrow a_1 \end{array} \\
 \end{array} \tag{3.30}$$

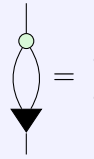
Proof.



□

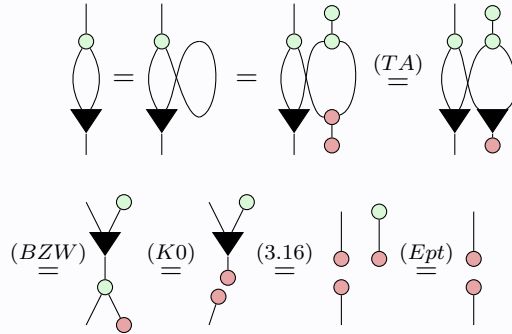
Very similarly,

Lemma 3.2.8



(3.31)

Proof.



□

4 | Controlled Diagrams

“One ring to control them all, one ring to find them, ...”

– JRR Tolkien [adapted]

It’s been known since 2011 that $(\boxed{a}, \blacktriangle, \text{green circle}) \simeq (\mathbb{C}, +, \times)$ [1]. It was the ability for W to do sums that helped the first completeness proof for the ZW calculus [31]. It was recently found that W could be used to sum entire ZXW diagrams [13]. In this section, I extend [13] to show that W and Z give rise to two rings: controlled states and controlled matrices.

4.1 Definitions

I take my definitions from [13]. Around the same time that paper was written, [32] introduced an alternative definition of controlled ZX diagrams. While both definitions enable addition and multiplication of diagrams, the copying lemmas of section 4.2 apply only to the definitions of [13] due to the essential role of (BZW) which is not a ZX rule.

Definition 4.1.1: Controlled Matrix (3.1 in [13])

For an arbitrary square matrix D , the controlled matrix of D is the diagram \tilde{D} such that

$$\begin{aligned} \begin{array}{c} \bullet \\ | \\ \boxed{\tilde{D}} \\ | \\ \vdots \end{array} &= \begin{array}{c} \vdots \end{array} \\ \begin{array}{c} \bullet \\ | \\ \boxed{\tilde{D}} \\ | \\ \vdots \end{array} &= \begin{array}{c} \pi \\ | \\ \boxed{D} \\ | \\ \vdots \end{array} \end{aligned} \quad (4.1)$$

Some examples of controlled matrices:

4.1. DEFINITIONS

- $\begin{array}{c} | \\ \boxed{\tilde{X}} \\ | \end{array} = \begin{array}{c} | \\ \bullet \\ | \end{array}.$

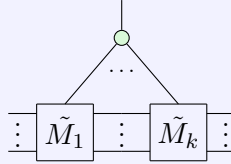
- $\begin{array}{c} | \\ \boxed{\tilde{Z}} \\ | \end{array} = \begin{array}{c} | \\ \text{yellow square} \bullet \text{yellow square} \\ | \end{array}.$

- $\begin{array}{c} | \\ \boxed{a} \\ | \end{array}$ is a very important special case of controlled scalars \tilde{a} .

Note it is crucial that D be square for it to be replaced with identity wires. One advantage of adding an extra wire to D in this way is that it gives new ways for diagrams to be composed diagrammatically. In particular, it gives a very natural way of expressing the sum and product of matrices.

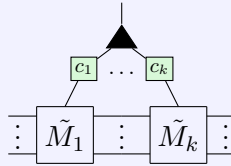
Proposition 4.1.2: 3.3 in [13]

Given controlled matrices $\tilde{M}_1, \dots, \tilde{M}_k$, the controlled matrix $\widetilde{\prod_i \tilde{M}_i}$ is given by



Proposition 4.1.3: 3.4 in [13]

Given controlled matrices $\tilde{M}_1, \dots, \tilde{M}_k$ and complex numbers c_1, \dots, c_k , the controlled matrix $\widetilde{\sum_i c_i \tilde{M}_i}$ is given by



Since the above definition does not apply to non-square matrices, it is natural to provide an alternative definition for states.

Definition 4.1.4: Controlled State (3.2 in [13])

For an arbitrary state ψ , the controlled state of ψ is the diagram $\tilde{\psi}$ such that:

$$\begin{aligned} \text{Diagram 1} &= \text{Diagram 2} \\ \text{Diagram 3} &= \text{Diagram 4} \end{aligned} \tag{4.2}$$

The diagrams in (4.2) are as follows:
 Diagram 1: A triangle labeled $\tilde{\psi}$ with a red dot on its top input line and three output lines labeled \dots .
 Diagram 2: Two red dots on two separate input lines labeled \dots .
 Diagram 3: A triangle labeled $\tilde{\psi}$ with a red circle containing π on its top input line and three output lines labeled \dots .
 Diagram 4: A triangle labeled ψ with three input lines labeled \dots and three output lines labeled \dots .

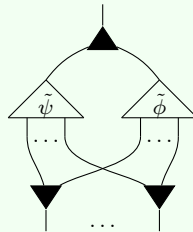
Some examples of controlled states:

- W is controlled EPR: $\blacktriangle = |\widetilde{01}\rangle + |\widetilde{10}\rangle$
- By (K0), $\text{green dot} = |\widetilde{1..1}\rangle$

Just as with matrices, we can combine two controlled states using a W node to produce a controlled sum. To preserve the number of outputs, \blacktriangledown 's are appended to the bottom so that (4.2) is still satisfied.

Definition 4.1.5

Given two controlled states $\tilde{\psi}, \tilde{\phi}$, the controlled state $\tilde{\psi} \boxplus \tilde{\phi}$ is given by



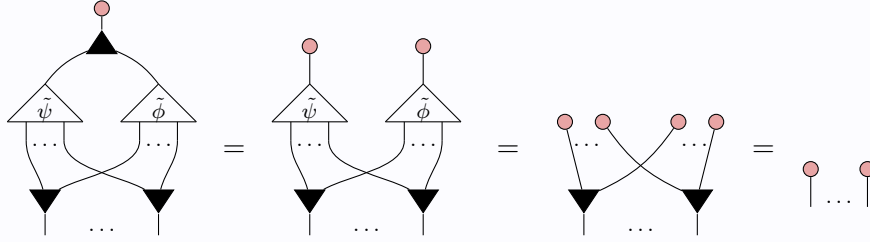
This does indeed give the controlled sum of states.

4.1. DEFINITIONS

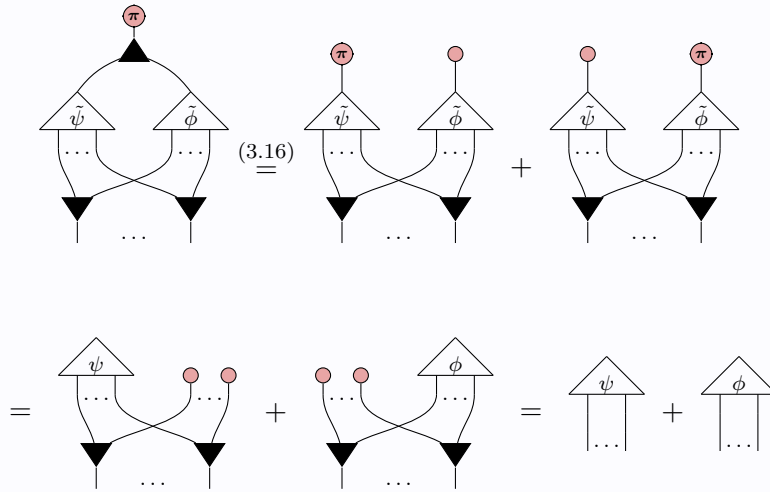
Proposition 4.1.6

$$\tilde{\psi} \boxplus \tilde{\phi} = \widetilde{\psi + \phi}$$



Proof. Firstly, $\tilde{\psi} \boxplus \tilde{\phi}$ is itself a controlled state:



Secondly, it does in fact give $\psi + \phi$ when plugged with a $|1\rangle$ since:

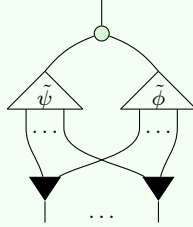


□

We can define $\tilde{\psi} \boxtimes \tilde{\phi}$ very similarly by putting a  on top rather than a .

Definition 4.1.7

Given two controlled states $\tilde{\psi}, \tilde{\phi}$, the controlled state $\tilde{\psi} \boxtimes \tilde{\phi}$ is given by



While this operation does return another controlled state, it is not obvious what this “multiplication” of states does:

(4.3)

(4.4)

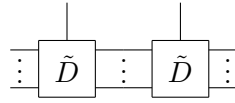
Though the top half is just $\psi \otimes \phi$, the \blacktriangledown ’s at the bottom create some entangled mess. The semantics of controlled state multiplication will be elucidated further in chapter 5.

You might complain that appending \blacktriangledown ’s along the bottom is a little ad hoc. Clearly, one important property is that is a unit. In some sense, this is the only property that matters (plus associativity). Indeed, an X spider could just have easily been chosen (as was done in [13]) and by (TA) this is equivalent. We shall be seeing many more diagrams with \blacktriangledown ’s at

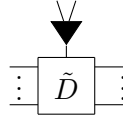
the bottom. In the right context, these gadgets turn out to be extremely powerful, as the following subsection reveals.

4.2 Copy coWs

Imagine you were promised that the top inputs to the following diagram were restricted to $\{|00\rangle, |01\rangle, |10\rangle\}$:



Then you can be sure that at most one of the \tilde{D} boxes will ever be activated. So in a sense, there is only one \tilde{D} box, but controlled by the **sum** of the inputs. But this is just:



So it appears that \blacktriangledown copies \tilde{D} ! To properly define this notion of there being no $|11\rangle$, our old friend W comes to the rescue.

Definition 4.2.1: Single Particle Subspace

A diagram D is said to be restricted to the single particle subspace (SPS) if it can be rewritten as:

$$\begin{array}{c} \text{---} \\ | \\ \boxed{D} \\ | \\ \dots \end{array} = \begin{array}{c} \text{---} \\ \blacktriangledown \\ \swarrow \quad \searrow \\ \text{---} \quad \text{---} \\ | \quad | \\ \boxed{D_1} \quad \boxed{D_2} \quad \boxed{D_3} \\ | \quad | \quad | \\ \dots \quad \dots \end{array} \quad (4.5)$$

For some diagrams D_1, D_2, D_3 , where D_2 should not be disconnected.

SPS diagrams can also have multiple inputs which can be bent into outputs

in order to satisfy (4.5). The idea is that W produces at most one $|1\rangle$ and the Z spiders simply copy the outputs and take them elsewhere. An important example of an SPS diagram is (TA) which effectively says that under SPS,

$$\blacktriangle = \text{red circle}.$$

We are now ready for a very nifty result: under SPS, \blacktriangledown copies arbitrary controlled diagrams. The proof essentially translates the initial observation from above into sums.

Lemma 4.2.2: Copying Lemma I

For any square matrix D ,

$$\begin{array}{c} \text{Diagram 1} \\ \text{[Green circle} \rightarrow \text{Black triangle} \rightarrow \text{Green circle]} \\ \text{[Box } \tilde{D} \text{]} \end{array} = \begin{array}{c} \text{Diagram 2} \\ \text{[Green circle} \rightarrow \text{Black triangle} \leftarrow \text{Green circle]} \\ \text{[Box } \tilde{D} \text{]} \end{array} \quad (4.6)$$

Proof. First of all, using (BZW) we can rewrite the LHS to

$$\begin{array}{c} \text{Diagram 1} \\ \text{[Green circle} \rightarrow \text{Black triangle} \rightarrow \text{Green circle]} \\ \text{[Box } \tilde{D} \text{]} \end{array} \stackrel{(BZW)}{=} \begin{array}{c} \text{Diagram 3} \\ \text{[Green circle} \rightarrow \text{Black triangle]} \\ \text{[Box } \tilde{D} \text{]} \end{array}$$

Then clearly

$$\begin{array}{c} \text{Diagram 3} \\ \text{[Green circle} \rightarrow \text{Black triangle]} \\ \text{[Box } \tilde{D} \text{]} \end{array} = \begin{array}{c} \text{Diagram 4} \\ \text{[Red circle} \rightarrow \text{Red circle]} \\ \text{[Box } \tilde{D} \text{]} \end{array} = \begin{array}{c} \text{Diagram 2} \\ \text{[Green circle} \rightarrow \text{Black triangle} \leftarrow \text{Green circle]} \\ \text{[Box } \tilde{D} \text{]} \end{array}$$

Meanwhile,

The diagram shows two rows of equations. The top row starts with a diagram where a red circle labeled π is connected to a black triangle, which then branches into two green circles. Each green circle is connected to a box labeled \tilde{D} . This is followed by an equals sign, a label (3.16), and another equals sign. The middle part of the top row shows two terms separated by a plus sign. The first term has a red circle labeled π connected to a green circle, which is connected to a \tilde{D} box. The second term has a red circle connected to a green circle, which is connected to a \tilde{D} box. The bottom row starts with a diagram where a red circle labeled π is connected to a green circle, which is connected to a box labeled D . This is followed by a plus sign and another diagram where a red circle is connected to a green circle, which is connected to a D box. This is followed by an equals sign, a label (3.16), and another equals sign. The middle part of the bottom row shows a diagram where a red circle labeled π is connected to a black triangle, which is connected to a D box. The final part of the bottom row shows a diagram where a red circle labeled π is connected to a green circle, which is connected to a \tilde{D} box.

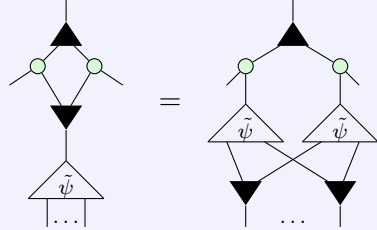
Thus the two sides are equal over the Z basis and so are equal as diagrams. \square

We have already seen a special case of this for controlled scalars in (3.20). Due to its reliance on plugging and sums, this proof is regrettably rather undiagrammatic. Typically, one can rely on completeness to guarantee that there exists a sumless, purely diagrammatic proof of every equality of ZXW diagrams. However, completeness does not apply in this case since \tilde{D} is not a concrete matrix and so this proof does not prove a specific equality, but an entire family of equalities. It would be nice to see a more diagrammatic proof of this lemma.

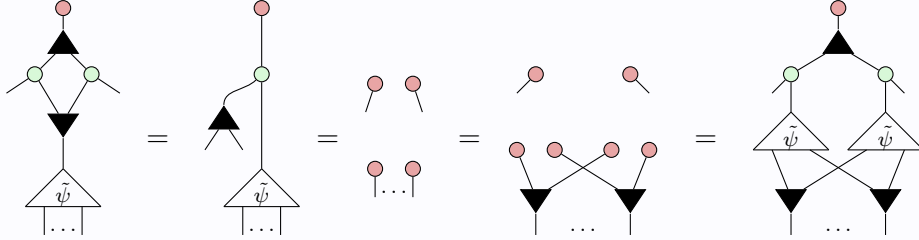
It turns out that \blacktriangledown also copies controlled states but once again appends a layer of \blacktriangledown 's at the bottom to preserve the number of outputs. This time, the interpretation of the bottom layer is that it will recursively copy any other controlled diagrams as they are plugged in.

Lemma 4.2.3: Copying Lemma II

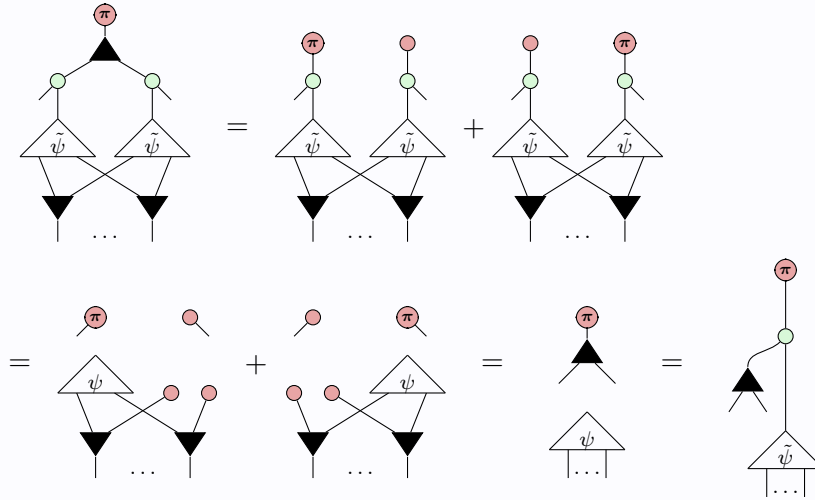
For any state ψ ,


(4.7)

Proof. As before, plugging $|0\rangle$ gives



Meanwhile, plugging $|1\rangle$ gives



Completing the proof

□

Given the no-cloning theorem (2.3.1), this is a surprising result. Of course

it does not violate no-cloning since it takes place in the restricted context of SPS and only applies to controlled states. Though the SPS restriction appears quite strong, it appears surprisingly often and so lemma 4.2.3 is an extremely useful result both in the remainder of this thesis and, hopefully, beyond.

4.3 Rings

Let \tilde{S}_n be the set of controlled n -partite states. In section 4.1, we defined an addition \boxplus and a “multiplication” \boxtimes on this set. In the spirit of compositionality, we’d like to know what happens when we plug them together. It turns out that not only does \boxtimes distribute over \boxplus , but the operations form a ring!

Theorem 4.3.1

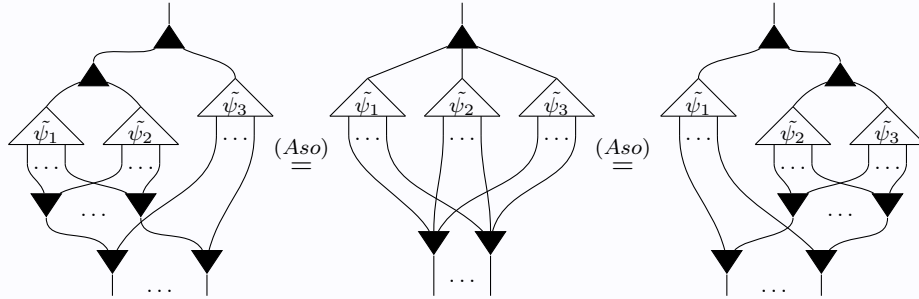
$(\tilde{S}_n, \boxplus, \boxtimes)$ defines a commutative ring

To form a ring, we must show that \boxplus defines an Abelian group, \boxtimes defines a commutative monoid and that \boxtimes distributes over \boxplus (see appendix A.3). We shall break each part of the proof into a separate lemma.

Lemma 4.3.2

$$(\tilde{\psi}_1 \boxplus \tilde{\psi}_2) \boxtimes \tilde{\psi}_3 = \tilde{\psi}_1 \boxtimes (\tilde{\psi}_2 \boxplus \tilde{\psi}_3)$$

Proof. Follows immediately from (Aso):

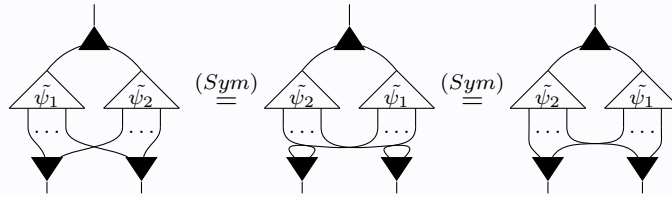


□

Lemma 4.3.3

$$\tilde{\psi}_1 \boxplus \tilde{\psi}_2 = \tilde{\psi}_2 \boxplus \tilde{\psi}_1$$

Proof. Follows from (Sym):



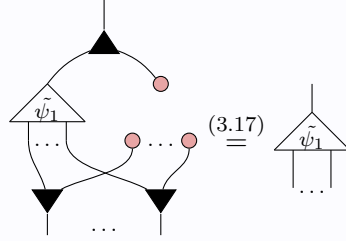
□

Lemma 4.3.4

There is an additive identity $id_+ :=$

Proof. It is clear that is the controlled state $\tilde{\mathbf{0}}$.

Then we have:



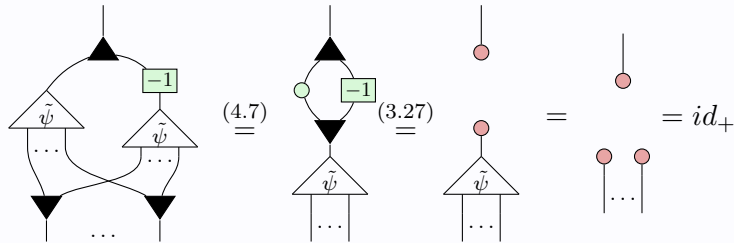
□

Finding inverses is a little more interesting, making use of the copying lemma from section 4.2.

Lemma 4.3.5

For a controlled state $\tilde{\psi}$, its additive inverse is $\tilde{\psi} \circ \boxed{-1}$

Proof. $\tilde{\psi} \circ \boxed{-1}$ is still a controlled state since $\boxed{-1}$ does nothing to \bullet .
Then $\tilde{\psi} \circ \boxed{-1}$ inverts $\tilde{\psi}$ since:



□

Thus we have shown that \boxplus defines an Abelian group. Showing \boxtimes defines a commutative monoid is very similar, except using (S1) for associativity and commutativity and using $\bullet \dots \bullet$ as the multiplicative identity.

The final component is distributivity. Again making use of the copying lemma, this proof also relies on (BZW).

Lemma 4.3.6

$$\tilde{\psi}_1 \boxtimes (\tilde{\psi}_2 \boxplus \tilde{\psi}_3) = (\tilde{\psi}_1 \boxtimes \tilde{\psi}_2) \boxplus (\tilde{\psi}_1 \boxtimes \tilde{\psi}_3)$$

Proof.

$$\begin{aligned} \tilde{\psi}_1 \boxtimes (\tilde{\psi}_2 \boxplus \tilde{\psi}_3) &= \text{Diagram 1} \stackrel{(BZW)}{=} \text{Diagram 2} \\ &= \text{Diagram 3} \stackrel{(4.7)}{=} \text{Diagram 4} \\ &= \text{Diagram 5} = \text{Diagram 6} \\ &= (\tilde{\psi}_1 \boxtimes \tilde{\psi}_2) \boxplus (\tilde{\psi}_1 \boxtimes \tilde{\psi}_3) \end{aligned}$$

The diagrams are string diagrams representing the proof of the distributive law. They use triangles to represent multiplication (pointing down) and comultiplication (pointing up), with wires labeled $\tilde{\psi}_1, \tilde{\psi}_2, \tilde{\psi}_3$. Green circles represent the multiplication and comultiplication of the tensor product. The proof proceeds through several steps: first, a braiding move (BZW) is applied; then, a more complex move (4.7) is used to rearrange the wires; finally, the result is shown to be the direct sum of the tensor products of the individual components.

□

This completes the proof of theorem 4.3.1.

Letting \tilde{M}_n be the set of controlled $2^n \times 2^n$ matrices, we also have:

Theorem 4.3.7

$(\tilde{M}_n, \blacktriangle, \circlearrowleft)$ forms a non-commutative ring.

The additive and multiplicative units are $\circlearrowleft \otimes I_n$ and $\circlearrowright \otimes I_n$, respectively. The proofs are mostly very similar to that of theorem 4.3.1, so I shall not repeat every part. I prove the two following lemmas to give the main idea.

Lemma 4.3.8

$$\begin{array}{c} \text{---} \tilde{M}_1 \text{---} \text{---} \tilde{M}_2 \text{---} \\ \text{---} \text{---} \blacktriangle \text{---} \text{---} \end{array} = \begin{array}{c} \text{---} \tilde{M}_2 \text{---} \text{---} \tilde{M}_1 \text{---} \\ \text{---} \text{---} \blacktriangle \text{---} \text{---} \end{array} \quad (4.8)$$

Proof. We prove by plugging red and commutativity of matrix addition. By definition of controlled matrices, plugging \circlearrowleft gives I_n on both sides. Meanwhile, plugging π gives:

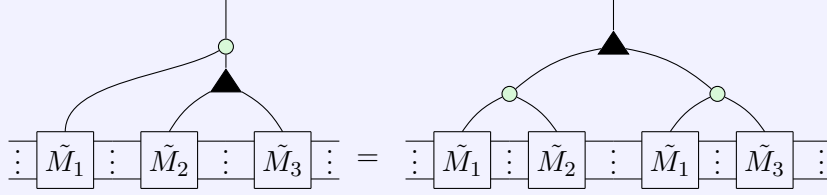
$$\begin{aligned} \begin{array}{c} \text{---} \tilde{M}_1 \text{---} \text{---} \tilde{M}_2 \text{---} \\ \text{---} \text{---} \blacktriangle \text{---} \text{---} \end{array} &\stackrel{(4.1.3)}{=} \begin{array}{c} \text{---} M_1 \text{---} \\ \text{---} \end{array} + \begin{array}{c} \text{---} M_2 \text{---} \\ \text{---} \end{array} \\ &= \begin{array}{c} \text{---} M_2 \text{---} \\ \text{---} \end{array} + \begin{array}{c} \text{---} M_1 \text{---} \\ \text{---} \end{array} \stackrel{(4.1.3)}{=} \begin{array}{c} \text{---} \tilde{M}_2 \text{---} \text{---} \tilde{M}_1 \text{---} \\ \text{---} \text{---} \blacktriangle \text{---} \text{---} \end{array} \end{aligned}$$

□

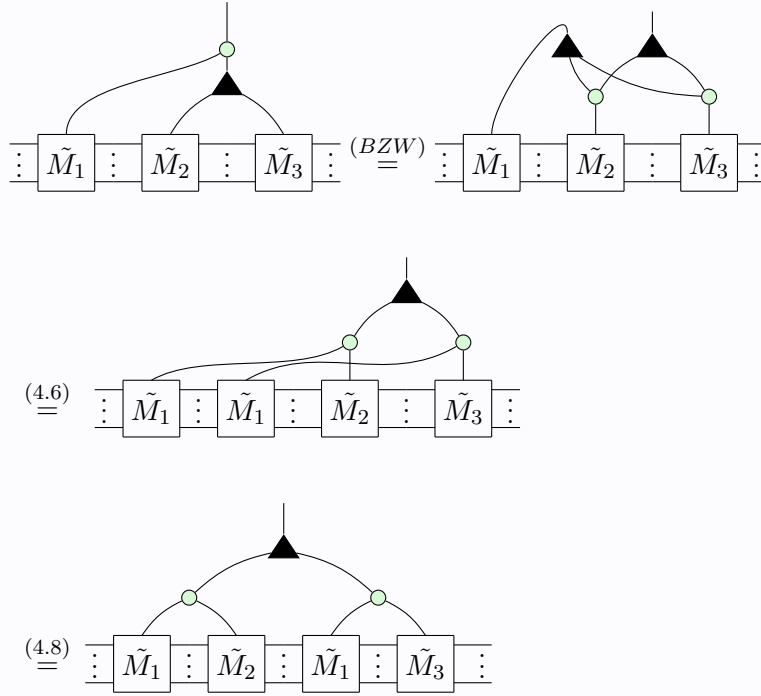
Note that commutativity of addition does not immediately follow from (Sym) unless M_1, M_2 commute. Instead, it relies on the fact that the W node can only activate at most one of the two matrices which will of course commute with the identity. Since the Z node can activate both matrices, commutativity of multiplication only holds when M_1, M_2 commute.

Finally, the proof of distributivity again relies on the copying lemma and (BZW).

Lemma 4.3.9



Proof.



□

The remaining parts follow similarly. That both types of controlled diagrams form a ring makes them much easier to work with. As they were defined to help with the diagrammatic treatment of Hamiltonians [13], I am hopeful these results will have applications in quantum chemistry. For the purposes

of this thesis, the ring of controlled states in particular will have a big part to play in the following chapter, where we interpret a broad class of controlled states as polynomials.

5 | Polynomials

*“... one ring to bring them all, and in the diagrams
bind them”*

– JRR Tolkein [adapted]

In this section, we look at how to represent polynomials in ZXW diagrams. After looking at some examples and preliminary lemmas, the section ends with a striking isomorphism. Since this chapter is based around plugging green, diagrams should mainly be read up the page. However, I will still refer to wires on top as inputs and wires below as outputs to remain consistent with bra-ket notation.

5.1 Arithmetic in ZXW

After so many years of primary school algebra, we all feel a little uneasy when encountering expressions like $-(2x + 3(x + 2(1 - y)))$. There is something unsettling about all these nested parentheses. To ease this discomfort, we intuitively employ a little ring theory and hurriedly simplify it to $-2x - 3x - 6(1 - y) = -5x - 6 + 6y$. Much better! With a sum of products, the most normal of forms, order has been restored. I draw attention to this because it is easy to forget how deeply we yearn for a sum of products and how automatically we manipulate expressions to reach it. I mention it now as we begin to explore the ways in which ZXW diagrams yearn to be written as a sum of products of copies. We begin with a basic example. Note that some W nodes will be drawn sideways for visual convenience.

Lemma 5.1.1

$$\begin{array}{c} \text{---} \blacktriangleright \text{---} \text{---} \text{---} \text{---} \blacktriangleleft \text{---} \end{array} = \begin{array}{c} \text{---} \blacktriangleleft \text{---} \text{---} \text{---} \text{---} \blacktriangleright \text{---} \end{array} \quad (5.1)$$

Proof.

$$\begin{array}{c} \text{---} \blacktriangleright \text{---} \text{---} \text{---} \text{---} \blacktriangleleft \text{---} \end{array} \stackrel{(BZW)}{=} \begin{array}{c} \text{---} \blacktriangleleft \text{---} \text{---} \text{---} \text{---} \blacktriangleright \text{---} \end{array} \stackrel{(WW)}{=} \begin{array}{c} \text{---} \blacktriangleleft \text{---} \text{---} \text{---} \text{---} \blacktriangleright \text{---} \end{array} \\
 \stackrel{(BZW)}{=} \begin{array}{c} \text{---} \blacktriangleleft \text{---} \text{---} \text{---} \text{---} \blacktriangleright \text{---} \end{array} \stackrel{(BZW)}{=} \begin{array}{c} \text{---} \blacktriangleleft \text{---} \text{---} \text{---} \text{---} \blacktriangleright \text{---} \end{array} = \begin{array}{c} \text{---} \blacktriangleleft \text{---} \text{---} \text{---} \text{---} \blacktriangleright \text{---} \end{array}$$

□

Clearly, the RHS of (5.1) is an SPS diagram and so each of the sideways W nodes will copy numbers. So plugging green, this lemma simply states the arithmetic identity:

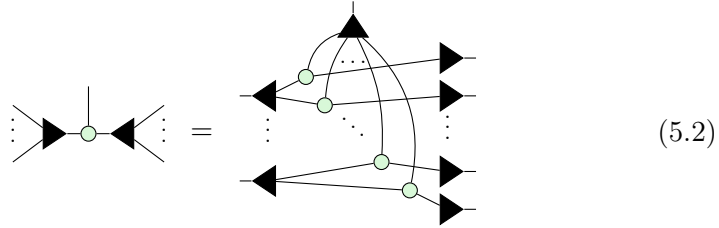
$$\begin{array}{c} \boxed{a_1} \\ \boxed{a_2} \end{array} \blacktriangleright \text{---} \text{---} \text{---} \text{---} \text{---} \blacktriangleleft \begin{array}{c} \boxed{b_1} \\ \boxed{b_2} \end{array} = \boxed{(a_1 + a_2) \times (b_1 + b_2)} \\
 = \boxed{a_1 b_1 + a_1 b_2 + a_2 b_1 + a_2 b_2} = \begin{array}{c} \boxed{a_1} \\ \boxed{a_2} \end{array} \blacktriangleleft \text{---} \text{---} \text{---} \text{---} \text{---} \blacktriangleright \begin{array}{c} \boxed{b_1} \\ \boxed{b_2} \end{array}$$

This highlights how intricately linked (BZW) is to distributivity. To emphasise this further, consider the equation below which gives the arithmetic interpretation of each individual step of the preceding proof. Note in particular that the only steps which do more than rearrange brackets are precisely

the steps that use (BZW) in the proof.

$$\begin{aligned}
 (a_1 + a_2) \times (b_1 + b_2) &\stackrel{(BZW)}{=} a_1 \times ((b_1 + b_2)) + a_2 \times ((b_1 + b_2)) \\
 &\stackrel{(WW)}{=} a_1 \times (b_1 + b_2) + a_2 \times (b_1 + b_2) \\
 &\stackrel{(BZW)}{=} (a_1 b_1 + a_1 b_2) + a_2 \times (b_1 + b_2) \\
 &\stackrel{(BZW)}{=} a_1 b_1 + a_1 b_2 + (a_2 b_1 + a_2 b_2) \\
 &= a_1 b_1 + a_1 b_2 + a_2 b_1 + a_2 b_2
 \end{aligned}$$

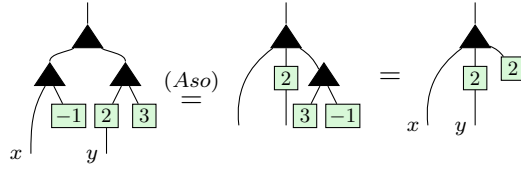
The bipartite connectivity of (5.1) naturally generalises in order to represent $(a_1 + \dots + a_n) \times (b_1 + \dots + b_m) = a_1 b_1 + a_1 b_2 + \dots + a_n b_{m-1} + a_n b_m$:



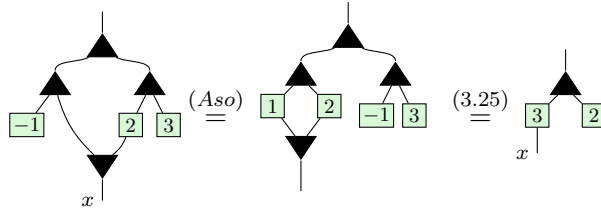
Of course, always plugging in numbers just leaves us with the complex field $\left(\boxed{a}, \blacktriangle, \circ \right)$. But things get more interesting when we leave some wires unplugged. Consider the following diagrams. Reading bottom-up, the first subtracts 1 from its input, while the second multiplies its input by 2 then adds 3.



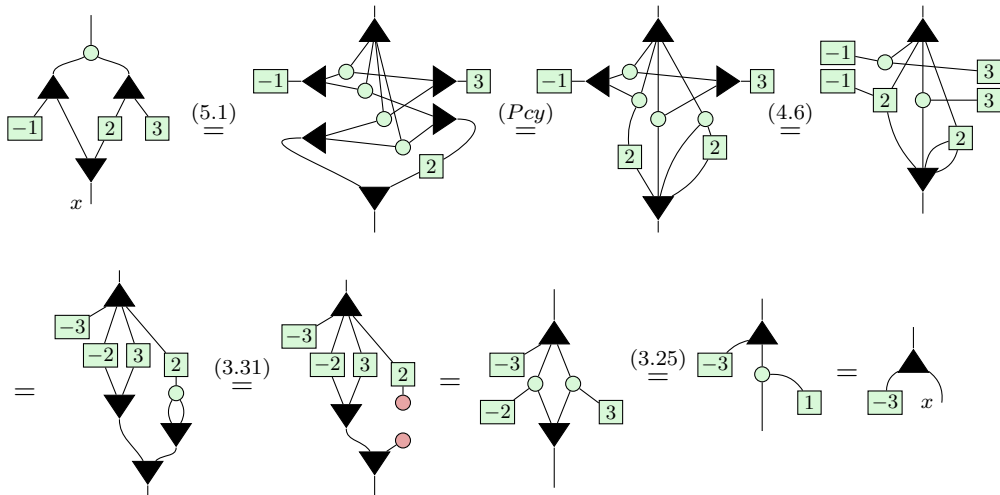
Rather than plugging numbers into the bottom, what if instead we added them from the top?



we'll bring back our old friend ∇ .



polynomials! Next, see what happens when we try to multiply them:



is clear that (3.31) is directly responsible for cancelling the x^2 term, which

otherwise would have picked up a coefficient of 2. Other than that, we seem to have come very close to polynomial arithmetic!

One justification for this failure to represent quadratic terms and higher is that it would require an operation $\boxed{x^2} :: \boxed{a} \mapsto \boxed{a^2}, \forall a$. This is clearly non-linear so cannot be performed by any matrix. However, we have also seen that copying is non-linear and yet can still occur in the restricted context of SPS. Moreover, qudit ZXW diagrams seems capable of encoding powers of x up to x^{d-1} . Another justification is that restricting to multilinear polynomials leaves exactly 2^n degrees of freedom, which coincides perfectly with the number of coefficients in an n -partite (controlled) state! The following section shows this is more than a coincidence.

5.2 Normal Forms

We have seen that certain ZXW diagrams seem to correspond to polynomials. To help formalise this, I introduce the following definition:

Definition 5.2.1: Arithmetic Diagram

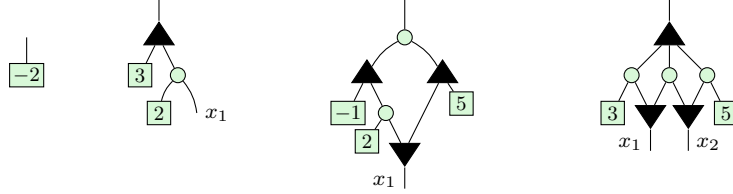
A ZXW diagram with a single input on top is **arithmetic** if it contains only $|$, \times wires, \blacktriangle , \circ , \blacktriangledown nodes and \boxed{a} boxes.



To interpret an arithmetic ZXW diagram as an arithmetic expression, read \blacktriangle as $+$, \circ as \times , \boxed{a} as the number a , \blacktriangledown as copy and output/bottom wires as variables x_1, \dots, x_n numbered from left to right. While “algebraic diagram” would have perhaps been better suited, this name has already been used for different purposes in [33]. We’ll see in section 6.2.1 that arithmetic ZXW diagrams can be interpreted as something called arithmetic circuits from complexity theory.

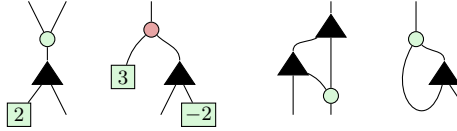
The following are all examples of arithmetic diagrams, respectively repre-


sending:

$$-2, \quad 2x_1 + 3, \quad (-1 + 2x_1)(x_1 + 5), \quad 3x_1 + x_1x_2 + 5x_2$$



The following are not arithmetic diagrams. The first has more than one input, the second contains a , the third contains a  and the fourth contains a cup.









Note the definition applies only to syntactic expressions, not what they may happen to be equal to. So an arithmetic diagram may be rewritten to a non-arithmetic diagram or vice versa, for example by applying (TA). A basic observation about arithmetic diagrams is that they copy  and thus are a type of controlled state.

Lemma 5.2.2

For any arithmetic diagram A ,

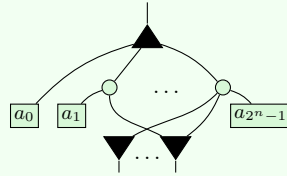
$$\begin{array}{c} \bullet \\ | \\ \boxed{A} \\ | \dots | \end{array} = \begin{array}{c} \bullet \quad \bullet \\ | \quad | \\ \dots \end{array}$$

Proof. By definition, other than wires A contains only , , , and . All 's can be removed with (Ept). Meanwhile all the spiders copy  due to (Bs0, K0, 3.17) respectively. \square

Formally, an arithmetic expression involving sums and products of scalars and indeterminates is called a polynomial (see Appendix A.3.1). As mentioned at the beginning of this chapter, we typically represent polynomials in normal form as a sum of products. By analogy, we define the following:

Definition 5.2.3: PNF

An n -output arithmetic diagram is said to be written in **polynomial form** (PNF) if it looks like:



This normal form is very closely related to the completeness normal form (CoNF, see Definition 3.2.1). Simply applying (TA) to the \blacktriangledown s at the bottom of a PNF and fusing the number boxes gives a CoNF diagram. The reason I introduce the definition of a PNF is that it is an arithmetic diagram and therefore has an immediate arithmetic interpretation. The completeness normal form however, originally coming from the ZW calculus [31], has a different interpretation based on summing over red basis states. It is something of a coincidence that the normal form of an arithmetic diagram has such striking resemblance to a normal form from almost 10 years ago.

The wiring for the coefficients a_k of a PNF follows a particular pattern. Coefficient a_k will be connected to input i iff the i th bit of k 's binary expansion is 1. For example, for $n = 3$, a_5 will be connected to x_1 and x_3 since $5 = 101_2$. The reason this convention is useful is due to the following universality result:

Proposition 5.2.4

$$\begin{array}{c} \text{Diagram: A PNF node with inputs } a_0, a_1, \dots, a_{2^n-1} \text{ and one output.} \end{array} = \begin{bmatrix} 1 & a_0 \\ 0 & a_1 \\ \dots & \dots \\ 0 & a_{2^n-1} \end{bmatrix} \quad (5.3)$$

Proof. We prove by induction on n .

For the base case, $n = 0$. The only PNF with no outputs is a number so we have:

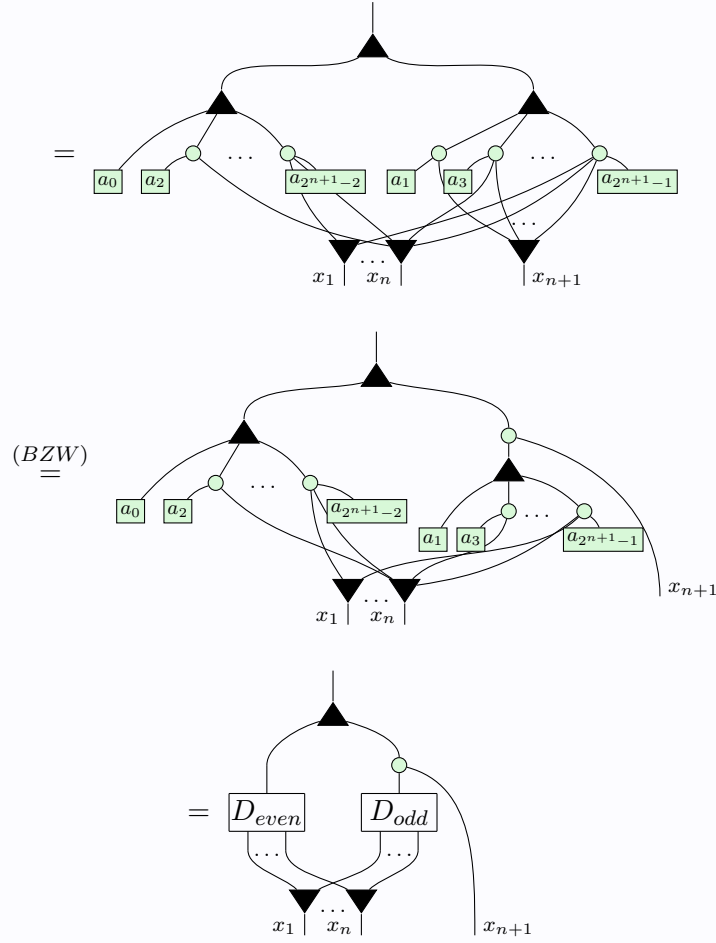
$$\boxed{a_0} = \begin{bmatrix} 1 & a_0 \end{bmatrix}$$

as desired.

For inductive hypothesis, we assume that (5.3) holds for every PNF on n outputs. We use this hypothesis to extend it to PNFs with $n + 1$ outputs.

Let D be an arbitrary PNF with $n + 1$ outputs. Firstly, observe that x_{n+1} is connected to only the odd coefficients $\{a_{2k+1}\}$ since these are exactly the indices with 1 in the least significant bit. Thus we can rewrite:

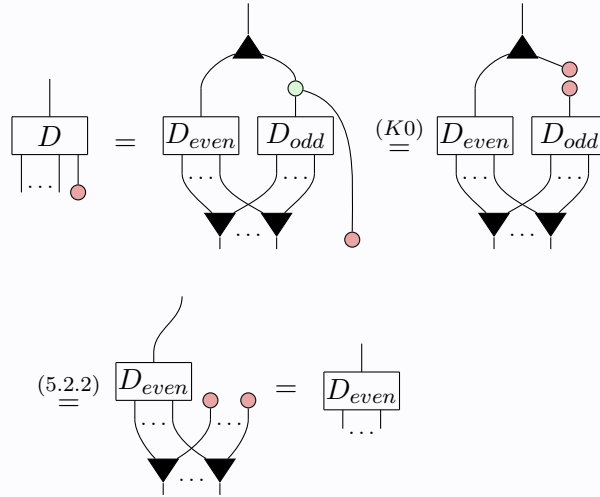
$$\begin{array}{c} \text{Diagram: A PNF node } D \text{ with multiple inputs and outputs } x_1, \dots, x_{n+1}. \end{array} = \begin{array}{c} \text{Diagram: A PNF node with inputs } a_0, a_1, \dots, a_{2^{n+1}-1} \text{ and one output.} \end{array}$$



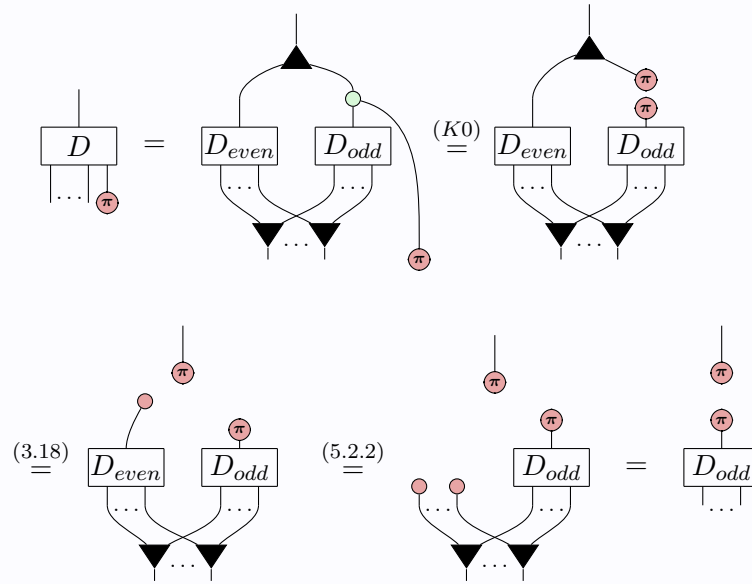
Where D_{even}, D_{odd} are PNF diagrams. Since they are over n variables, we can apply the inductive hypothesis and obtain:

$$D_{even} = \begin{bmatrix} 1 & a_0 \\ 0 & a_2 \\ \dots & \dots \\ 0 & a_{2^{n+1}-2} \end{bmatrix}, D_{odd} = \begin{bmatrix} 1 & a_1 \\ 0 & a_3 \\ \dots & \dots \\ 0 & a_{2^{n+1}-1} \end{bmatrix} \quad (*)$$

Next, plugging red we observe:



Meanwhile,



Summing these together,

$$\begin{aligned}
 \text{Diagram: } D &= \text{Diagram: } D + \text{Diagram: } D = \text{Diagram: } D_{\text{even}} + \text{Diagram: } D_{\text{odd}} \\
 &= (D_{\text{even}} \otimes |0\rangle) + (D_{\text{odd}} |1\rangle \langle 1| \otimes |1\rangle) \\
 &\stackrel{(*)}{=} \begin{bmatrix} 1 & a_0 \\ 0 & a_2 \\ \dots & \dots \\ 0 & a_{2^{n+1}-2} \end{bmatrix} \otimes |0\rangle + \begin{bmatrix} 0 & a_1 \\ 0 & a_3 \\ \dots & \dots \\ 0 & a_{2^{n+1}-1} \end{bmatrix} \otimes |1\rangle \\
 &= \begin{bmatrix} 1 & a_0 \\ 0 & 0 \\ 0 & a_2 \\ 0 & 0 \\ \dots & \dots \\ 0 & a_{2^{n+1}-2} \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & a_1 \\ 0 & 0 \\ 0 & a_3 \\ \dots & \dots \\ 0 & 0 \\ 0 & a_{2^{n+1}-1} \end{bmatrix} = \begin{bmatrix} 1 & a_0 \\ 0 & a_1 \\ 0 & a_2 \\ 0 & a_3 \\ \dots & \dots \\ 0 & a_{2^{n+1}-2} \\ 0 & a_{2^{n+1}-1} \end{bmatrix}
 \end{aligned}$$

Completing the inductive step. \square

This also shows that PNF can in fact represent any controlled state. We could have proven this indirectly by rewriting PNF to CoNF from definition 3.2.1 using (TA). However, I chose to include this proof as the PNF has a more immediate arithmetic interpretation. In fact, the idea behind the induction of this proof is to use distributivity, rewriting $a_0 + a_1x_2 + a_2x_1 + a_3x_1x_2 = a_0 + a_2x_1 + (a_1 + a_3x_1)x_2$. The only reason this was not done explicitly is because that would invoke theorem 5.4.1, whose proof relies on



the very result we were trying to show!




We have still yet to formally establish that other diagrams can actually be rewritten into PNF. We shall now prove that PNF is a normal form for arithmetic diagrams. The proof takes heavy inspiration from how you would intuitively simplify arithmetic expressions: expand brackets then collect terms. The only difference is some additional book-keeping for copying. Since a PNF always has 2^n Z boxes, the writing process is very inefficient.

Proposition 5.2.5


All arithmetic diagrams can be written into PNF

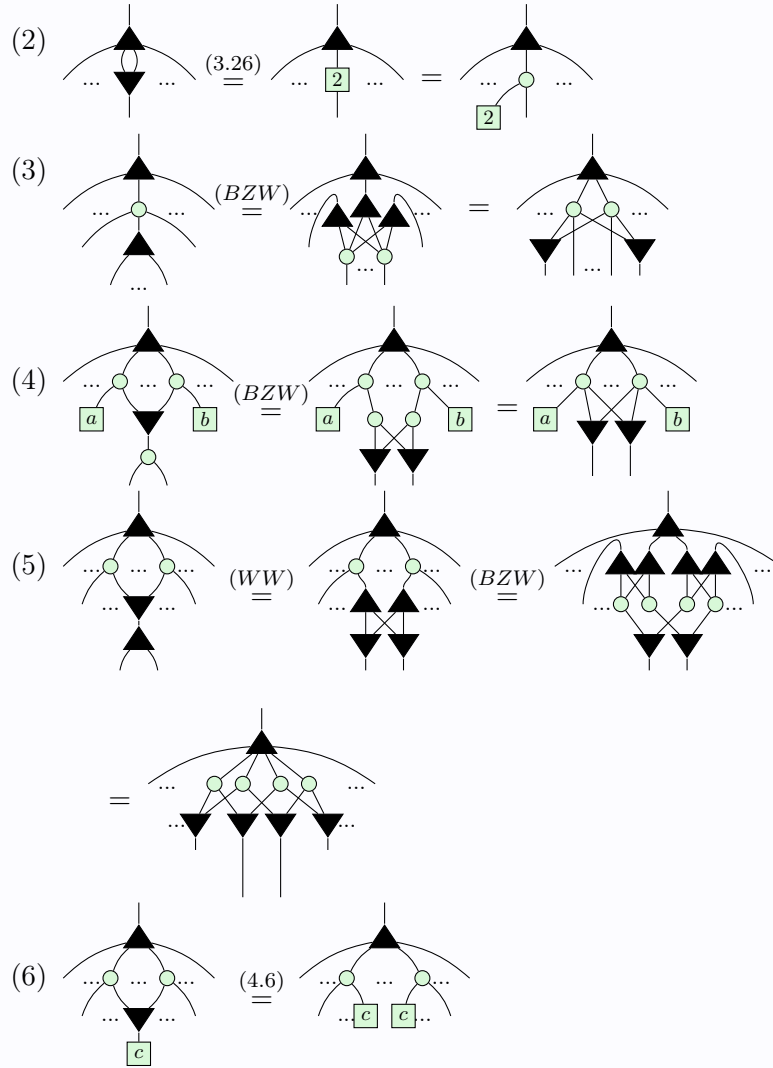
Proof. Let A be an arithmetic diagram. If $A = \boxed{a}$, we are done.

Otherwise, A has at least one output. First, we shall rewrite A into three layers, consisting of: (1) a single W at the top, (2) a layer of  and (3) a layer of \boxed{a} 's and 's. Then we shall collect terms and order the boxes to produce a PNF.

If the top of A is not already , it must be . It cannot be \boxed{a} since the remaining arithmetic diagram would then have no inputs which is impossible. It cannot be  since there is only one input and arithmetic diagrams cannot contain \cap . Thus we can rewrite:

$$(1) \quad \begin{array}{c} \text{green circle} \\ \vdots \end{array} \quad \stackrel{(3.17)}{=} \quad \begin{array}{c} \text{black triangle} \\ \vdots \end{array} \quad \begin{array}{c} \boxed{0} \\ \vdots \end{array} \quad \begin{array}{c} \text{green circle} \\ \vdots \end{array}$$

(1) guarantees there is a W at the top. We shall now repeatedly apply rewrites underneath the W until there are exactly three layers. Assume that fusion is applied as much as possible between each stage and (3.31) is applied and simplified with (K0) to remove  whenever possible. Then for as long as there are at least 4 layers, we can apply one of the following rewrites:



Clearly, we can only stop applying these rules once A is a sum of products of copies. Steps (2) and (3) ensure the top of A has such a structure and steps (4) - (6) ensure that there is nothing beneath the \blacktriangledown 's. To see that this will always terminate, observe that (2) and (3) preserve the depth of A while (4), (5), (6) all decrease it. (2) and (3) can only be applied a finite number of times before another simplification must be used. So repeatedly applying these rewrites must eventually shrink the depth down to 3, as desired. Finally, to put A in PNF we must:

(7) Collect terms: whenever there are two boxes connected to exactly the same set of \blacktriangledown 's, use (3.29) to fuse them together.

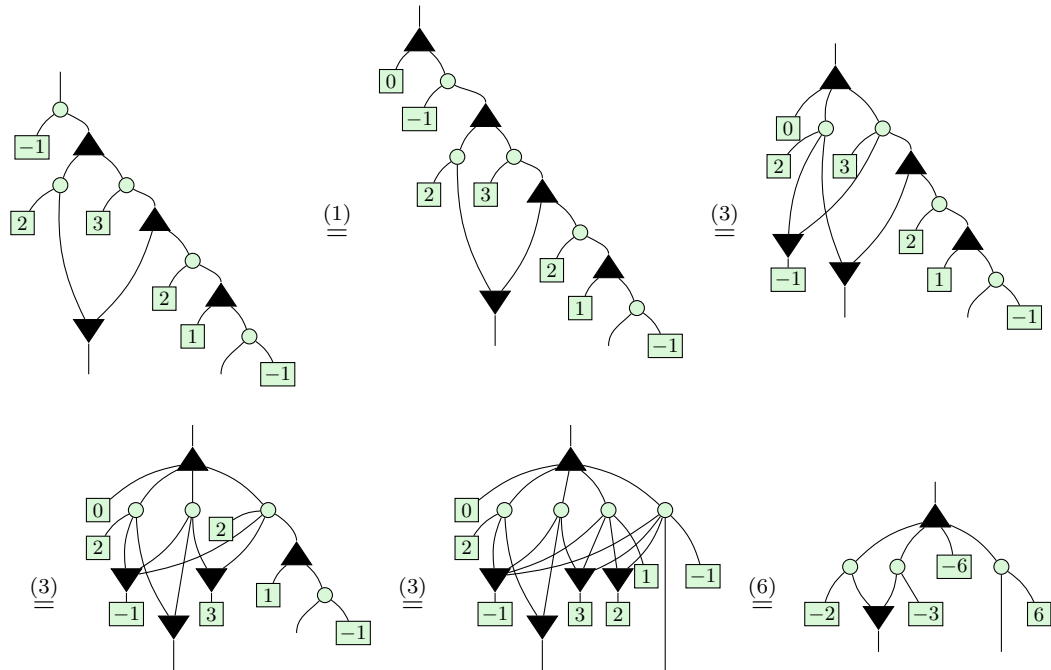
(8) Pad: use (3.23) to insert $\boxed{0}$ for any connectivities that do not exist in A .

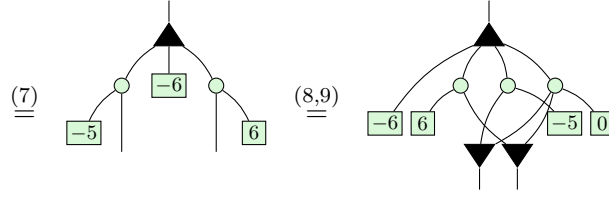
(9) Reorder: use (Sym) to reorder coefficients into the canonical order.

Step (7) ensures that every $\begin{array}{c} \circ \\ \vdots \end{array}$ has unique connectivity. Step (8) ensures there are exactly 2^n coefficients so that step (9) can order them in the appropriate way.

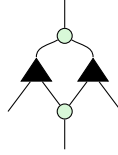
Thus A has been written in PNF, completing the proof. □

It is instructive to work through an example. The following corresponds to an abbreviated simplification of the expression from the beginning of the chapter: $-(2x_1 + 3(x_1 + 2(1 - x_2))) = -6 - 5x_1 + 6x_2$



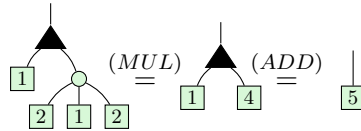


Note that the algorithm outlined in the proof is very reliant on A being an arithmetic diagram. For example, applying the same procedure to the following non-arithmetic diagram results in an infinite loop of (BZW) expansions:

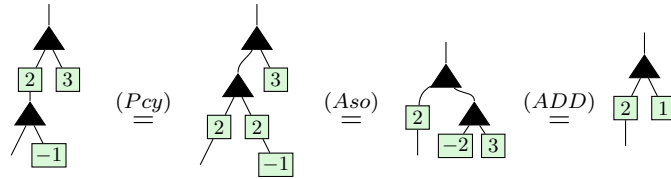


5.3 Substitution

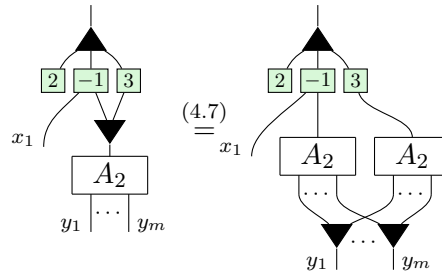
A naive interpretation of a polynomial $p \in \mathbb{C}[x_1, \dots, x_n]$ is as an **evaluation** function $ev : \mathbb{C}^n \rightarrow \mathbb{C}$. For example, if $p(x_1, x_2) = 1 + 2x_1x_2$, then $p|_{x_1=1, x_2=2} = 1 + 2(1)(2) = 5$. That we can bind variables diagrammatically follows immediately from (ADD, MUL):



However, polynomials are more general because one can also substitute variables for other polynomials. For example, plugging $y - 1$ into $2x + 3$ gives $2(y - 1) + 3 = 2y + 1$. This also holds diagrammatically.



In fact, we can see that diagrammatic substitution holds in general. Let A_1, A_2 be arithmetic diagrams. Imagine plugging A_2 into A_1 . By proposition 5.2.5, A_1 can be written into PNF which is a type of SPS diagram. By lemma 5.2.2, A_2 is a controlled state. Thus we can apply the copying lemma and push A_2 onto every term of A_1 . For example, if A_1 represents the expression $2 + 3x_2 - x_1x_2$, then for any arithmetic diagram A_2 over variables y_1, \dots, y_m :



The new \blacktriangledown 's at the bottom ensure that the A_2 's are still over the same variables and that any further substitutions can also be copied through. Moreover, a consequence of the next section is that multiplying each of A_1 's terms by A_2 does in fact correspond to multiplication of the corresponding arithmetic expressions.

5.4 Isomorphism

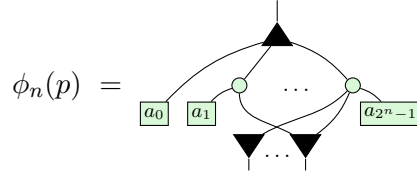
Section 5.1 suggests arithmetic diagrams have something to do with polynomials. Since (3.31) ruled out quadratic and higher powers of x , we seem to be looking at *multilinear* polynomials, i.e. elements of the ring $\mathcal{P}_n := \mathbb{C}[x_1, \dots, x_n]/(x_1^2, \dots, x_n^2)$. Lemma 5.2.2 shows that arithmetic diagrams are controlled states, which we proved in section 4.3 also form a ring $(\tilde{S}_n, \boxplus, \boxtimes)$. Now for the big reveal: these rings are *isomorphic*!

Theorem 5.4.1: Polynomial Isomorphism Theorem

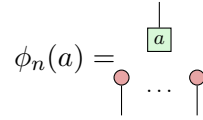
There is an isomorphism $\mathcal{P}_n \simeq \tilde{S}_n$

5.4. ISOMORPHISM

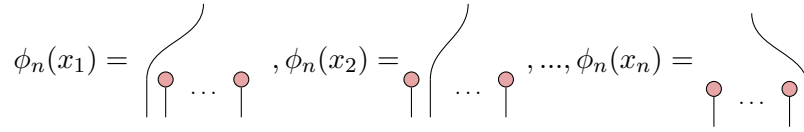
First, we shall define the map $\phi : \mathcal{P}_n \rightarrow \tilde{\mathcal{S}}_n$ before proving it induces an isomorphism. ϕ_n is defined to map an arbitrary polynomial $p(x_1, \dots, x_n) = a_0 + a_1x_1 + \dots + a_{2^n-1}x_1x_2\dots x_n$ to the following PNF:



Some important special cases are mapping scalars $a \in \mathbb{C}$:



And mapping indeterminates x_i :



We can now prove this yields an isomorphism.

Proof. First, we show ϕ_n is a homomorphism, i.e.

$$\forall p, q \in \mathcal{P}_n, \phi_n(p + q) = \phi_n(p) \boxplus \phi_n(q) \wedge \phi_n(p \times q) = \phi_n(p) \boxtimes \phi_n(q)$$

The strategy for the proof will be an induction on n .

Base case: We have not defined controlled states for $n = 0$, so the base case begins with $n = 1$. Let $p, q \in \mathcal{P}_1$. Write as $p(x_1) = a_0 + a_1x_1, q(x_1) = b_0 + b_1x_1$, where $a_0, a_1, b_0, b_1 \in \mathbb{C}$. Then since $p + q =$

$$a_0 + b_0 + (a_1 + b_1)x_1,$$

$$\phi_1(p) \boxplus \phi_1(q) = \begin{array}{c} \text{Diagram 1} \end{array} = \begin{array}{c} \text{Diagram 2} \end{array} = \begin{array}{c} \text{Diagram 3} \end{array}$$

Diagram 1: A square with nodes a_0, a_1, b_0, b_1 in green boxes. Arrows: $a_0 \rightarrow a_1$, $a_1 \rightarrow b_0$, $b_0 \rightarrow b_1$, $b_1 \rightarrow a_0$. External arrows: top (up), bottom (down), left (in), right (out).

Diagram 2: Same as Diagram 1.

Diagram 3: A square with nodes $a_0 + b_0$ (green box), a_1 (green box), b_1 (green box). Arrows: $a_0 + b_0 \rightarrow a_1$, $a_1 \rightarrow b_1$, $b_1 \rightarrow a_0 + b_0$. External arrows: top (up), bottom (down), left (in), right (out).

$$\stackrel{(3.25)}{=} \begin{array}{c} \text{Diagram 4} \end{array} = \phi_1(p + q)$$

Diagram 4: A vertical stack of two green boxes. Top box: $a_0 + b_0$. Bottom box: $a_1 + b_1$. Arrows: $a_0 + b_0 \rightarrow a_1 + b_1$. External arrows: top (up), bottom (down), left (in), right (out).

Meanwhile, since $p \times q = a_0a_1 + (a_0b_1 + a_1b_0)x_1$,

$$\phi_1(p) \boxtimes \phi_1(q) = \begin{array}{c} \text{Diagram 5} \end{array} \stackrel{(5.1)}{=} \begin{array}{c} \text{Diagram 6} \end{array}$$

Diagram 5: A square with nodes a_0, a_1, b_0, b_1 in green boxes. Arrows: $a_0 \rightarrow a_1$, $a_1 \rightarrow b_0$, $b_0 \rightarrow b_1$, $b_1 \rightarrow a_0$. External arrows: top (up), bottom (down), left (in), right (out).

Diagram 6: A square with nodes a_0, b_0, a_1, b_1 in green boxes. Arrows: $a_0 \rightarrow b_0$, $b_0 \rightarrow a_1$, $a_1 \rightarrow b_1$, $b_1 \rightarrow a_0$. External arrows: top (up), bottom (down), left (in), right (out).

$$\stackrel{(4.6)}{=} \begin{array}{c} \text{Diagram 7} \end{array} \stackrel{(Pcy)}{=} \begin{array}{c} \text{Diagram 8} \end{array}$$

Diagram 7: A square with nodes $a_0b_0, b_0, a_0, a_1, b_1$ in green boxes. Arrows: $a_0b_0 \rightarrow b_0$, $b_0 \rightarrow a_0$, $a_0 \rightarrow a_1$, $a_1 \rightarrow b_1$, $b_1 \rightarrow a_0b_0$. External arrows: top (up), bottom (down), left (in), right (out).

Diagram 8: A square with nodes $a_0b_0, a_1b_0, a_0b_1, a_1b_1$ in green boxes. Arrows: $a_0b_0 \rightarrow a_1b_0$, $a_1b_0 \rightarrow a_0b_1$, $a_0b_1 \rightarrow a_1b_1$, $a_1b_1 \rightarrow a_0b_0$. External arrows: top (up), bottom (down), left (in), right (out).

$$\stackrel{(3.31)}{=} \begin{array}{c} \text{Diagram 9} \end{array} \stackrel{(3.24)}{=} \begin{array}{c} \text{Diagram 10} \end{array} \stackrel{(3.25)}{=} \begin{array}{c} \text{Diagram 11} \end{array}$$

Diagram 9: A square with nodes $a_0b_0, a_1b_0, a_0b_1, a_1b_1$ in green boxes. Arrows: $a_0b_0 \rightarrow a_1b_0$, $a_1b_0 \rightarrow a_0b_1$, $a_0b_1 \rightarrow a_1b_1$, $a_1b_1 \rightarrow a_0b_0$. External arrows: top (up), bottom (down), left (in), right (out).

Diagram 10: A square with nodes a_0b_0, a_1b_0, a_0b_1 in green boxes. Arrows: $a_0b_0 \rightarrow a_1b_0$, $a_1b_0 \rightarrow a_0b_1$, $a_0b_1 \rightarrow a_0b_0$. External arrows: top (up), bottom (down), left (in), right (out).

Diagram 11: A vertical stack of two green boxes. Top box: a_0b_0 . Bottom box: $a_0b_1 + a_1b_0$. Arrows: $a_0b_0 \rightarrow a_0b_1 + a_1b_0$. External arrows: top (up), bottom (down), left (in), right (out).

$= \phi_1(p \times q)$

Completing the base case.

Inductive step:

Let $Hom(n)$ assert that ϕ_n is a homomorphism. Then for the inductive step we wish to prove that $\forall n, Hom(n) \implies Hom(n+1)$.

The proof relies on the recursive definition of $R[x_1, x_2] = R[x_1][x_2]$, for any ring R , to rewrite an arbitrary polynomial $p(x_1, \dots, x_{n+1}) = a_0 + a_1x_{n+1} + \dots + a_{2^{n+1}-1}x_1x_2\dots x_{n+1} \in \mathcal{P}_{n+1}$ as $p(x_{n+1}) = p_0 + p_1x_{n+1}$, where $p_0, p_1 \in \mathcal{P}_n$. This allows the p_i to be treated similarly to the scalars in the base case. To emphasise this, they will be drawn in green boxes. To help distinguish when an operation is covered by the inductive hypothesis, the wires for variables x_1, \dots, x_n will be drawn in light blue, while the x_{n+1} wires will be drawn in black. Thus the inductive hypothesis states that:

$$\begin{array}{c} \text{Diagram 1: A triangle with a black wire from the top to a box labeled 'a', and a light blue wire from the top to a box labeled 'b'. Both boxes have light blue wires to inputs } x_1, \dots, x_n. \end{array} = \begin{array}{c} \text{Diagram 2: A box labeled 'a+b' with light blue wires to inputs } x_1, \dots, x_n. \end{array} \quad (IH1)$$

$$\begin{array}{c} \text{Diagram 1: A triangle with a black wire from the top to a box labeled 'a', and a light blue wire from the top to a box labeled 'b'. Both boxes have light blue wires to inputs } x_1, \dots, x_n. \end{array} = \begin{array}{c} \text{Diagram 2: A box labeled 'a \times b' with light blue wires to inputs } x_1, \dots, x_n. \end{array} \quad (IH2)$$

Let $p(x_{n+1}) = p_0 + p_1x_{n+1}$, $q(x_{n+1}) = q_0 + q_1x_{n+1}$, where $p_0, p_1, q_0, q_1 \in \mathcal{P}_n$. Then for addition:

$$\phi_{n+1}(p) \boxplus \phi_{n+1}(q) = \begin{array}{c} \text{Diagram 1: A triangle with a black wire from the top to a box labeled 'p_0+p_1x_{n+1}', and a light blue wire from the top to a box labeled 'q_0+q_1x_{n+1}'. Both boxes have light blue wires to inputs } x_1, \dots, x_n \text{ and a black wire to input } x_{n+1}. \end{array} \stackrel{(Aso)}{=} \begin{array}{c} \text{Diagram 2: A triangle with a black wire from the top to a box labeled 'p_0+q_0', and a light blue wire from the top to a box labeled 'p_1+q_1'. Both boxes have light blue wires to inputs } x_1, \dots, x_n \text{ and a black wire to input } x_{n+1}. \end{array}$$

$$\begin{aligned}
 & \stackrel{(IH1)}{=} \text{Diagram 1} \stackrel{(BZW)}{=} \text{Diagram 2} \stackrel{(IH1)}{=} \text{Diagram 3} \\
 & = \phi_{n+1}(p_0 + q_0 + (p_1 + q_1)x_{n+1}) = \phi_{n+1}(p + q)
 \end{aligned}$$

Similarly, for multiplication:

$$\begin{aligned}
 \phi_{n+1}(p) \boxtimes \phi_{n+1}(q) &= \text{Diagram 4} \stackrel{(5.1)}{=} \text{Diagram 5} \\
 & \stackrel{(4.7)}{=} \text{Diagram 6} \stackrel{(IH2)}{=} \text{Diagram 7} \\
 & \stackrel{(BZW)}{=} \text{Diagram 8} = \text{Diagram 9}
 \end{aligned}$$

$$\begin{array}{c}
 \begin{array}{ccc}
 \begin{array}{c} (4.7) \\ \equiv \end{array} & \begin{array}{c} \text{Diagram 1} \end{array} & \begin{array}{c} (IH2) \\ \equiv \end{array} & \begin{array}{c} \text{Diagram 2} \end{array} \\
 \\
 \begin{array}{ccc}
 \begin{array}{c} (BZW) \\ \equiv \end{array} & \begin{array}{c} \text{Diagram 3} \end{array} & = & \begin{array}{c} \text{Diagram 4} \end{array} \\
 \\
 \begin{array}{ccc}
 \begin{array}{c} (3.31) \\ \equiv \end{array} & \begin{array}{c} \text{Diagram 5} \end{array} & \begin{array}{c} (5.2.2, 3.17) \\ \equiv \end{array} & \begin{array}{c} \text{Diagram 6} \end{array} \\
 \\
 \begin{array}{ccc}
 \begin{array}{c} (IH2) \\ \equiv \end{array} & \begin{array}{c} \text{Diagram 7} \end{array} & \begin{array}{c} (BZW) \\ \equiv \end{array} & \begin{array}{c} \text{Diagram 8} \end{array} & \begin{array}{c} (IH1) \\ \equiv \end{array} & \begin{array}{c} \text{Diagram 9} \end{array} \\
 \\
 = \phi_{n+1}(p_0q_0 + (p_0q_1 + p_1q_0)x_{n+1}) = \phi_{n+1}(p \times q)
 \end{array}$$

This completes the inductive step, proving that $\forall n > 1$, ϕ_n is a homo-

morphism.

Finally, to see ϕ_n is an isomorphism, we use proposition 5.2.4 to write an arbitrary controlled state in PNF:

$$\begin{bmatrix} 1 & a_0 \\ 0 & a_1 \\ \dots & \dots \\ 0 & a_{2^n-1} \end{bmatrix} = \text{Diagram}$$

Then all we have to do is interpret it as the image of a polynomial:

$$\begin{aligned} & \text{Diagram} = \text{Diagram} \\ &= \phi_n(a_0) + \phi_n(a_1 x_n) + \dots + \phi_n(a_{2^n-1} x_1 x_2 \dots x_n) \\ &= \phi_n(a_0 + a_1 x_n + \dots + a_{2^n-1} x_1 x_2 \dots x_n) \end{aligned}$$

□

A subtle detail in this proof is that the inductive step is technically extending the homomorphism $\phi_n : \mathcal{P}_n \rightarrow \tilde{S}_n$ to the homomorphism $\phi_{n+1} : \mathcal{P}_n[x_{n+1}]/(x_{n+1}^2) \rightarrow \tilde{S}_{n+1}$. Fortunately, there is a straightforward isomorphism

$$\mathcal{P}_n[x_{n+1}]/(x_{n+1}^2) \simeq \mathcal{P}_{n+1}$$

In other words, extending and taking the quotient commute. So there is nothing to worry about.

It has been known since 2011 that W adds numbers and Z multiplies them [1]. So while the lifting of numbers to polynomials may not seem particularly surprising in hindsight, this discovery was not a simple generalisation. *A priori*, it was not obvious to me that the ZXW calculus could only represent multilinear polynomials. Nor was it obvious that these polynomials were *isomorphic* (as opposed to merely homomorphic) to controlled states, thus making them universal for quantum computation. Both of these facts play essential roles in the proof of the isomorphism. I do not believe this discovery would have been possible without the invention of the ZXW calculus (in particular (TA) and its role in the copying of numbers (3.20)) , the definitions in [13] and the proof of completeness in [18], all of which happened in the last year. It is exciting to see that these were sufficient to complete a generalisation that was over a decade in the making. I am very hopeful that the ZXW calculus will continue to bring new developments to the world of diagrams, and beyond.

6 | Applications

Chapter 5 ended with a striking isomorphism between (controlled) states and multi-linear polynomials. In this chapter, I try to demonstrate the significance of this isomorphism by surveying some potential applications of identifying qubits with their polynomials. Polynomials are extremely well-studied mathematical objects so the purpose of this section is to show illustrate how they may be useful for quantum information. The selection of applications here should not be taken as comprehensive and largely reflects my own tastes and interests.

Throughout this chapter, I will treat a state $|\psi\rangle$, its controlled state $\tilde{\psi}$ and its polynomial p_ψ interchangeably. If p is a polynomial over indeterminates $X = \{x_1, \dots, x_n\}$, I will refer to p as an X -poly.

6.1 Entanglement Detection

This section investigates entangled states based on their polynomial representations. After characterising separability in terms of irreducible polynomials, I propose a novel algorithm for entanglement detection and attempt to generalise it to mixed states.

6.1.1 Separability

Recall from section 2.6 that a bipartite pure state $|\psi\rangle_{AB}$ is entangled iff it cannot be written as a product $|\psi_1\rangle_A \otimes |\psi_2\rangle_B$. Translating separability into polynomials gives a relatively simple algebraic characterisation of entanglement.

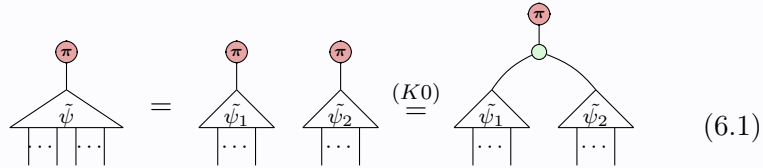
Proposition 6.1.1

Let $A = \mathbb{C}^{2^{\otimes n}}, B = \mathbb{C}^{2^{\otimes m}}$. Let $X = \{x_1, \dots, x_n\}, Y = \{y_1, \dots, y_m\}$ be the corresponding indeterminates. Let $|\psi\rangle_{AB}$ be a bi-partite pure state.

Then $|\psi\rangle$ is separable iff each of the irreducible factors of p_ψ (over $\mathbb{C}[x_1, \dots, x_n, y_1, \dots, y_m]$) is an X -poly or a Y -poly.

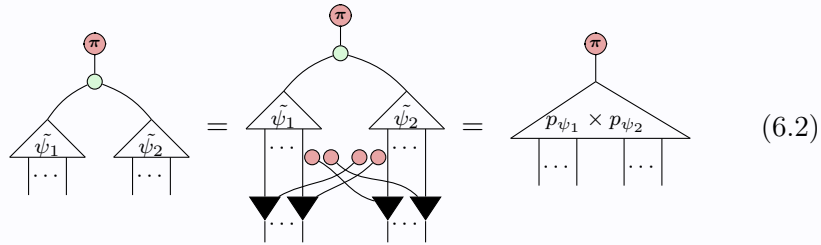
Proof. \Rightarrow :

Suppose $|\psi\rangle_{AB}$ is separable, i.e. $|\psi\rangle_{AB} = |\psi_1\rangle_A \otimes |\psi_2\rangle_B$. Then we can plug π into the respective controlled states to yield:



$$\text{Diagram (6.1): } \triangle_{\tilde{\psi}}^{\pi} = \triangle_{\tilde{\psi}_1}^{\pi} \triangle_{\tilde{\psi}_2}^{\pi} \stackrel{(K0)}{=} \triangle_{\tilde{\psi}_1}^{\pi} \triangle_{\tilde{\psi}_2}^{\pi} \quad (6.1)$$

We can now interpret the RHS as a multiplication of polynomials:




$$\text{Diagram (6.2): } \triangle_{\tilde{\psi}_1}^{\pi} \triangle_{\tilde{\psi}_2}^{\pi} = \triangle_{\tilde{\psi}_1}^{\pi} \triangle_{\tilde{\psi}_2}^{\pi} \stackrel{(K0)}{=} \triangle_{p_{\psi_1} \times p_{\psi_2}}^{\pi} \quad (6.2)$$

Thus, $p_\psi(x_1, \dots, x_n, y_1, \dots, y_m) = p_{\psi_1}(x_1, \dots, x_n) \times p_{\psi_2}(y_1, \dots, y_m)$ and so p_ψ can be factored as a product of an X -poly and a Y -poly. Regardless of how p_{ψ_1}, p_{ψ_2} themselves can be factored, there will be no irreducible factor of p_ψ containing both x_i and y_j variables.

\Leftarrow : Suppose p_ψ can be factored into $a \times p_1 \times \dots \times p_k$ for $a \in \mathbb{C}, p_1, \dots, p_k$ irreducible polynomials. Since $\mathbb{C}[x_1, \dots, x_n, y_1, \dots, y_m]$ is a unique factorisation domain (UFD), this factorisation is unique up to units and the order of the p_i 's.

By assumption, each of the p_i is either an X -poly or a Y -poly. Let p_{ψ_1} be the product of the X -polys and p_{ψ_2} be the product of the Y -polys. Then read equations (6.2, 6.1) backwards to find that ψ separates.

□

That  performs the tensor product of controlled states was also observed in the proof of ZXW completeness [18]. The innovation of this proof is to further translate this as a product of polynomials over disjoint variables. It is extremely fortunate that this factorisation occurs in $\mathbb{C}[x_1, \dots, x_n, y_1, \dots, y_m]$ rather than $\mathbb{C}[x_1, \dots, x_n, y_1, \dots, y_m]/(x_1^2, \dots, y_m^2)$. While the former is a UFD, the latter is not. In fact, the quotient ring is not even an integral domain since, for example, $x_1 \times x_1 = x_1^2 = 0$. This means that factorisation is not well-defined, let alone uniquely. On the other hand, unique factorisation is well defined in $\mathbb{C}[x_1, \dots, x_n, y_1, \dots, y_m]$ and in fact there exists efficient polynomial-time algorithms for performing it (see [34] for a survey). Thus, proposition 6.1.1 suggests a novel algorithm for deciding whether a state ψ is entangled, given its entire state-vector:

1. Translate ψ into its polynomial p_ψ .
2. Factor p_ψ .
3. Ensure none of the factors depend on both x_i and y_j for some i, j .

The bottleneck for this algorithm is the factorisation step. This can be done in time polynomial in 2^{n+m} - the length of ψ 's state vector. It is interesting to consider how this factorisation could be performed with a more compact representation of p_ψ . Some preliminary numerical results on the performance of this algorithm are included in the code supplement https://anonymous.4open.science/r/thesis_files-7C0B/entanglement_algs.ipynb. It appears this method scales slightly better than computing the Schmidt number. However, both methods are inefficient since they rely on the entire state vec-

tor of ψ .

6.1.2 Mixed States

Mixed states can also be entangled. To see what the polynomials of mixed states look like, we must first represent them diagrammatically. As outlined in section 2.7, mixed states are typically represented using density matrices. Diagrammatically, mixed states are represented using a technique called **doubling**: $|\psi\rangle \mapsto |\psi^\dagger\rangle \otimes |\psi\rangle, U \mapsto U^\dagger \otimes U$ [28]. The wires and boxes for doubled processes are drawn as thicker versions of their undoubled counterparts. So:

$$\text{thick wire} = \text{thin wire} \otimes \text{thin wire}$$

A doubled state ψ is called $\hat{\psi}$:

$$\hat{\psi} = \psi \otimes \psi$$

Similarly, for $\hat{f} = \text{double}(f)$:

$$\hat{f} = f \otimes f$$

Then composition is defined to respect the wiring:

$$\begin{array}{c} \text{---} \\ | \\ \boxed{\hat{g}} \\ | \\ \boxed{\hat{f}} \\ | \end{array} = \begin{array}{cc} \text{---} & \text{---} \\ | & | \\ \boxed{g} & \boxed{g} \\ | & | \\ \boxed{f} & \boxed{f} \\ | & | \end{array}$$

$$\begin{array}{c} \text{---} \\ | \\ \boxed{\hat{f}} \\ | \end{array} \begin{array}{c} \text{---} \\ | \\ \boxed{\hat{g}} \\ | \end{array} = \begin{array}{cc} \text{---} & \text{---} \\ | & | \\ \boxed{f} & \boxed{f} \\ | & | \end{array} \begin{array}{cc} \text{---} & \text{---} \\ | & | \\ \boxed{g} & \boxed{g} \\ | & | \end{array}$$

Though doubling preserves horizontal and vertical composition, it very crucially does not preserve sums. While a doubled sum corresponds to a superposition, a sum of doubles corresponds to a classical mixture. For example:

$$\begin{aligned}
 \text{double} \left(\begin{array}{c} \textcolor{red}{\bullet} \textcolor{red}{\bullet} \\ | \quad | \end{array} + \begin{array}{c} \textcolor{red}{\pi} \textcolor{red}{\pi} \\ | \quad | \end{array} \right) &= \text{double} \left(\begin{array}{c} \frown \\ | \end{array} \right) = \text{---} = \text{---} \\
 &\neq \text{---} = \begin{array}{c} \textcolor{red}{\bullet} \textcolor{red}{\bullet} \\ | \quad | \end{array} + \begin{array}{c} \textcolor{red}{\pi} \textcolor{red}{\pi} \\ | \quad | \end{array} = \text{double} \left(\begin{array}{c} \textcolor{red}{\bullet} \textcolor{red}{\bullet} \\ | \quad | \end{array} \right) + \text{double} \left(\begin{array}{c} \textcolor{red}{\pi} \textcolor{red}{\pi} \\ | \quad | \end{array} \right)
 \end{aligned}$$

An important result concerning maps on mixed states is the following:

Proposition 6.1.2: (6.46 in [28])

All maps on mixed states are of the form:



For some linear map f .

Where $\text{---} := \text{---}$ is called the discard effect.

Due to the presence of classical mixing, the definition of separability for mixed states is expanded to include mixtures of product states [35]. More

formally, a mixed state ρ is separable iff it can be written as:

$$\rho = \sum_i p_i \rho_i^{(A)} \otimes \rho_i^{(B)}$$

For some probability distribution $\{p_i\}$. As with pure states, a state that cannot be written in a separable form is called entangled. While the algorithm outlined in section 6.1.1 can still detect mixed product states, it is therefore no longer sufficient for detecting separability. It is not at all obvious how to adapt the reducibility test to account for linear combinations of product states. It seems necessary to return to treating ψ as a vector rather than a polynomial since vector spaces are the home of linear combinations and thus the polynomial perspective may not be able to offer any further insights.

6.2 Complexity Theory

Computational complexity theory is concerned with the inherent difficulty of computational problems. It classifies problems according to their consumption of computational resources such as space, time, randomness and magic. In this section, I invoke some applications of polynomials in complexity theory to demonstrate the applicability of the polynomial-qubit isomorphism¹. In particular, I show how the polynomial isomorphism theorem opens up connections between algebraic complexity theory and quantum circuits. Ideally, I would have had more time and space to write this section, but I hope this section still indicates the wealth of possibilities that are available.

I assume familiarity with basic complexity concepts such as reductions and oracles and classes such as P, NP and BPP. For a thorough introduction, see [36]. I define some less familiar classes here.

¹Note that the use of polynomials as mathematical objects in complexity theory is completely distinct from the assumption that efficiency corresponds to polynomial time.

Definition 6.2.1: BQP [37]

$L \in \text{BQP}$ iff there exists a family of quantum circuits (Q_n) where for each n , Q_n is of size $\text{poly}(n)$ on n qubits such that:

1. $x \in L \implies \Pr[Q_n(x) = 1] \geq 2/3$
2. $x \notin L \implies \Pr[Q_n(x) = 1] \leq 1/3$

For $|x| = n$.

It is usually assumed that Q_n is built from some fixed gateset such as $\{H, T, \text{CNOT}\}$. The choice of gateset is usually not important since all known universal gatesets can simulate one another with only a polynomial overhead. BQP is considered to capture the notion of efficient quantum computation, just as BPP is considered to capture the notion of efficient probabilistic computation. It is easy to see that $\text{P} \subseteq \text{BQP} \subseteq \text{PSPACE}$. The potential advantage of quantum computers effectively depends on the conjecture $\text{P} \neq \text{BQP}$. Unfortunately, this seems very difficult to prove since it would imply $\text{P} \neq \text{PSPACE}$, perhaps the most glaring open problem of complexity theory.

Definition 6.2.2: RP

A language $L \in \text{RP}$ iff there exists a polynomial-time probabilistic Turing Machine M such that:

1. $x \in L \implies \Pr[M(x) = 1] \geq 1/2$
2. $x \notin L \implies \Pr[M(x) = 1] = 0$

Intuitively, this corresponds to a restriction of NP where acceptance requires at least half of the computational paths to accept. Recall that for a complexity class \mathcal{C} , the class $\text{co}\mathcal{C} := \{\bar{L} : L \in \mathcal{C}\}$, where \bar{L} is the set of strings **not** in L . Thus coRP corresponds to always accepting correctly and rejecting correctly with probability at least $1/2$. Note that $\text{P} \subseteq \text{RP} \cup \text{coRP} \subseteq \text{BPP} \subseteq \text{BQP}$.

Definition 6.2.3: NQP [38]

A language $L \in \text{NQP}$ iff there exists a non-deterministic polynomial time quantum Turing Machine M such that $x \in L$ iff M accepts x with nonzero probability.

Just as $P \subseteq \text{BQP}$, we of course have $\text{NP} \subseteq \text{NQP}$. Moreover, it turns out that $\text{NQP} = \text{coC=P}$, the class of counting problems where the number of accepting paths does not equal the number of rejecting paths [39].

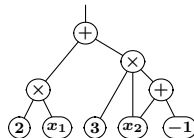
6.2.1 Algebraic Complexity

While researching potential applications of polynomials of complexity theory, I was delighted to learn there is a whole sub-field known as *algebraic complexity theory* dedicated to the study of computing with polynomials. In algebraic complexity, polynomials are not represented as a fully expanded sum of monomials since they might be exponentially sized in the number of variables. Instead, they are encoded as *arithmetic circuits*.

Definition 6.2.4: Arithmetic Circuit [40]

An arithmetic circuit C over n variables and some field \mathbb{F} is a directed acyclic graph where the leaves are labelled with constants $c \in \mathbb{F}$ or variables x_i and internal nodes labelled with $+$ or \times gates. The root of C is identified with the polynomial $p \in \mathbb{F}[x_1, \dots, x_n]$ that is computed by C .

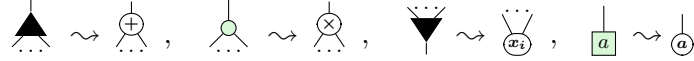
An example of an arithmetic circuit computing $2x_1 + 3x_2(x_2 - 1)$ is depicted below:



Equality of polynomials $p = q$ means that they give the same output on

all inputs. Equivalence of polynomials $p \equiv q$ means they are syntactically identical. For example, $(x_1 + 2)(x_2 - 1) = x_1x_2 + 2x_2 - x_1 - 2$ though they are not equivalent. Meanwhile $1 + 2(x_1 - 3) \equiv 1 + 2(x_1 - 3)$. The maximum number of edges leaving an internal node is called the **fan-out** and if C has fan-out 1 (i.e. is a tree), then it is called an arithmetic **formula**.

It follows immediately from theorem 5.4.1 that arithmetic ZXW diagrams can be interpreted as arithmetic circuits. Specifically, the generators can be translated as follows:



When a \blacktriangledown is above a \blacktriangle or green circle , it translates to a fan-out of the corresponding arithmetic gate. Note that only multilinear arithmetic circuits (i.e. arithmetic circuits whose corresponding polynomial is multilinear) can be translated back into arithmetic ZXW diagrams. Identifying arithmetic ZXW diagrams with arithmetic circuits means we can immediately access a wealth of results from algebraic complexity theory. For example, incorporating a technique called depth reduction [41] means that any arithmetic ZXW diagram of size n^k can be rewritten to an arithmetic ZXW diagram of size $\text{poly}(n^k)$ and depth $O(\log^2 n)$.² Another example is an efficient test for arithmetic diagram equality thanks to an algorithm for the following problem:

Definition 6.2.5: Polynomial Identity Testing (PIT) [40]

Given an arithmetic circuit C that describes a polynomial $p(x_1, \dots, x_n) \in \mathbb{F}[x_1, \dots, x_n]$, the PIT problem is to decide whether $p = 0$.

²This relies on the fact that multilinear polynomials can have degree at most n .

Where \mathbb{F} is some “sufficiently large” field. *PIT* can be used to check whether two polynomials are equal since $p = q \iff p - q = 0$. More surprising examples of problems that reduce to PIT are bipartite perfect matching in graphs and primality testing [40]. PIT has an efficient probabilistic algorithm thanks to the following lemma:

Lemma 6.2.6: Schwartz-Zippel [40]

Let $p \in \mathbb{F}[x_1, \dots, x_n]$ be a polynomial of degree $d \geq 0$. Then for any finite subset $S \subseteq \mathbb{F}$:

$$\Pr_{r_1, \dots, r_n \in S}[p(r_1, \dots, r_n) = 0] \leq \frac{d}{|S|}$$

Thus polynomials of low degree can only have very few roots. This is exploited in the Schwartz-Zippel algorithm for PIT which randomly evaluates p with points from a finite subfield of \mathbb{F} [40]. If $p = 0$, then clearly $\Pr[p(r_1, \dots, r_n) = 0] = 1$ which means $PIT \in \text{coRP}$ (i.e. $\overline{PIT} \in \text{RP}$). We shall return to this algorithm a little later.

Arithmetic circuits give rise to their own algebraic complexity classes. Two of the most important are as follows.

Definition 6.2.7: VP [42]

A family of polynomials (f_n) is in VP iff for every n , f_n has at most $\text{poly}(n)$ variables and degree and there exists an arithmetic circuit computing f_n of size $\text{poly}(n)$.

Definition 6.2.8: VNP [42]

A family of polynomials (g_n) is in VNP iff for every n , g_n has at most $\text{poly}(n)$ variables and degree and there exists some family $(f_n) \in \text{VP}$ such that

$$g_n(x_1, \dots, x_{\text{poly}(n)}) = \sum_{\vec{e} \in \{0,1\}^{\text{poly}(n)}} f_{\text{poly}(n)}(x_1, \dots, x_{\text{poly}(n)}, e_1, \dots, e_{\text{poly}(n)})$$

As the name might suggest³, VP (the class of polynomially sized polynomials) is considered the algebraic analogue of P. Meanwhile, VNP is considered the algebraic analogue of NP. Informally, VNP is the class of polynomials such that the coefficient of any given monomial can be efficiently computed. Clearly, $\text{VP} \subseteq \text{VNP}$. As with the P versus NP problem, it is widely believed but remains to be proven that $\text{VP} \neq \text{VNP}$ [42]. There is also a notion of reductions for algebraic complexity classes.

Definition 6.2.9: p -projections [42]

A polynomial $f \in \mathbb{F}[x_1, \dots, x_n]$ is a projection of $g \in \mathbb{F}[x_1, \dots, x_m]$ if there exists some assignment $\rho \in (\{x_1, \dots, x_n\} \cup \mathbb{F})^m$ such that $f(x_1, \dots, x_n) \equiv g(\rho_1, \dots, \rho_m)$ identically as polynomials.

A family of polynomials (f_n) is a p -projection of another family (g_n) if for every n , f_n is a projection of $g_{\text{poly}(n)}$.

Informally, a projection means that g can simulate f using a simple substitution of variables and constants. For example $f(x_1, x_2) = 2x_1x_2 + 4$ is a projection of $g(x_1, x_2, x_3) = x_1x_2x_3 + 2x_3$ since $g(x_1, x_2, 2) \equiv f$. A landmark result in algebraic circuit complexity is that computing the permanent of a matrix is VNP-complete (i.e. $\text{PERM} \in \text{VNP}$ and every other problem in VNP is a p -projection of PERM) [43]. Meanwhile, computing the

³The V is after Valiant, who defined them in [43].

determinant is in VP.

Since we are interpreting arithmetic ZXW diagrams as arithmetic circuits, it is natural to relate these diagrams to the complexity classes we have just introduced. Observe that by proposition 5.2.4, every quantum state is equivalent to *some* arithmetic diagram (namely, its PNF). Though the PNF is exponentially large, there are clearly a number of quantum states with much smaller arithmetic diagrams (for example, product states). I propose the following definition which aims to capture the algebraic analogue of BQP, in other words the class of polynomials corresponding to polynomially sized quantum circuits.

Definition 6.2.10: VQP

A family of polynomials (f_n) is in VQP iff for every n there exists a polynomially sized quantum circuit Q_n on n qubits such that

$$f_n = p_{Q_n|0\dots 0\rangle}$$

Intuitively, $(f_n) \in \text{VQP}$ means that f_n is the arithmetic diagram of some polynomially sized quantum circuit evaluated on $|0\dots 0\rangle$. For example, it is clear that $(\prod_{i=1}^n (1 + x_i))_n \in \text{VQP}$ since $\prod_{i=1}^n (1 + x_i)$ corresponds to $|+\dots +\rangle$ so can be prepared with n Hadamard gates. In the definition, it is necessary to run the circuit on some input in order to produce a state so that an arithmetic diagram can be found. The choice of $|0\dots 0\rangle$ is completely arbitrary. Including only states is no loss of generality due to a bending trick demonstrated in the following subsection.

An interesting question is whether $\text{VP} \stackrel{?}{=} \text{VQP}$. If so, then every quantum state has a compact polynomial representation and every arithmetic circuit has a short quantum circuit. This would open up an arsenal of algebraic complexity techniques for quantum circuit complexity. Even if $\text{VP} = \text{VQP}$

were true, however, the following section shows it is very unlikely for there to be an *efficient* way of converting between arithmetic diagrams and quantum circuits.

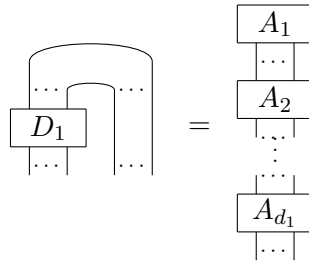
6.2.2 Proof Complexity

In this section, I show how the efficiency of algorithms relating to polynomials can be used to provide lower bounds on the length of ZXW proofs. While the bound itself is not particularly surprising, the proof demonstrates how results from distant areas of complexity theory may relate to quantum computing.

Proof complexity is a part of complexity theory that focuses on the efficiency of proof systems [44]. This includes questions like what are the upper and lower bounds on the lengths of a proof from a given system, how can one proof system reduce to another, etc. Since the ZXW calculus is a proof system, these questions also apply.

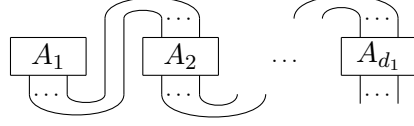
For polynomially sized ZXW diagrams on n wires, the proof of completeness [18] gives an exponential upper bound on the length of proofs of equality. I summarise their technique now, except using PNF rather than CoNF.

Suppose one wishes to show that $D_1 = D_2$. First, bend D_1 and D_2 into states, if necessary. Then write D_1 as a stack:

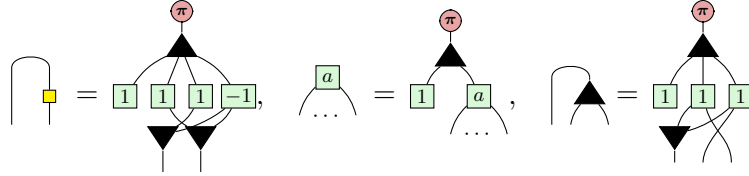


Where each A_i is a tensor product of ZXW generators and $d_1 = \text{poly}(n)$ is

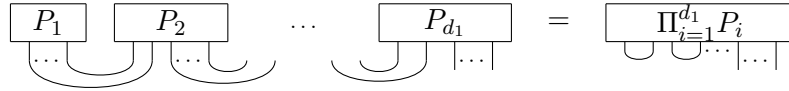
the depth of D_1 . Now bend the vertical stack of A_i 's into a horizontal train:



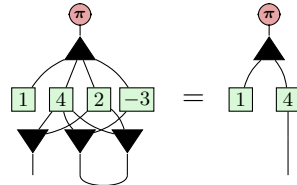
Then convert each of the bent A_i 's into a PNF. Each of the generators are mapped as follows:



Proposition 6.1.1 outlines how taking the tensor product of PNFs can be performed with polynomial multiplication (over distinct variables). Thus each bent A_i (the product of n bent generators) can be converted to some PNF P_i . So we are left with:



Finally, to compute the partial trace of a PNF between i and j (i.e. a cup between i and j), remove any Z boxes which are connected to only i or only j . On the corresponding polynomial, this means sending coefficients with only one of x_i, x_j to 0 and retaining the rest. For example:



Doing the same to D_2 , we have two PNFs that are equal iff $D_1 = D_2$.

However, since the PNF is a direct representations of the state vector (see proposition 5.2.4), both PNFs will be of size $O(2^n)$ so this process must take exponential time. This seems unnecessary, but I shall provide evidence that it is close to optimal.

A serious bottleneck is performing the polynomial multiplications corresponding to the tensor products. So one possible idea for more efficient proofs is to simply leave the polynomials in a more compact, unexpanded form. More generally, one could try and rewrite D_1 and D_2 to their nearest arithmetic diagram (which one hopes to be exponentially smaller than the corresponding PNFs). Then simply use *PIT* to check for equality. The problem with this is that there is no clear way of taking the partial trace of a product of polynomials without first expanding the product. The polynomial representation of tracing the tensor product is as follows.

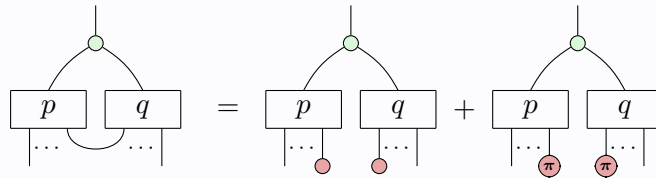
Lemma 6.2.11

Let $\rho_{x_i, x_j}(p)$ correspond to the partial trace between x_i and x_j on p . Let $p_{x_i} \in \mathbb{C}[x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n]$ be the sum of all x_i coefficients in p .

For arbitrary polynomials $p(x_1, \dots, x_n), q(y_1, \dots, y_m)$, taking the trace of the product yields:

$$\rho_{x_i, y_j}(p \times q) = p|_{x_i=0} \times q|_{y_j=0} + p_{x_i} \times q_{y_j} \quad (6.3)$$

Proof. For visual ease, we draw the case of $i = n, j = 1$:



The left part of the sum is clearly equal to $p|_{x_n=0} \times q|_{y_1=0}$. For the

right part of the sum, use the same technique as the proof of theorem 5.4.1 to write $p = a_0 + a_1 x_n, q = b_0 + b_1 y_1$, for $a_0, a_1 \in \mathcal{P}_{n-1}, b_0, b_1 \in \mathbb{C}[y_1, \dots, y_{m-1}]/(y_i^2)$. Then as in the proof of proposition 5.2.4, plugging π will yield $a_1 = p_{x_n}$ and $b_1 = q_{y_1}$, respectively. \square

While evaluating $p|_{x_i=0}$ and $q|_{y_j=0}$ can be done efficiently, since π is not a number then plugging π does not correspond to evaluation, but rather “pops out” coefficients as seen in the proof of proposition 5.2.4. So, if p is given in a compact form, it is not clear how to efficiently compute p_{x_i} . In fact, one almost certainly cannot. The ability to efficiently rewrite an arbitrary state to an arithmetic ZXW diagram would be too good to be true (even if $\text{VP} = \text{VQP}$). Not only would it allow efficient simulation of quantum circuits, it would allow a reduction from an NQP -complete problem to \overline{PIT} (which is in RP). This is highly implausible since it is generally assumed that $\text{P} \neq \text{NP} \subseteq \text{NQP}$ and yet it is considered likely that $\text{PIT} \in \text{P}$ [40]. The problem to be reduced is as follows.

Definition 6.2.12: Exact non-identity problem (ENI) [45]

Let \mathcal{H}_{in} be a Hilbert space and let $\mathbb{C}^{2^{\otimes m}}$ be an m -qubit ancilla space. Given a classical description $x \in \{0, 1\}^*$ of a quantum circuit U_x that acts on $\mathcal{H}_{in} \otimes \mathbb{C}^{2^{\otimes m}}$, the *ENI* problem decides whether

$$\exists |\psi\rangle, |(\langle\psi| \otimes \langle 0\dots 0| U'_x(|\psi\rangle \otimes |0\dots 0\rangle))^2 \neq 1$$

or

$$\forall |\psi\rangle, |(\langle\psi| \otimes \langle 0\dots 0| U'_x(|\psi\rangle \otimes |0\dots 0\rangle))^2 = 1$$

Where U'_x is a version of U_x that uncomputes the ancilla state back to $|0\rangle^{\otimes m}$.

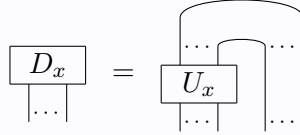
Put simply, ENI decides whether U_x is not the identity on some non-ancilla inputs. It is proven in [45] that ENI is QNP-complete. When this problem is modified to allow a circuit to be *close* to the identity, it becomes QMA-complete [46]. However, since PIT tests for exact equality we are able to reduce the stronger ENI .

Proposition 6.2.13

If all quantum states can be rewritten to an arithmetic diagram in polynomial time, then $NQP = RP$.

Proof. Suppose that every ZXW state diagram can be rewritten to an arithmetic diagram in polynomial time. We shall use this assumption to give a poly-time reduction from ENI to \overline{PIT} .

Let x be a classical description of a quantum circuit U_x over $n = \text{poly}(|x|)$ qubits. We can assume U_x is built using a fixed gateset where each gate has a constant sized ZXW diagram (for example, $H, T, CNOT$). Then we can write the following as a ZXW diagram in $\text{poly}(|x|)$ time:



Now use the assumption to rewrite D_x into an arithmetic diagram A_x . Interpret A_x as a polynomial p_x with left variables x_1, \dots, x_n and right variables y_1, \dots, y_n . The polynomial for the bent identity is represented by the linear sized arithmetic circuit $p_{\cap}^n := \prod_{i=1}^n (1 + x_i y_{n-i+1})$. So $p_x = p_{\cap}^n \iff U_x = I$ which means $\overline{PIT}(p_x - p_{\cap}^n) = 1 \iff ENI(U_x) = 1$.

This gives a polytime reduction from ENI to \overline{PIT} . Since ENI is NQP-complete and $\overline{PIT} \in RP$, this puts $NQP \subseteq RP$.

□

Note that since $\text{RP} \subseteq \text{NP} \subseteq \text{NQP}$, this collapse would further imply $\text{NP} = \text{RP}$.



Proposition 6.2.13 demonstrates the hardness of translating descriptions of states into arithmetic ZXW diagrams. For any proof of equality that relies on going through arithmetic diagrams, the result gives a (conditional) exponential lower bound. Even if $\text{VQP} \subseteq \text{VP}$, this gives strong evidence that arithmetic diagrams cannot be efficiently found from arbitrary circuits. If $\text{VQP} \not\subseteq \text{VP}$, this result would not be at all surprising since there would not be enough time to write the super-polynomially sized arithmetic diagram of an arbitrary quantum circuit.

We might also consider whether $\text{VP} \subseteq \text{VQP}$. This would imply polysize arithmetic circuits can be converted to polysize quantum circuits. While this may be possible, it surely cannot be done efficiently due to an even stronger hardness result. [19] shows that ZX circuit extraction (i.e. finding a quantum circuit for an arbitrary ZX diagram, promised that it is unitary) is $\#P$ hard. This is proved using a reduction from $\#SAT$, the problem of counting the number of solutions to a SAT instance. The proof finds a way of representing NOT and AND gates as ZX diagrams and then running the corresponding SAT instance on $|+\dots+\rangle$ to count the number of solutions. This is then used as the input of a controlled operation which guarantees the resulting circuit is unitary. It is not difficult to find the arithmetic ZXW representation of AND and NOT gates. Thus the proof can easily be modified to show the $\#P$ -hardness of converting arithmetic diagrams to quantum circuits.

Finally, what happens if $\text{VNP} \subseteq \text{VQP}$? Since $PERM \in \text{VNP}$ this would give polysize quantum circuits for $PERM$. But since $PERM$ is also $\#P$ -hard [47], this would mean $P^{\#P} \subseteq \text{BQP}$, which is highly implausible.

7 | Conclusion

In this thesis, I explored polynomial arithmetic in the ZXW calculus. Other than an almost entirely self-contained introduction to the relevant prerequisites, my main contributions were as follows:

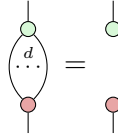
- Section 3.2.5 proved a number of new lemmas for the ZXW calculus.
- Lemmas 4.2.2 and 4.2.3 showed that under natural conditions,  copies arbitrary controlled diagrams ( would have also sufficed). Theorems 4.3.1 and 4.3.7 used the copying lemmas to show that controlled states and controlled matrices both form a ring. From such a simple definition, it is quite a surprise that controlled diagrams form such rich algebraic structures.
- Section 5.1 found that a certain fragment of ZXW diagrams could be interpreted as arithmetic circuits. Section 5.2 proved this fragment was in fact universal for controlled states, thanks to a coincidental connection with a pre-existing normal form. Section 5.4 used the arithmetic interpretation to show that controlled states are in fact isomorphic to multilinear polynomials. This is the main result of the thesis, opening up a new perspective on the ZXW calculus and quantum information as a whole.
- Chapter 6 discussed some applications of the polynomial isomorphism theorem. This culminated in a novel algorithm for entanglement detection, the definition of a new algebraic complexity class and some powerful complexity theoretic reductions based on polynomial identity testing and arithmetic circuits.

7.1 Future Directions

The bulk of this thesis was spent building up to the proof of the isomorphism theorem. In chapter 6, I tried to convey the significance of the result by exploring some applications. I did not have as much time to spend on this section as I would have liked and the choice of topics was not comprehensive. I finish by offering some thoughts on the directions I was not able to pursue.

Qudits

Diagrammatically, the most obvious future direction is qudits. In qudit ZXW calculus, Z phases are now tuples $(a_1, \dots, a_{d-1}) \in \mathbb{C}^{d-1}$ which are added and multiplied component-wise by the W and Z nodes, respectively. While many of the ZXW rules continue to hold for qudits, (3.30) generalises to:



Thus the CoNF is generalised to include up to $d - 1$ repeated connections, hence requiring d^n parameters which coincides with the number coefficients in a multivariate polynomial of individual degree $d - 1$. So one might hope that

$$\tilde{S}_n^d \simeq \mathbb{C}^{d-1}[x_1, \dots, x_n] / (x_1^d, \dots, x_n^d)$$

Where \tilde{S}_n^d is the ring of controlled qudit states. However, as shown in the code supplement (https://anonymous.4open.science/r/thesis_files-7C0B/qudits.ipynb), the multiplication is slightly more complicated than killing x^d powers and higher. I am hopeful there exists some other quotient to take care of this.

Controlled Matrices

Chapter 4 discovered two rings: controlled states and controlled matrices. Chapter 5 only gave semantics to the former. Thus it is worth considering whether the ring of controlled matrices is isomorphic to anything interesting. Since the addition and multiplication operations correspond to the control of literal matrix addition and multiplication, it is unlikely to stray too far from the realm of matrices. One hope is that they correspond to matrix polynomials, i.e. polynomials waiting to be evaluated on square matrices. Matrix polynomials can be used to simulate matrix exponentials - which are of particular importance for Hamiltonian simulation. Thus, a connection to matrix polynomials would help continue the work in [13] to give a diagrammatic treatment of quantum chemistry.

Algebraic Complexity

Recall that the obstacle to rewriting a quantum circuit to an arithmetic diagram was finding the partial trace of a product. One way of reinterpreting the p_{x_n} in equation (6.3) is using partial derivatives. Since p can be written $p = p_0 + p_{x_n} x_n$, for $p_0, p_{x_n} \in \mathcal{P}_{n-1}$, then $\partial p / \partial x_n = p_{x_n}$.

Fortunately, partial derivatives have been very well studied in the context of arithmetic circuits [48] and so I am hopeful this may simplify the arithmetisation of quantum circuits somewhat. Of course, it cannot make the conversion *efficient* due to proposition 6.2.13. However, I hope that it makes it *possible*. In that vein, I make the following conjectures about the newly defined VQP:

1. $\text{VP} = \text{VQP}$. Intuitively, small arithmetic circuits should have small quantum circuits and vice versa. Unfortunately, performing the transformation in either direction is very difficult, as argued in section 6.2.2. Nevertheless, this immediately allows for a transfer of many of the rich arithmetic circuit lower bounds [49] to quantum circuits, for whom

lower bounds are more sparse.

2. $\text{VNP} \not\subseteq \text{VQP}$. As mentioned at the end of section 6.2.2, $\text{VNP} \subseteq \text{VQP} \implies \text{P}^{\#P} \subseteq \text{BQP}$.

Since it is known that $\text{VP} \subseteq \text{VNP}$, together these imply $\text{VNP} \not\subseteq \text{VP}$. As this would resolve a long-standing open problem, I expect they'll be quite hard to prove!

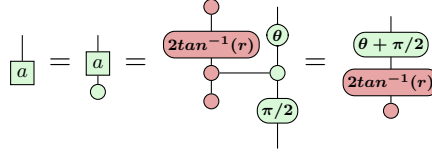
Not only does algebraic complexity theory study algorithms for computing polynomials, it also studies proof systems built out of polynomials themselves [50]. Usually, these systems use the ring axioms to show that a given system of equations is unsatisfiable. It would therefore be very interesting to interpret the ZXW calculus as a type of algebraic proof system and see whether any connections can be made with more traditional algebraic proof systems.

Circuit Optimisation

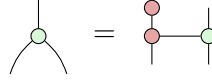
It was briefly mentioned in section 6.2.1 that depth reduction allows size s arithmetic circuits to be rewritten to depth $O(\log^2 n)$ circuits of size $\text{poly}(s)$ [41]. Suppose there were some non-trivial class of circuits S such that everything in S could be efficiently transformed to arithmetic diagrams and back again, approximately preserving size and depth. Then one could use depth reduction techniques to get efficient parallel circuits for S .

One possible candidate for S could be stabilizer states. A complete set of flow-preserving rewrite rules for stabilizer diagrams is found in [51]. Flow preservation guarantees that all equalities are between diagrams that can be deterministically executed on a quantum computer (in this case in the MBQC paradigm). However, since stabilizer states are efficiently classically simulable, this wouldn't be a hugely useful practical result.

Another possible candidate for S would be those corresponding to arithmetic diagrams with only $O(\log n)$ W nodes, $O(\log n)$ Z phases of non-unit magnitude and $O(\log n)$ \blacktriangledown nodes with bounded fan-out also of $O(\log n)$. [52] outlines how to prepare an m -output W node with only a single post-selection. A \blacktriangledown can be prepared from a W node with $O(m)$ cups, each requiring two post-selections. Assuming that phases of $e^{i\theta}$ can be prepared efficiently, the following preparation of \boxed{a} for $a = re^{i\theta}$ also uses only a single post-selection:



Finally, $\text{green circle with two cups}$ can be prepared as:







Since the only generators of an arithmetic diagram are \blacktriangle , $\text{green circle with two cups}$, \boxed{a} , \blacktriangledown , we can therefore efficiently map each generator to a quantum circuit, incurring only $O(\log n)$ post-selections. Then one can efficiently sample this circuit by running it $2^{O(\log n)} = \text{poly}(n)$ times to account for the post-selections.

Entanglement

Section 6.1.1 gave an algebraic characterisation of entanglement in terms of irreducible polynomials, which enabled an new algorithm for entanglement detection. More generally, I believe it is worth considering what an algebraic theory of entanglement would look like. Product states become products of polynomials, while the two fundamental entangled states become:

$$|W_n\rangle = \sum_{i=1}^n x_i, \quad |GHZ_n\rangle = 1 + \prod_{i=1}^n x_i$$

This characterisation nicely captures the local versus global nature of the two entangled states. So a preliminary question would be: can interesting entangled states be constructed as trees of  ,  ? Of course, a W node should be at the top to guarantee inseparability. Otherwise, one presumably has complete freedom over how to choose the subsequent nodes. Of course, the more sums in the resulting arithmetic formula, the more the state will resemble the W state and the more products, the more the state will resemble the GHZ state. Other than that, how do algebraic properties of the arithmetic formula reflect on the information-theoretic properties of the entangled state?

Another important question is how to represent local operations. Since  ,  are universal single-qubit operations, we can represent local unitaries on some $p(x_1, \dots, x_n)$ as:

$$\begin{aligned} \begin{array}{c} | \\ \boxed{a} \\ | \\ x_i \end{array} &:: p \mapsto p(x_1, \dots, ax_i, \dots, x_n) \\ \begin{array}{c} | \\ \boxed{y} \\ | \\ x_i \end{array} &:: p \mapsto p(x_1, \dots, \frac{1-x_i}{1+x_i}, \dots, x_n) \times (1+x_i) \end{aligned}$$

Imagine this can be extended to SLOCC operations. Then it would be very interesting to examine the algebraic criteria for SLOCC equivalence, much as the majorisation criteria shed light on entanglement classification [53]. Even better, can an equivalence relation on entangled states be defined that doesn't break down at $n = 4$?

Bibliography

- [1] B. Coecke, A. Kissinger, A. Merry, and S. Roy, “The ghz/w-calculus contains rational arithmetic,” *arXiv preprint arXiv:1103.2812*, 2011.
- [2] C. E. Shannon, “A mathematical theory of communication,” *The Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.
- [3] G. Tononi, “An information integration theory of consciousness,” *BMC neuroscience*, vol. 5, pp. 1–22, 2004.
- [4] T. Mukerji, P. Avseth, G. Mavko, I. Takahashi, and E. F. González, “Statistical rock physics: Combining rock physics, information theory, and geostatistics to reduce uncertainty in seismic reservoir characterization,” *The Leading Edge*, vol. 20, no. 3, pp. 313–319, 2001.
- [5] D. Deutsch, *It from qubit*. Cambridge University Press, 2004, pp. 90–102.
- [6] C. BENNET, “Quantum cryptography: Public key distribution and coin tossing,” in *Proceedings of the IEEE International Conference on Computers, Systems, and Signal Processing, Bangalore, Dec. 1984*, 1984, pp. 175–179.
- [7] S. Pirandola, U. L. Andersen, L. Banchi, M. Berta, D. Bunandar, R. Colbeck, D. Englund, T. Gehring, C. Lupo, C. Ottaviani *et al.*, “Advances in quantum cryptography,” *Advances in optics and photonics*, vol. 12, no. 4, pp. 1012–1236, 2020.
- [8] G. Brassard, “Quantum communication complexity (a survey),” *arXiv preprint quant-ph/0101005*, 2001.
- [9] A. Montanaro, “Quantum algorithms: an overview,” *npj Quantum Information*, vol. 2, no. 1, pp. 1–8, 2016.

- [10] P. W. Shor, “Algorithms for quantum computation: discrete logarithms and factoring,” in *Proceedings 35th annual symposium on foundations of computer science*. Ieee, 1994, pp. 124–134.
- [11] 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*, vol. 70, no. 13, p. 1895, 1993.
- [12] S. Abramsky and B. Coecke, “A categorical semantics of quantum protocols,” in *Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science, 2004*. IEEE, 2004, pp. 415–425.
- [13] R. A. Shaikh, Q. Wang, and R. Yeung, “How to sum and exponentiate hamiltonians in zxw calculus,” *arXiv preprint arXiv:2212.04462*, 2022.
- [14] A. Toumi, R. Yeung, and G. de Felice, “Diagrammatic differentiation for quantum machine learning,” *arXiv preprint arXiv:2103.07960*, 2021.
- [15] R. Duncan, A. Kissinger, S. Perdrix, and J. Van De Wetering, “Graph-theoretic simplification of quantum circuits with the zx-calculus,” *Quantum*, vol. 4, p. 279, 2020.
- [16] N. de Beaudrap and D. Horsman, “The zx calculus is a language for surface code lattice surgery,” *Quantum*, vol. 4, p. 218, 2020.
- [17] K. Meichanetzidis, S. Gogioso, G. De Felice, N. Chiappori, A. Toumi, and B. Coecke, “Quantum natural language processing on near-term quantum computers,” *arXiv preprint arXiv:2005.04147*, 2020.
- [18] B. Poór, Q. Wang, R. A. Shaikh, L. Yeh, R. Yeung, and B. Coecke, “Completeness for arbitrary finite dimensions of zxw-calculus, a unifying calculus,” *arXiv preprint arXiv:2302.12135*, 2023.

- [19] N. de Beaudrap, A. Kissinger, and J. van de Wetering, “Circuit extraction for zx-diagrams can be $\#$ p-hard,” *arXiv preprint arXiv:2202.09194*, 2022.
- [20] J. Alman and V. V. Williams, “A refined laser method and faster matrix multiplication,” in *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 2021, pp. 522–539.
- [21] M. A. Nielsen and I. L. Chuang, “Quantum computation and quantum information,” *Phys. Today*, vol. 54, no. 2, p. 60, 2001.
- [22] R. Cleve, “An introduction to quantum complexity theory,” *Collected Papers on Quantum Computation and Quantum Information Theory*, pp. 103–127, 2000.
- [23] W. K. Wootters and W. H. Zurek, “A single quantum cannot be cloned,” *Nature*, vol. 299, no. 5886, pp. 802–803, 1982.
- [24] A. Einstein, B. Podolsky, and N. Rosen, “Can quantum-mechanical description of physical reality be considered complete?” *Physical review*, vol. 47, no. 10, p. 777, 1935.
- [25] A. Aspect, P. Grangier, and G. Roger, “Experimental realization of einstein-podolsky-rosen-bohm gedankenexperiment: A new violation of bell’s inequalities,” *Physical review letters*, vol. 49, no. 2, p. 91, 1982.
- [26] R. Raussendorf and H. J. Briegel, “A one-way quantum computer,” *Physical review letters*, vol. 86, no. 22, p. 5188, 2001.
- [27] W. Dür, G. Vidal, and J. I. Cirac, “Three qubits can be entangled in two inequivalent ways,” *Physical Review A*, vol. 62, no. 6, p. 062314, 2000.
- [28] B. Coecke and A. Kissinger, “Picturing quantum processes: A first course on quantum theory and diagrammatic reasoning,” in *Diagram-*

- matic Representation and Inference: 10th International Conference, Diagrams 2018, Edinburgh, UK, June 18-22, 2018, Proceedings 10.* Springer, 2018, pp. 28–31.
- [29] B. Coecke, D. Pavlovic, and J. Vicary, “A new description of orthogonal bases,” *Mathematical structures in computer science*, vol. 23, no. 3, pp. 555–567, 2013.
 - [30] B. Coecke and A. Kissinger, “The compositional structure of multipartite quantum entanglement,” in *International Colloquium on Automata, Languages, and Programming*. Springer, 2010, pp. 297–308.
 - [31] A. Hadzihasanovic, “A diagrammatic axiomatisation for qubit entanglement,” in *2015 30th Annual ACM/IEEE Symposium on Logic in Computer Science*. IEEE, 2015, pp. 573–584.
 - [32] E. Jeandel, S. Perdrix, and M. Veshchezerova, “Addition and differentiation of zx-diagrams,” *arXiv preprint arXiv:2202.11386*, 2022.
 - [33] D. Camps, E. Kokcu, L. Bassman Oftelie, W. A. De Jong, A. F. Kemper, and R. Van Beeumen, “An algebraic quantum circuit compression algorithm for hamiltonian simulation,” *SIAM Journal on Matrix Analysis and Applications*, vol. 43, no. 3, pp. 1084–1108, 2022.
 - [34] M. A. Forbes and A. Shpilka, “Complexity theory column 88: Challenges in polynomial factorization¹,” *ACM SIGACT News*, vol. 46, no. 4, pp. 32–49, 2015.
 - [35] O. Gühne and G. Tóth, “Entanglement detection,” *Physics Reports*, vol. 474, no. 1-6, pp. 1–75, 2009.
 - [36] S. Aaronson, “ $P = \text{limits}^? \text{ np}$,” *Open problems in mathematics*, pp. 1–122, 2016.

- [37] E. Bernstein and U. Vazirani, “Quantum complexity theory,” in *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, 1993, pp. 11–20.
- [38] L. M. Adleman, J. Demarrais, and M.-D. A. Huang, “Quantum computability,” *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1524–1540, 1997.
- [39] S. Fenner, F. Green, S. Homer, and R. Pruim, “Determining acceptance possibility for a quantum computation is hard for ph,” in *Proc. 6th Italian Conference on Theoretical Computer Science*. Citeseer, 1998, pp. 241–252.
- [40] N. Saxena, “Progress on polynomial identity testing.” *Bull. EATCS*, vol. 99, pp. 49–79, 2009.
- [41] L. G. Valiant and S. Skyum, “Fast parallel computation of polynomials using few processors,” in *Mathematical Foundations of Computer Science 1981: Proceedings, 10th Symposium Štrbské Pleso, Czechoslovakia August 31–September 4, 1981 10*. Springer, 1981, pp. 132–139.
- [42] A. Shpilka, A. Yehudayoff *et al.*, “Arithmetic circuits: A survey of recent results and open questions,” *Foundations and Trends in Theoretical Computer Science*, vol. 5, no. 3–4, pp. 207–388, 2010.
- [43] L. G. Valiant, “Completeness classes in algebra,” in *Proceedings of the eleventh annual ACM symposium on Theory of computing*, 1979, pp. 249–261.
- [44] J. Krajíček, *Proof complexity*. Cambridge University Press, 2019, vol. 170.
- [45] Y. Tanaka, “Exact non-identity check is nqp-complete,” *International Journal of Quantum Information*, vol. 8, no. 05, pp. 807–819, 2010.

- [46] D. Janzing, P. Wocjan, and T. Beth, ““non-identity-check” is qma-complete,” *International Journal of Quantum Information*, vol. 3, no. 03, pp. 463–473, 2005.
- [47] L. G. Valiant, “The complexity of computing the permanent,” *Theoretical computer science*, vol. 8, no. 2, pp. 189–201, 1979.
- [48] X. Chen, N. Kayal, A. Wigderson *et al.*, “Partial derivatives in arithmetic complexity and beyond,” *Foundations and Trends in Theoretical Computer Science*, vol. 6, no. 1–2, pp. 1–138, 2011.
- [49] S. Saraf, “Recent progress on lower bounds for arithmetic circuits,” in *2014 IEEE 29th Conference on Computational Complexity (CCC)*. IEEE, 2014, pp. 155–160.
- [50] J. A. Grochow and T. Pitassi, “Circuit complexity, proof complexity, and polynomial identity testing: The ideal proof system,” *Journal of the ACM (JACM)*, vol. 65, no. 6, pp. 1–59, 2018.
- [51] T. McElvanney and M. Backens, “Complete flow-preserving rewrite rules for mbqc patterns with pauli measurements,” *arXiv preprint arXiv:2205.02009*, 2022.
- [52] L. Yeh, “Scaling w state circuits in the qudit clifford hierarchy,” *arXiv preprint arXiv:2304.12504*, 2023.
- [53] M. A. Nielsen, “Conditions for a class of entanglement transformations,” *Physical Review Letters*, vol. 83, no. 2, p. 436, 1999.
- [54] D. S. Dummit and R. M. Foote, *Abstract algebra*. Wiley Hoboken, 2004, vol. 3.
- [55] C. Heunen and J. Vicary, *Categories for Quantum Theory: an introduction*. Oxford University Press, 2019.

A | Algebra

Algebra is the study of operations, i.e. mathematical ways of doing things. This appendix briefly reviews the necessary algebra for this thesis. For a proper treatment see [54].

A.1 Magmas

The most basic algebraic operation is a **magma** which is simply a set M and a binary operation $\cdot : M \times M \rightarrow M$. The operation \cdot is often called multiplication, though all we know about \cdot at this point is that it takes in two elements of M and spits out another one. For convenience, $\cdot(x, y)$ is usually written as $x \cdot y$.

A magma is **commutative** if it satisfies $\forall x, y \in X, x \cdot y = y \cdot x$.

Given two magmas $(M_1, \cdot), (M_2, \circ)$, a function $\phi : M_1 \rightarrow M_2$ is called a **homomorphism** if it satisfies

$$\phi(x \cdot y) = \phi(x) \circ \phi(y)$$

In which case, ϕ is said to “preserve multiplication”. Often the operation symbols are dropped to give $\phi(xy) = \phi(x)\phi(y)$. If ϕ is also a bijection, then it is called an **isomorphism** and we write $M_1 \simeq M_2$. Intuitively, isomorphic magmas effectively do the same thing as one another except for a relabelling of their elements.

The definition of magmas is so general that there is very little structure to work with. Algebra mostly studies operations with additional conditions which allow for a richer theory. I only include magmas here to give the most general definition of commutativity and homomorphism.

A.2 Monoids

A monoid (M, \cdot) is a magma which moreover satisfies:

- *identity*: $\exists e \in M, \forall x \in M, e \cdot x = x = x \cdot e$
- *associativity*: $\forall x, y, z \in M, x \cdot (y \cdot z) = (x \cdot y) \cdot z$

The identity element allows us to do nothing. One can show that the identity element is unique. Associativity allows us to unambiguously write $x \cdot y \cdot z$.

Some useful examples of monoids (which both happen to be commutative) are:

- $(\mathbb{N}, +)$, with 0 as the identity element
- For any set X , $(\mathcal{P}(X), \cup)$ is a monoid with \emptyset as the identity element

If a monoid also satisfies the inverse condition: $\forall x \in M, \exists x^{-1} \in M, x \cdot x^{-1} = e = x^{-1} \cdot x$, then it is called a **group**. Commutative groups are sometimes called **Abelian**. The classic example is $(\mathbb{Z}, +)$.

A.3 Rings

A ring $(R, +, \cdot)$ is a set R together with two binary operations such that:

- $(R, +)$ forms an Abelian group
- (R, \cdot) forms a monoid
- *distributivity*: $\forall x, y, z \in R, x \cdot (y + z) = (x \cdot y) + (x \cdot z)$ and $(x + y) \cdot z = (x \cdot z) + (y \cdot z)$

The operations are usually referred to as addition with unit 0 and multiplication with unit 1, respectively. A commutative ring is when the multiplication is commutative. It follows from distributivity that $x \cdot 0 = 0$ for any $x \in R$. Some examples of rings are as follows:

- $(\mathbb{Q}, +, \times)$
- $(M_n, +, \circ)$ is a non-commutative ring, where M_n is the set of $n \times n$ matrices.
- $\mathbb{C}[x]$, the set of polynomials over \mathbb{C} . This will be discussed in more detail shortly.

For two rings R, S , a ring homomorphism is map $\phi : R \rightarrow S$ that preserves both addition and multiplication.

For a commutative ring R , a non-empty subset $I \subseteq R$ is called an **ideal** if it closed under addition from the inside and multiplication from the outside, i.e. $\forall x, y \in I, r \in R, :$

$$x + y \in I, r \cdot x \in I$$

For any $x \in R$, the smallest ideal containing x is written (x) . In general, (x_1, \dots, x_n) is the smallest ideal containing x_1, \dots, x_n . If an ideal contains 1 then it will contain the whole ring in order to be closed under multiplication. Thus fields have only two ideals: $(0) = \{0\}, (1) = R$.

Given an ideal $I \subseteq R$, a **coset** is set $a + I := \{a + x : x \in I\}$. If $a \in I$, then clearly $a + I = I$. It can also be seen that $a \in (b + I) \implies b \in (a + I)$ so the set of cosets partition R . In fact, this yields a ring if we define:

$$(a + I) + (b + I) := (a + b)I, \quad (a + I) \cdot (b + I) := (a \cdot b) + I$$

This is called the **quotient** ring and written R/I . One can think of R/I as R , except that everything in I becomes 0.

A **field** is when \cdot forms a group (i.e. every nonzero element has a multiplicative inverse). This includes rings like \mathbb{Q} and \mathbb{C} but not \mathbb{Z} .

A.3.1 Polynomials

For any commutative ring R , $R[x]$ denotes the ring of polynomials over an indeterminate x . Formally, elements $p(x) \in R[x]$ are sums $p(x) = \sum_{i=1}^k a_i x^i$ for some coefficients $(a_1, \dots, a_k) \in R^k$. The addition and multiplication operations are inherited from R . Polynomials over several variables can be defined recursively as $R[x_1, \dots, x_n] = R[x_1, \dots, x_{n-1}][x_n]$.

Polynomials can be evaluated on elements of R , interpreting them as functions $p(x) : R \rightarrow R$. However, often the indeterminates can be interpreted as elements foreign to R . For example, the isomorphism $\mathbb{R}[x]/(x^2 + 1) \simeq \mathbb{C}$ allows x to be interpreted as $i = \sqrt{-1}$.

In many polynomial rings, every $p(x) \in R[x]$ can be written as $r \cdot f_1 \cdot \dots \cdot f_k$ for $r \in R, f_1, \dots, f_k \in R[x]$, where all the f_i are **irreducible** meaning that they cannot be factored any further themselves. Whether a polynomial is reducible is very sensitive to which coefficients are in R . For example, $x^2 - 2$ is irreducible in $\mathbb{Z}[x]$, but not in $\mathbb{R}[x]$ where it can be factored as $(x + \sqrt{2})(x - \sqrt{2})$.

In a ring where every non-zero element can be *uniquely* factored into a product of irreducible elements, then it is called a unique factorisation domain (UFD). An important result in ring theory is that if R is a UFD, so is $R[x]$.

A.4 Vector Spaces

A vector space V is always defined over some field F . The elements of V are often called vectors and the elements of F are often called scalars. V is equipped with an addition operation $+: V \times V \rightarrow V$ and a multiplication $\times: F \times V \rightarrow V$ such that:

- $+$ forms an Abelian group.

- Scalar multiplication is compatible with the field operations:

$$\forall \vec{v} \in V, x, y \in F, \quad a \times (b \times \vec{v}) = (a \cdot b) \times \vec{v}$$

This is often abbreviated to $a(b\vec{v}) = (ab)\vec{v}$.

- Scalar multiplication is compatible with F 's multiplicative unit:

$$\forall \vec{v} \in V, \quad 1\vec{v} = \vec{v}$$

- Scalar multiplication distributes over vector addition:

$$\forall \vec{v}, \vec{w} \in V, x \in F, \quad x(\vec{v} + \vec{w}) = x\vec{v} + x\vec{w}$$

- Scalar multiplication distributes over field addition:

$$\forall \vec{v} \in V, x, y \in F, \quad (x + y)\vec{v} = x\vec{v} + y\vec{v}$$

A familiar example of a vector space is the Cartesian plane \mathbb{R}^2 . Elements $\vec{v} \in \mathbb{R}^2$ are represented as vectors $\begin{bmatrix} x \\ y \end{bmatrix}$, for some $x, y \in \mathbb{R}$. The multiplicative identity is the identity matrix $I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$.

Given a set of vectors $B = \{\vec{v}_1, \dots, \vec{v}_n\} \subset V$, B is said to **span** V if every element of V can be written as a linear combination $x_1\vec{v}_1 + \dots + x_n\vec{v}_n$, for some $x_1, \dots, x_n \in F$. If $x_1\vec{v}_1 + \dots + x_m\vec{v}_m = 0 \implies x_1 = \dots = x_m = 0$ for any finite subset of B , then B is said to be **linearly independent**. This is equivalent to saying no element of B can be written as a linear combination of B 's remaining elements. If B simultaneously spans V and is linearly independent, then B is called a **basis**. A fundamental result in the study

of vector spaces (which is usually called linear algebra) is that every vector space has a basis. The standard basis for \mathbb{R}^2 is $\left\{ \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right\}$.

A vector space homomorphism $f : V \rightarrow W$ is typically called a **linear map** and must satisfy:

$$\forall \vec{v}, \vec{w} \in V, x \in F, \quad f(\vec{v} + \vec{w}) = f(\vec{v}) + f(\vec{w}), \quad f(a\vec{v}) = af(\vec{v})$$

The beauty of linearity is that f 's action on a basis B determines its action on the entire vector space V . Thus linear maps are typically represented as matrices, where the i th column corresponds to the image of the i th basis vector, $f(\vec{v}_i)$. For example, a 90° rotation in \mathbb{R}^2 can be represented by the matrix $\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$.

If A is an $m_1 \times n_1$ matrix and B is an $m_2 \times n_2$ matrix, then the **Kronecker product** $A \otimes B$ is the $m_1 m_2 \times n_1 n_2$ matrix with entries

$$\begin{bmatrix} a_{11}B & \cdots & a_{1n_1}B \\ \vdots & \ddots & \vdots \\ a_{m_1 1}B & \cdots & a_{m_1 n_1}B \end{bmatrix}$$

Some vector spaces have a well-defined measure of distance between vectors. Such a distance measure is called an **inner product** $\langle \cdot, \cdot \rangle : V \times V \rightarrow F$. An inner product always defines a norm $\|\vec{v}\| = \sqrt{\langle \vec{v}, \vec{v} \rangle}$. In a **Hilbert space**, this norm must be complete meaning that every Cauchy sequence of elements of V must converge to an element of V . The prime example is \mathbb{C}^n , for any $n > 0$.

B | Category Theory

This section is drawn from <https://ncatlab.org/nlab/show/HomePage> and [28]. The only relevant difference is that the equation numbers align with section 3.1.

B.1 Category

A category \mathcal{C} consists of:

- A collection of **objects** $X, Y, Z, \dots \in \mathcal{C}$.
- For every two objects $X, Y \in \mathcal{C}$, a collection of **morphisms** $\mathcal{C}(X, Y)$.
Such a morphism is often written $f : X \rightarrow Y$. These morphisms must satisfy the following rules:

– **Composition:**

$$\forall f : X \rightarrow Y, g : Y \rightarrow Z, \exists g \circ f : X \rightarrow Z \quad (\text{B.1})$$

– **Associativity:** for any compatible morphisms f, g, h ,

$$h \circ (g \circ f) = (h \circ g) \circ f \quad (\text{B.2})$$

– **Identity:** for every object $X \in \mathcal{C}$, there is an identity morphism $id_X : X \rightarrow X$ such that for any $f : X \rightarrow Y$,

$$f \circ id_X = f = id_Y \circ f \quad (\text{B.3})$$

If \mathcal{C} is a category, then \mathcal{C}^{op} is the category with the same objects as \mathcal{C} but all arrows reversed.

B.2 Monoidal Category

A monoidal category is a category \mathcal{C} plus:

- An operation $\otimes : \mathcal{C} \times \mathcal{C} \rightarrow \mathcal{C}$, defined both on objects and morphisms such that:

– **Identity:**

$$id_X \otimes id_Y = id_{X \otimes Y} \quad (\text{B.4})$$

– **Associativity:**

$$X \otimes (Y \otimes Z) = (X \otimes Y) \otimes Z$$

$$h \otimes (g \otimes f) = (h \otimes g) \otimes f \quad (\text{B.5})$$

– **Coherence:**

$$(g_1 \circ f_1) \otimes (g_2 \circ f_2) = (g_1 \otimes g_2) \circ (f_1 \otimes f_2) \quad (\text{B.6})$$

- A designated **unit** object $\mathbf{1} \in \mathcal{C}$ such that:

$$A \otimes \mathbf{1} = A = \mathbf{1} \otimes A$$

$$id_{\mathbf{1}} \otimes f = f = f \otimes id_{\mathbf{1}} \quad (\text{B.7})$$

Bialgebra??

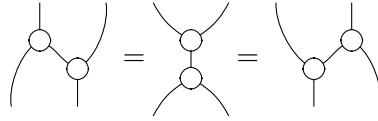
B.2.1 Frobenius Algebra

Within a monoidal category $(\mathcal{C}, \otimes, I)$, a Frobenius algebra consists of the following:

- A set of objects A

- A monoid $\mu : A \otimes A \rightarrow A$, with unit $\eta : I \rightarrow A$
- A comonoid $\delta : A \rightarrow A \otimes A$ with counit $\epsilon : A \rightarrow I$. A comonoid is simply a monoid in \mathcal{C}^{op} .
- *Frobenius laws*: $(id_A \otimes \mu) \circ (\delta \otimes id_A) = \delta \circ \mu = (\mu \otimes id_A) \circ (id_A \otimes \delta)$

Diagrammatically, the Frobenius laws look like:



A Frobenius algebra is commutative if μ is commutative. A Frobenius algebra is **special** if $\mu \circ \delta = id_A$. A special commutative Frobenius algebra is often abbreviated to SCFA.

A very important result known as the **spider theorem** says that any network of nodes from an SCFA is uniquely determined by the number of inputs and number of outputs. See [55] for more details.

B.3 Symmetric Monoidal Category

A symmetric monoidal category is a monoidal category $(\mathcal{C}, \otimes, \mathbf{1})$ plus a morphism $\sigma_{X,Y} : X \otimes Y \rightarrow Y \otimes X$ for all objects X, Y such that:

$$\sigma_{Y,X} \circ \sigma_{X,Y} = id_{X \otimes Y} \quad (\text{B.8})$$

$$\sigma_{X,\mathbf{1}} = id_X \quad (\text{B.9})$$

$$(f \otimes g) \circ \sigma_{X_1, X_2} = \sigma_{Y_1, Y_2} \circ (g \otimes f) \quad (\text{B.10})$$

$$(id_Y \otimes \sigma_{X,Z}) \circ (\sigma_{X,Y} \otimes id_Z) = \sigma_{X,Y \otimes Z} \quad (\text{B.11})$$

B.4 Compact Closed Category

A compact closed category is a symmetric monoidal category $(\mathcal{C}, \otimes, \mathbf{1})$ in which for every $X \in \mathcal{C}$, there exists another object $X^* \in \mathcal{C}$ and morphisms $\epsilon_X : X \otimes X^* \rightarrow \mathbf{1}, \eta_X : \mathbf{1} \rightarrow X^* \otimes X$ such that

$$\begin{aligned} (\epsilon_X \otimes id_X) \circ (id_X \otimes \eta_X) &= id_X, \\ (id_{X^*} \otimes \epsilon_X) \circ (\eta_X \otimes id_{X^*}) &= id_{X^*} \end{aligned} \quad (\text{B.12})$$

When $X = X^*$ and

$$\sigma_{X,X} \circ \eta_X = \eta_X \quad (\text{B.13})$$

Then X is called coherently self-dual.