

Equational Reasoning with Controlled ZXW Diagrams

Edwin Agnew

Department of Computer Science
University of Oxford

Lia Yeh

Department of Computer Science
University of Oxford
Quantinuum
17 Beaumont Street
Oxford OX1 2NA, UK
lia.yeh@cs.ox.ac.uk

Richie Yeung

Department of Computer Science
University of Oxford
Quantinuum
17 Beaumont Street
Oxford OX1 2NA, UK
richie.yeung@cs.ox.ac.uk

An important concept for graphical rewriting in the ZXW calculus is a *controlled diagram* — a diagram extended with an additional input wire for triggering the operation on or off. Controlled diagrams have been applied in a number of areas including quantum chemistry, quantum machine learning, and photonics. In this work, we investigate the algebraic structure of controlled diagrams. First, we prove that the higher-order map which sends square matrices to their controlled square matrix is a lax monoidal functor. We then show that controlled matrices form a ring, yielding powerful rewrite rules for large classes of diagrams; we also show that controlled states form a ring, in this case isomorphic to the ring of multilinear polynomials. Augmenting the ZXW calculus with controlled square matrices as black-box generators, we prove completeness for polynomials over same-size square matrices. Through formalising the algebraic properties of quantum control, we give rise to powerful new graphical rewrite rules for black-box diagrams.

1 Introduction

Controlling or branching to different possible linear maps, relations, or channels is important across quantum information and quantum computation, and has been studied through many different approaches. In quantum algorithms common techniques are block encodings [9, 17] and linear combination of unitaries [4], while a number of categorical formalisations have included routed quantum circuits [25], the many-worlds calculus [3], categorifying signal flow diagrams [2], and classical and quantum control in quantum modal logic [20].

The question we are interested in is how quantum graphical calculi such as the ZX [5], ZW [6], and ZH [1] calculus can be augmented to support properties of quantum control. An early use of controlled state diagrams was for proving constructive and rational angle ZX calculus completeness [13]. More recently, controlled state and controlled matrix diagrams have been applied to addition and differentiation of ZX diagrams [12], differentiating and integrating ZX diagrams for quantum machine learning [26], Hamiltonian exponentiation and simulation [21], and non-linear optical quantum computing [8]. To sum ZX diagrams, these works have used controlled states along with the W generator from the ZW calculus.

Given how useful controlled diagrams are, a natural question to ask is why they work: What their underlying mathematical structures are, and which equational rewrites they satisfy.

In this paper, we first introduce a higher-order map Ctrl which sends states to controlled states, and square matrices to controlled square matrices. We prove that Ctrl is a lax monoidal functor on the subcategory of all square matrices. This allows us to use the functorial boxes of Ref. [15] to control ZX diagrams so long as functoriality is only applied when the number of input and output wires are equal. Moreover, AND of the controls is a natural transformation corresponding to nested applications of Ctrl .

Next, we show that the set of all controlled n -partite states defines a commutative ring $(\tilde{\mathcal{S}}^n, \boxplus, \boxtimes)$. We introduce \boxplus which defines an Abelian group and \boxtimes which defines a commutative monoid, and show that \boxtimes distributes over \boxplus . The fragment of the qubit ZW calculus corresponding to controlled states hence defines a subcategory we prove is isomorphic to multilinear polynomials $\mathbb{C}[x_1, \dots, x_n]/(x_1^2, \dots, x_n^2)$, and proved completeness for. Analogously, we show that the set of all controlled square matrices on n qubits defines a non-commutative ring $(\tilde{\mathcal{M}}^n, \blacktriangle, \blacktriangleright)$.

We add controlled square matrices and rewrite rules for them to the ZXW calculus, in which we plug controlled states into each control wire of controlled square matrices. We prove their completeness and that this is isomorphic to multivariate polynomials over same-size square matrices. Commutativity of controlled square matrices holds in the special case that the controls target mutually exclusive sectors, allowing copying of arbitrary controlled diagrams. As a result, we can factor multivariate polynomials over same-size square matrices. This means we now have the ability to factor any Hamiltonian in the ZXW calculus [21], even with all its terms black-boxed. In summary, these algebraic properties of quantum control give rise to powerful new graphical rewrite rules for black-box diagrams.

2 ZXW Calculus

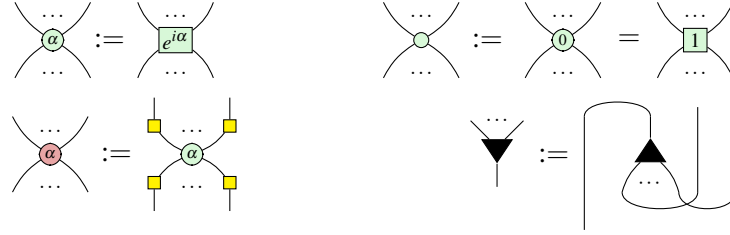
This section introduces the generators of the ZXW calculus. The ZXW calculus is a diagrammatic formalism for qudit computation, unifying the ZX and ZW calculi and synthesising their relative strenghts. The ZXW calculus consists of diagrams built from a small number of generators and equipped with a complete set of rewrite rules which enables all equalities between linear maps to be proven diagrammatically. Diagrams are to be read top to bottom or, in later sections, left to right.

2.1 Generators

The qubit ZXW calculus is built from the following generators:

$$\begin{aligned}
 \left[\begin{array}{c} | \\ | \end{array} \right] &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & \left[\begin{array}{c} \diagup \quad \diagdown \\ \diagdown \quad \diagup \end{array} \right] &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & \left[\begin{array}{c} \cap \end{array} \right] &= \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} & \left[\begin{array}{c} \cup \end{array} \right] &= \begin{bmatrix} 1 & 0 & 0 & 1 \end{bmatrix} \\
 \left[\begin{array}{c} n \\ \vdots \\ \text{c} \\ \vdots \\ m \end{array} \right] &= |0^n\rangle\langle 0^n| + c|1^m\rangle\langle 1^m|, c \in \mathbb{C} & \left[\begin{array}{c} \blacktriangle \end{array} \right] &= |00\rangle\langle 0| + |01\rangle\langle 1| + |10\rangle\langle 1| \\
 \left[\begin{array}{c} | \\ \text{y} \end{array} \right] &= \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}
 \end{aligned}$$

For simplicity, we introduce the following additional notation:



The complete rule set is given in Appendix A. Several important lemmas are found in Appendix B.

3 Controlled Diagrams

3.1 Definitions

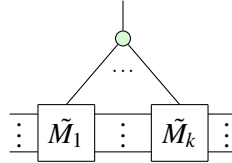
We recall the definition of a controlled diagram from [21],

Definition 3.1. For an arbitrary square matrix M , the controlled matrix of M is the diagram \tilde{M} such that:

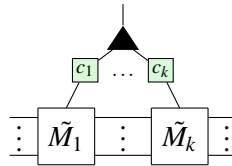
$$\begin{array}{c} \text{red dot} \\ | \\ \boxed{\tilde{M}} \end{array} = \text{horizontal line} \quad \begin{array}{c} \text{red dot labeled } \pi \\ | \\ \boxed{\tilde{M}} \end{array} = \boxed{M} \quad (3.1)$$

It is possible to perform matrix arithmetic with controlled diagrams.

Proposition 1 ([21]). Given controlled matrices $\tilde{M}_1, \dots, \tilde{M}_k$, the controlled matrix $\widetilde{\prod_i \tilde{M}_i}$ is given by



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



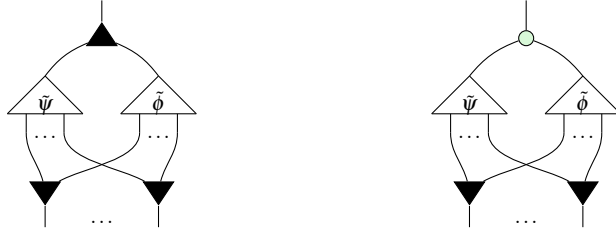
Proof. See propositions 3.3 and 3.4 in [21]. □

We can also define the analogue for states.

Definition 3.2. For an arbitrary state ψ , the controlled state of ψ is the diagram $\tilde{\psi}$ such that:

$$\begin{array}{c} \text{red dot} \\ | \\ \triangle \tilde{\psi} \end{array} = \text{two red dots} \quad \begin{array}{c} \text{red dot labeled } \pi \\ | \\ \triangle \tilde{\psi} \end{array} = \triangle \psi \quad (3.2)$$

The addition and multiplication of controlled states are defined similarly to controlled matrix arithmetic, except that a layer of \blacktriangledown s are appended at the bottom to preserve the number of outputs.



The role of \blacktriangledown is to *copy* controlled diagrams, as shown in section 4.1.

3.2 Control Functor

The operation of turning a square matrix to its controlled diagram can be made into a lax monoidal functor. Let \mathbf{Hilb}_{sq} be the subcategory of Hilbert spaces and square matrices. Adding an additional horizontal wire to facilitate composition, $\text{Ctrl} : \mathbf{Hilb}_{sq} \rightarrow \mathbf{Hilb}$ is defined as follows for arbitrary $M \in \text{Hom}_{\mathbf{Hilb}_{sq}}(V, V)$.

$$\text{Ctrl} :: V \xrightarrow{M} V \mapsto V \xrightarrow{\tilde{M}} V \quad (3.3)$$

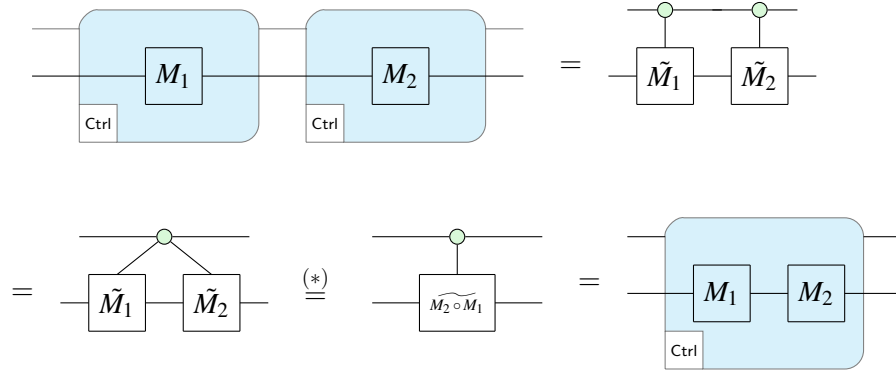
In the functorial box notation of [15], this would be:

$$\begin{array}{c} \mathbb{C}^2 \\ \hline V \end{array} \begin{array}{c} \text{Ctrl} \\ \hline M \end{array} \begin{array}{c} \hline \mathbb{C}^2 \\ V \end{array} = \begin{array}{c} \mathbb{C}^2 \\ \hline V \end{array} \begin{array}{c} \tilde{M} \end{array} \begin{array}{c} \hline \mathbb{C}^2 \\ V \end{array} \quad (3.4)$$

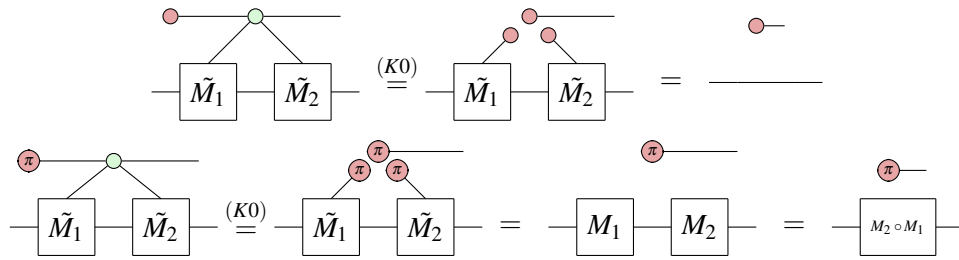
Proposition 2. The map Ctrl defined in (3.3) is a lax monoidal functor.

Proof. On $\text{id}_V : V \rightarrow V$:

Let $D_1 : V \rightarrow V, D_2 : V \rightarrow V$. Then composing $\text{Ctrl}(M_2) \circ \text{Ctrl}(M_1)$:

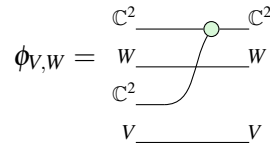


Where $(*)$ follows from:

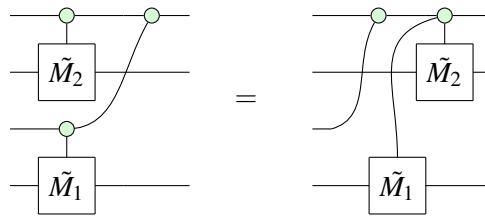


Which implies that controlled diagrams can fuse under Ctrl .

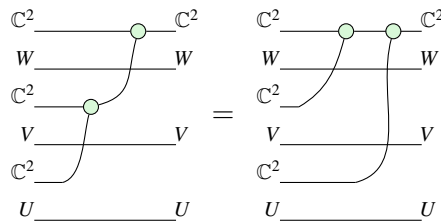
Ctrl preserves the monoidal unit since $\mathbf{1}_{\text{Hilb}_{\text{sq}}} = \mathbf{1}_{\text{Hilb}} = \text{Ctrl}(\mathbf{1})$. Ctrl is lax thanks to the following morphism: $\phi_{V,W} : \text{Ctrl}(V) \otimes \text{Ctrl}(W) \rightarrow \text{Ctrl}(V \otimes W)$:



ϕ is natural since for any $M_1 : V \rightarrow V, M_2 : W \rightarrow W$, we have:



Finally, ϕ satisfies the coherence condition since for any U, V, W :

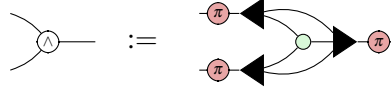


□

3.3 Multiple Control

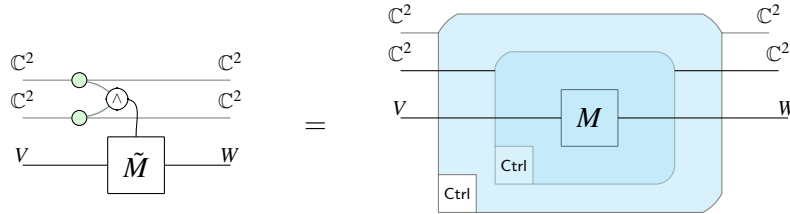
This section proves that controlling a controlled diagram gives the AND of the control wires. First we show how to represent the binary AND gate in the ZXW calculus, and check it in Appendix C. This construction is by deMorgan's law of a diagram for binary OR from Proposition 5.

Lemma 3.1.



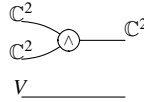
Multiple applications of Ctrl ANDs the controls, proved in Appendix C.

Proposition 3.

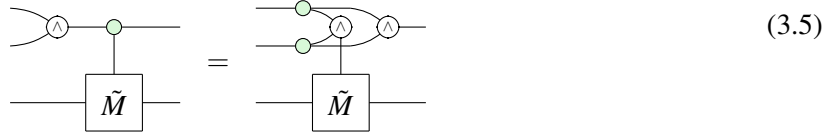


Since AND is a monoid, it is reasonable to expect that multiple control induces a monad. Unfortunately, this does not appear to go through. Although AND does define a natural transformation (in the relevant category), defining the unit of the monad necessitates defining Ctrl of non-square matrices, which breaks functoriality. Nevertheless we prove that AND is a natural transformation.

Proposition 4. $\mu : \text{Ctrl}^2 \rightarrow \text{Ctrl}$ is a natural transformation with components $\mu_V : \text{Ctrl}^2 V \rightarrow \text{Ctrl} V$ defined as follows:



Proof.



due to the $Z - H$ bialgebra rule in the ZH calculus, as the multiply box is an H gate after an H-box. \square

Corollary 1. By the $Z - X$ and $Z - W$ bialgebra rules as well as the Z spider fusion rule, in addition to the multiply box, Equation (3.5) holds for all other 2-input, 1-output generators: Z, X, and W.

4 Polynomials

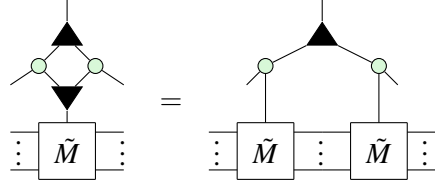
In this section, we reverse-engineer the underlying algebraic properties of controlled state and controlled square matrix diagrams. This builds up to diagrams for the unique normal form for states used for the first proofs of complete axiomatisation for qubit graphical calculi [10, 11].

All proofs in this section can be found in Appendix C.

4.1 Rings

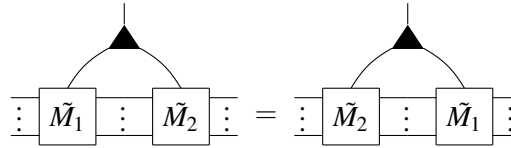
Let \tilde{M}_n be the set of controlled square matrices on n qubits. The goal of this section is to prove that the addition and multiplication operations introduced above induce a ring on \tilde{M}_n . Likewise, we show that the set of controlled n -qubit states \tilde{S}_n also forms a ring. Before doing so, we prove a few important lemmas. The first lemma enables us to copy controlled matrices.

Lemma 4.1. *For any square matrix M ,*

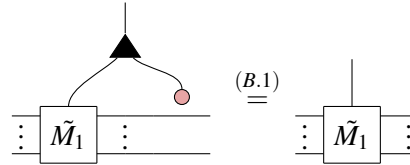

(4.1)

Now we show that controlled matrix addition and multiplication satisfy the ring axioms. Associativity of $+$, \times follow immediately from (Aso, S1), respectively. Commutativity of addition follows from the commutativity of matrix addition.

Lemma 4.2. *Let M_1, M_2 be $n \times n$ matrices.*


(4.2)

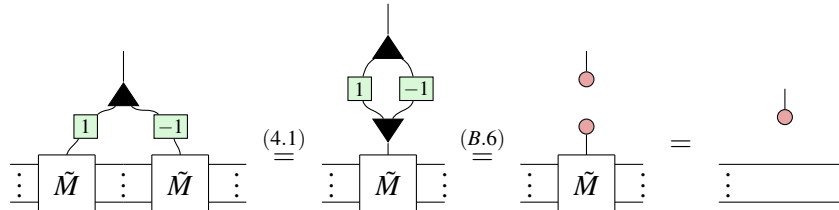
The additive identity is defined as $\text{red circle} \otimes I_n$:



The multiplicative identity is defined very similarly as $\text{green circle} \otimes I_n$. The existence of additive inverses relies on the copying lemma from before.

Lemma 4.3. *The additive inverse of \tilde{M} is $\text{green box } -1 \circ \tilde{M}$*

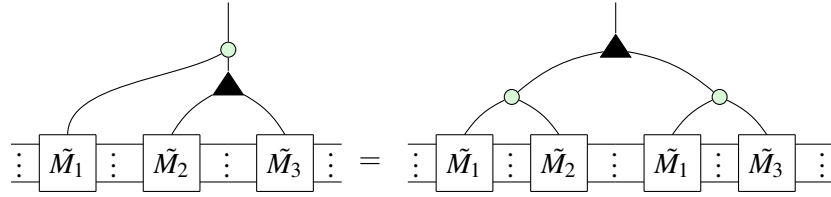
Proof.



□

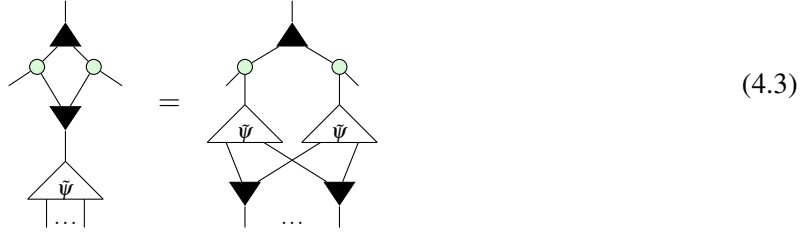
Finally, we prove distributivity.

Lemma 4.4.



Combining the lemmas of this section shows that controlled matrices form a ring. A similar result can be shown for controlled states. Once again, we start with the ability to copy controlled states.

Lemma 4.5. *For any state ψ ,*



Many of the ring axioms follow directly from basic ZXW rules. For example we can show commutativity of addition as follows:

Lemma 4.6. *For n -partite states ψ_1, ψ_2 , $\tilde{\psi}_1 \boxplus \tilde{\psi}_2 = \tilde{\psi}_2 \boxplus \tilde{\psi}_1$*

Associativity of \boxplus follows similarly, using (Aso). Next we have the additive identity.

Lemma 4.7. $\tilde{\psi} \boxplus \tilde{0} = \tilde{\psi}$

The additive inverse is defined similarly to the case of controlled matrices.

Lemma 4.8. *For a controlled state $\tilde{\psi}$, its additive inverse is $\tilde{\psi} \circ \boxed{-1}$*

Associativity and commutativity of \boxtimes follow as before, using (S1) for green circle . Finally, we must prove distributivity.

Lemma 4.9. $\tilde{\psi}_1 \boxtimes (\tilde{\psi}_2 \boxplus \tilde{\psi}_3) = (\tilde{\psi}_1 \boxtimes \tilde{\psi}_2) \boxplus (\tilde{\psi}_1 \boxtimes \tilde{\psi}_3)$

Remark 1. A different addition and multiplication for controlled states was defined in Ref. [13]. There corresponded to entry-wise addition and multiplication of statevectors, while our \boxplus and \boxtimes correspond to addition and multiplication of polynomials in bijective correspondence to controlled states, which we show next.

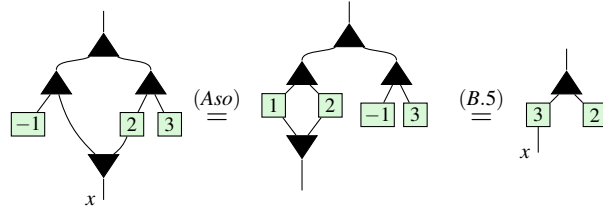
4.2 Arithmetic

It's been known since 2011 that \blacktriangle , green circle can be used to add and multiply number states \boxed{a} , respectively [7]. In the previous section we saw that \blacktriangle , green circle can moreover be used to copy controlled diagrams. In this section, we explain this connection by demonstrating that controlled states are in fact isomorphic to multilinear polynomials. This being a bijection is a well-known folklore result in the study of entangled states, but to the best of our inquiries we are not aware of a proof. More generally, Ref. [28] presented Cartesian Distributive Categories exemplified by polynomial circuits, which are isomorphic to polynomials over arbitrary commutative semirings or rings; their proof is non-constructive, giving explicit proof only for the case of Boolean circuits [27].

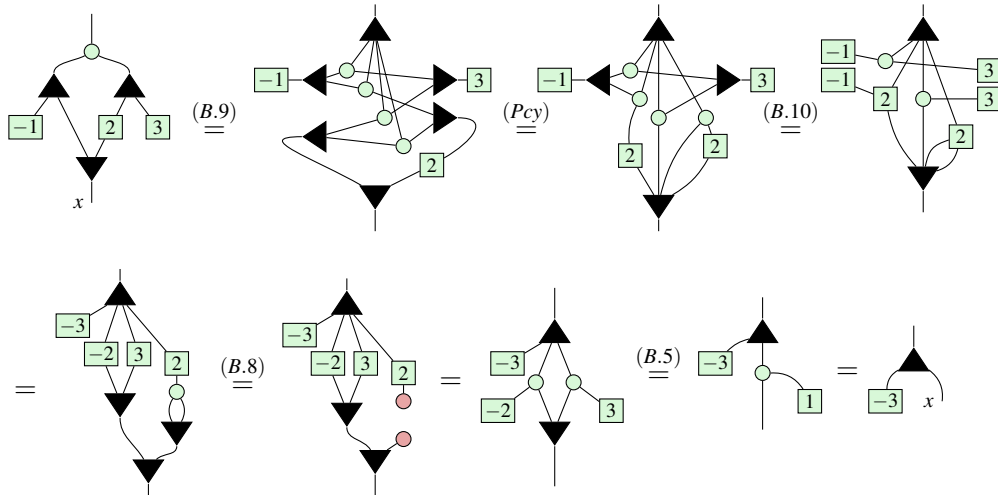
Firstly, we describe how to interpret certain ZXW diagrams as polynomials. Consider the diagrams:



If we treat the bottom wires as an indeterminate x , we can read these bottom-up as computing $x - 1$ and $2x + 3$, respectively. Moreover, since these diagrams are both controlled states, they can be added together, yield a diagram resembling $3x + 2$:








When trying to multiply these diagrams, rather than getting $(x - 1)(2x + 3) = 2x^2 + x - 3$, we instead get $x - 3$.







The reason for the missing $2x^2$ term is that (B.8) implies $x^2 = 0$. Other than that, controlled state arithmetic appears to faithfully reflect polynomial arithmetic. To help formalise this correspondence, we introduce the following definition.

Definition 4.1. A ZXW diagram with a single input on top is **arithmetic** if it contains only $|$, \times wires,

, ,  nodes and  boxes.

Remark 2. This fragment of the ZXW calculus defines a subcategory; adding , this is a Cartesian Distributive Category as defined in Ref. [28].

To interpret an arithmetic ZXW diagram as an arithmetic expression, read  as $+$,  as \times ,  as the number a ,  as fanout and output/bottom wires as variables x_1, \dots, x_n numbered from left to right. The following lemma establishes that all arithmetic diagrams are controlled states:

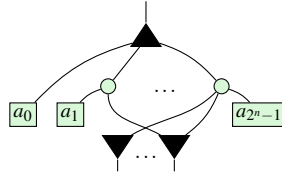
Lemma 4.10. *For any arithmetic diagram A ,*

$$\begin{array}{c} \text{red circle} \\ | \\ \boxed{A} \\ | \dots | \end{array} = \begin{array}{c} \text{red circle} \quad \text{red circle} \\ | \quad | \\ \dots \quad \dots \end{array} \quad (4.4)$$

Proof. By definition, other than wires A contains only \blacktriangle , \circ , \blacktriangledown , and \boxed{a} . All \boxed{a} 's can be removed with (Ept). Meanwhile all the spiders copy red circle due to (Bs0, K0, B.1) respectively. \square

Just as it is typical to represent a polynomial in normal form as a sum of products, it is possible to rewrite every arithmetic diagram into a normal form as a single \blacktriangle , followed by a layer of \circ , followed by a layer of \boxed{a} , \blacktriangledown .

Definition 4.2. An n -output arithmetic diagram is said to be written in **polynormal form (PNF)** if it looks like:



The i th coefficient a_i is connected to the k th \blacktriangledown iff the k th bit in the binary expansion of i is 1.

This normal form is familiar from completeness of all linear maps for qubits [11] and for qudits [16]. The reason we introduce the definition of a PNF is that it is an arithmetic diagram and therefore has a more immediate arithmetic interpretation. The reason for the specific connectivity condition is that it enables a PNF to directly represent its own matrix.

Proposition 5.

$$\begin{array}{c} \text{PNF Diagram} \end{array} = \begin{bmatrix} 1 & a_0 \\ 0 & a_1 \\ \dots & \dots \\ 0 & a_{2^n-1} \end{bmatrix} \quad (4.5)$$

Proof. See Appendix C. \square

Thus, every controlled state can be represented as at least one arithmetic diagram (namely, its PNF). Moreover, we now show that any other arithmetic diagram can always be rewritten to its PNF.

Theorem 4.1. *All arithmetic diagrams can be rewritten into PNF through application of ZXW rules. Therefore, those ZXW calculus rules applied suffice for completeness for the arithmetic fragment of the ZXW calculus.*

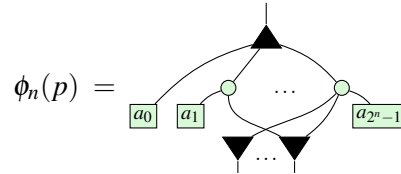
Proof. An algorithm to rewrite any arithmetic diagram to PNF is given in Appendix C. \square

4.3 Isomorphism

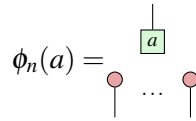
At last we can prove the isomorphism. Throughout we shall let \mathcal{P}_n denote the ring $\mathbb{C}[x_1, \dots, x_n]/(x_1^2, \dots, x_n^2)$.

Theorem 4.2. *There is an isomorphism $\mathcal{P}_n \simeq \tilde{S}_n$*

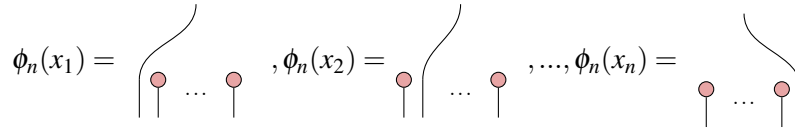
First, we shall define the map $\phi_n : \mathcal{P}_n \rightarrow \tilde{S}_n$ before proving it induces an isomorphism. ϕ_n is defined to map an arbitrary polynomial $p(x_1, \dots, x_n) = a_0 + a_1 x_1 + \dots + a_{2^n-1} x_1 x_2 \dots x_n$ to the following PNF:



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



And mapping indeterminates x_i :



The full proof is found in Appendix C.

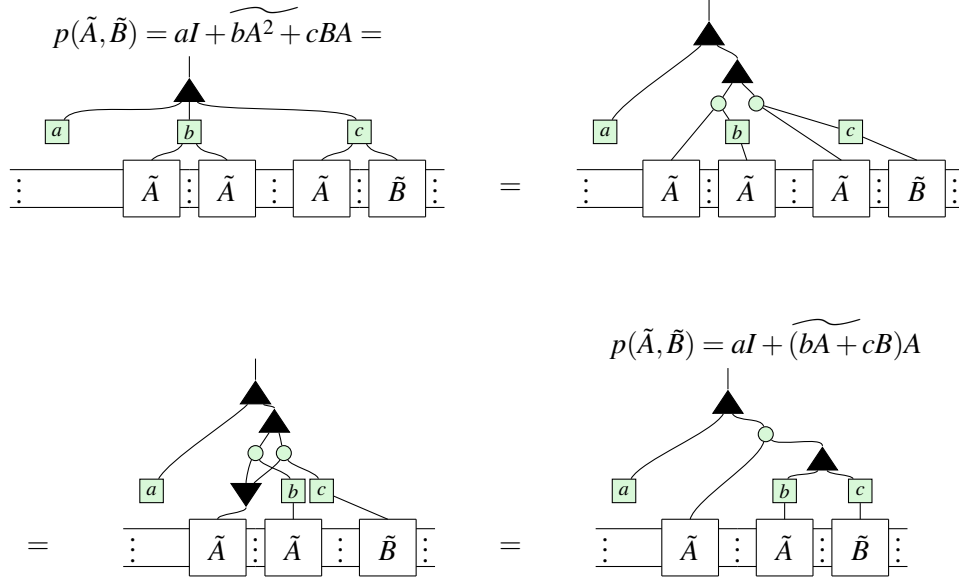
4.4 Factorisation

Instead of the indeterminates being complex numbers represented by \boxed{a} 's, we can let them be same-size matrices represented by controlled square matrix diagrams. We then have that:

Theorem 4.3. *ZXW diagrams where the outputs of an arithmetic ZXW diagram are each plugged into controls of same-size controlled matrices, are isomorphic to multivariate polynomials over same-size square matrices with complex number coefficients. The rules for their completeness are the same subset of ZXW rules used for completeness for arithmetic diagrams in the ZXW calculus in Theorem 4.1, plus the controlled square matrix as a generator along with the four rewrite rules for it in Definition 3.1, Lemma 4.1, and Lemma 4.2.*

Proof. The proof is by the same algorithm for rewriting to PNF as Theorem 4.1, modifying step (6) to copy controlled square matrices using Lemma 4.1, using Lemma 4.2 to commute controlled square matrices whose controls act on mutually exclusive sectors. \square

As an application, we leverage both our rewrite rules for arithmetic ZXW diagrams, and for controlled diagrams, to *factor* them. For example, for same size square matrices I, A, B and $a, b, c \in \mathbb{C}$:



Factoring Hamiltonians is important to optimise quantum algorithms for chemistry and physics simulations. However, previous graphical rewrites for factoring Hamiltonians had only been doable for Hamiltonians with concretely-specified matrix terms [21]. This completeness result guarantees that for any Hamiltonian, even if its matrix terms are black-box, these graphical rewrite rules are capable of deriving any of its possible factorisations.

5 Complexity

This section examines the significance of translating quantum circuits into polynomials from the perspective of computational complexity. In particular, since there is an efficient (randomised) algorithm for polynomial identity testing, we show it is unlikely there is an efficient algorithm for rewriting an arbitrary ZXW diagram into an arithmetic ZXW diagram.

5.1 Algebraic Circuits

While boolean complexity theory studies the number of AND/OR gates required to compute some specified function, algebraic circuit complexity studies the number of addition/multiplication operations required to compute some specified polynomial. More formally, we have the following definition from [?].

Definition 5.1. An algebraic 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 .

Definition 5.2. Given an arithmetic circuit C that describes a polynomial $p(x_1, \dots, x_n) \in \mathbb{F}[x_1, \dots, x_n]$, the polynomial identity testing problem (*PIT*) is to decide whether $p = 0$.

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 [?]. Thanks to the Schwartz-Zippel Lemma, there is an efficient randomised algorithm for *PIT* which simply evaluates the polynomial on random inputs and checks whether all of them are zero. More precisely, $PIT \in \text{coRP}$, a complexity class conjectured to equal P .

By theorem 4.2, we can interpret any quantum state as an algebraic circuit (formal proposition?). Therefore, *PIT* can be used to compare quantum states. The question is how easily this can be done.

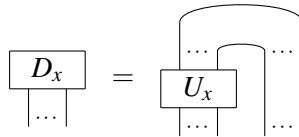
5.2 Proof Complexity

In the original ZXW completeness paper [16], the proof of completeness centres around a normal form equivalent to the one used in section ?? . The goal of the proof is to show that any state can be rewritten into this normal form using the ZXW rules. This implies completeness because one can simply reverse the rules to show equality between two different diagrams. Since the normal form is a direct representation of the diagram's state-vector, it may be exponentially larger than the starting diagram and so the proof of completeness gives an exponential upper bound for the length of proofs of equality in the ZXW calculus. A possible strategy for speeding this process up would be to rewrite the starting diagrams into compact arithmetic diagrams, and then use *PIT* to compare them. We now prove this impossible, under standard complexity assumptions.

Proposition 6. If all quantum states can be rewritten to arithmetic diagrams in polynomial time, then $\text{RP} = \text{NP}$.

Proof. Suppose that we can efficiently rewrite quantum states to arithmetic diagrams. Then we shall reduce an NQP-complete problem to $\overline{PIT} \in \text{RP}$. Since $\text{RP} \subseteq \text{NP} \subseteq \text{NQP}$, this implies $\text{RP} = \text{NP}$.

The NQP-complete problem we are considering is the exact non-identity problem (*ENI*) which decides whether some unitary does not compute the identity matrix [?]. Let x be a description of quantum circuit U_x over $n = \text{poly}(|x|)$ qubits. Then by translating each gate of U_x into a constant ZXW diagram, we can rewrite U_x into a ZXW diagram in polynomial time. Now bend the ZXW diagram into a state D_x , as below.



Now apply the assumption to rewrite D_x into an arithmetic diagram A_x . Interpret A_x as a polynomial p_x over variables $x_1, \dots, x_n, y_1, \dots, y_n$. 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 $\overline{PIT}(p_x - p_{\cap}^n) = 1 \iff \text{ENI}(U_x) = 1$. Thus we have reduced *ENI* to \overline{PIT} . □

Note that under standard derandomisation assumptions, $P = \text{RP} = \text{coRP}$. Therefore, the result above effectively shows that it is NP-hard to rewrite a ZXW diagram to an arithmetic diagram. In fact, NQP is believed to be strictly larger than NP so it even harder.

What remains open is whether it is at all possible to rewrite every quantum size into a small arithmetic diagram. More conclusion

6 Conclusion

To conclude, we first introduced the higher-order map `Ctrl` and showed it is a lax monoidal functor on all same-size square matrices. This enabled us to add functorial boxes to such ZXW diagrams. Moreover, we gave rewrite rules for interactions between controlled diagrams and all generators Z , X , W , and H .

We further proved completeness for all controlled n -partite states, which we showed form a commutative ring isomorphic to multilinear polynomials. Also, we showed that all controlled n -qubit square matrices form a non-commutative ring. Furthermore, we have completeness for plugging controlled states into the control wires of controlled diagrams, isomorphic to all multivariate polynomials over same-size square matrices, with application to factoring Hamiltonians. When the controls target mutually exclusive sectors, a rewrite rule can be applied to copy any controlled diagram, and thereby factor any Hamiltonian.

This work opens up connections between quantum circuit complexity and the far better understood algebraic complexity. We have shown that every (controlled) state computes a polynomial; hence, we can interpret a universal fragment of the ZXW calculus as corresponding to arithmetic circuits. This generalises [?], which found an algebraic interpretation of a certain fragment of ZW calculus. Reinterpreting quantum circuits as computing polynomials in relation to algebraic complexity was explored in the Master’s thesis associated with this work [?].

In another direction, we can apply completeness for polynomials isomorphic to controlled states to study entanglement. It can be easily shown diagrammatically that the polynomials corresponding to entangled (non-separable) states are exactly those that cannot be factored into irreducibles containing only variables corresponding to Alice’s subsystem or only corresponding to Bob’s subsystem. Since there are efficient algorithms for polynomial factorisation, this gives rise to a novel entanglement classification algorithm for pure states. Further developing this into a more refined algebraic theory of entanglement, building on the work in Ref. [?], could offer further insights.

The natural next step is to derive extensions of our results for controlled qubit diagrams to qudits. While the diagrams being controlled are over qudits, we can consider control in the qubit subspace, as done in the ZXW calculus completeness proof for any qudit dimension [16]. A starting guess would be that qudit controlled states are isomorphic to polynomials $\mathbb{C}^{d-1}[x_1, \dots, x_n]/(x_1^d, \dots, x_n^d)$ due to the Hopf law between Z and W . Qudit multiple-control would likely have more complex structure than the qubit case here, considering the constructions for all prime-dimensional d -ary classical reversible gates built in Ref. [19].

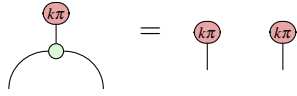
We would like to try sector-preserving channels [24] and scoped effects [14] as approaches to better formulate the monadic nature of multiple-control. We are also curious about reconciling the interpretation of diagrammatic differentiation of our arithmetic polynomial circuits by the approach in Ref. [28], with that of quantum circuits and ZX diagrams in Refs. [23, 26, 12]. Last but not least, these new semantics for quantum controlled states and matrices could be embedded categorically into a host functional programming language like in Ref. [18], or translated to an equational theory for a quantum programming language like in Ref. [22].

7 Acknowledgements

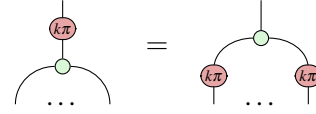
We are grateful to all the collaboration with Matt Wilson in finding that control is a lax monoidal functor. We thank Razin Shaikh, Itai Leigh, Tim Forrer, and Aleks Kissinger for insightful discussions. LY is funded by a Google PhD Fellowship.

References

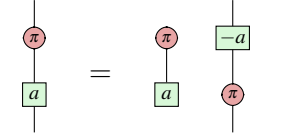
- [1] BACKENS, M., AND KISSINGER, A. ZH: A Complete Graphical Calculus for Quantum Computations Involving Classical Non-linearity. In *Proceedings of the 15th International Conference on Quantum Physics and Logic, Halifax, Canada, 3-7th June 2018* (2019), P. Selinger and G. Chiribella, Eds., vol. 287 of *Electronic Proceedings in Theoretical Computer Science*, Open Publishing Association, pp. 23–42.
- [2] BAEZ, J. C., AND ERBELE, J. Categories in control, 2015.
- [3] CHARDONNET, K., DE VISME, M., VALIRON, B., AND VILMART, R. The many-worlds calculus, 2023.
- [4] CHILDS, A. M., AND WIEBE, N. Hamiltonian simulation using linear combinations of unitary operations. *Quantum Info. Comput.* 12, 11–12 (nov 2012), 901–924.
- [5] COECKE, B., AND DUNCAN, R. Interacting quantum observables: categorical algebra and diagrammatics. *New Journal of Physics* 13 (2011), 043016.
- [6] COECKE, B., AND KISSINGER, A. The compositional structure of multipartite quantum entanglement. In *International Colloquium on Automata, Languages, and Programming* (2010), Springer, pp. 297–308.
- [7] COECKE, B., KISSINGER, A., MERRY, A., AND ROY, S. The ghz/w-calculus contains rational arithmetic. *arXiv preprint arXiv:1103.2812* (2011).
- [8] DE FELICE, G., SHAIKH, R. A., POÓR, B., YEH, L., WANG, Q., AND COECKE, B. Light-matter interaction in the zxw calculus. *arXiv preprint arXiv:2306.02114* (2023).
- [9] GILYÉN, A., SU, Y., LOW, G. H., AND WIEBE, N. Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing* (New York, NY, USA, 2019), STOC 2019, Association for Computing Machinery, p. 193–204.
- [10] HADZIHASANOVIC, A. The algebra of entanglement and the geometry of composition, 2017.
- [11] HADZIHASANOVIC, A., NG, K. F., AND WANG, Q. Two complete axiomatisations of pure-state qubit quantum computing. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science* (New York, NY, USA, 2018), LICS ’18, Association for Computing Machinery, p. 502–511.
- [12] JEANDEL, E., PERDRIX, S., AND VESHCHEREROVA, M. Addition and differentiation of zx-diagrams, 2024.
- [13] JEANDEL, E., PERDRIX, S., AND VILMART, R. A generic normal form for zx-diagrams and application to the rational angle completeness, 2018.
- [14] LINDLEY, S., MATACHE, C., MOSS, S., STATON, S., WU, N., AND YANG, Z. Scoped effects as parameterized algebraic theories, 2024.
- [15] MELLIÈS, P.-A. Functorial boxes in string diagrams. In *International Workshop on Computer Science Logic* (2006), Springer, pp. 1–30.
- [16] POÓR, B., WANG, Q., SHAIKH, R. A., YEH, L., YEUNG, R., AND COECKE, B. Completeness for arbitrary finite dimensions of zxw-calculus, a unifying calculus. *arXiv preprint arXiv:2302.12135* (2023).



$$(K0)$$



$$(K1)$$



$$(K2)$$



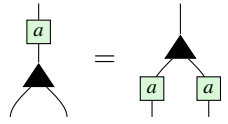
$$(Zer)$$



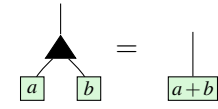
$$(H)$$

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

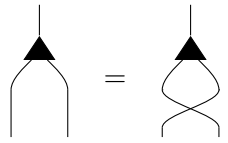
ZW Rules:



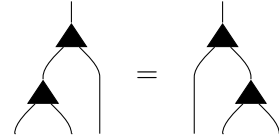
$$(Pcy)$$



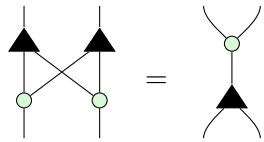
$$(ADD)$$



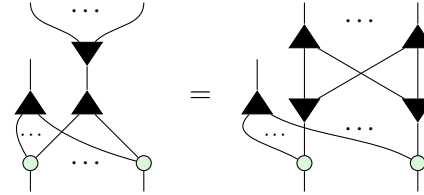
$$(Sym)$$



$$(Aso)$$

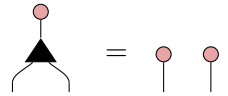


$$(BZW)$$

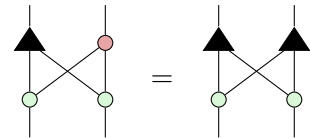


$$(WW)$$


ZXW Rules:



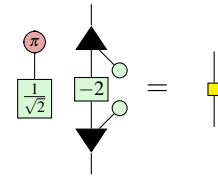
$$(Bs0)$$



$$(TA)$$



$$(Bsj)$$



$$(HD)$$

Appendix B Basic Lemmas

The following two lemmas follow immediately from the bra-ket definition of \blacktriangle :

Lemma B.1.

$$\begin{array}{c} \text{---} \\ \diagup \quad \diagdown \\ \bullet \end{array} = \text{---} = \begin{array}{c} \text{---} \\ \diagdown \quad \diagup \\ \bullet \end{array} \quad (\text{B.1})$$

Lemma B.2.

$$\begin{array}{c} \text{---} \\ \diagup \quad \diagdown \\ \bullet \end{array} = \begin{array}{c} \pi \\ \bullet \\ \text{---} \end{array} \quad (\text{B.2})$$

Lemma B.3.

$$\begin{array}{c} k\pi \\ \bullet \\ \text{---} \end{array} = \begin{array}{c} k\pi \\ \text{---} \end{array} \quad (\text{B.3})$$

Proof.

$$\begin{array}{c} k\pi \\ \bullet \\ \text{---} \end{array} = \begin{array}{c} k\pi \\ \text{---} \end{array} \stackrel{(H)}{=} \begin{array}{c} k\pi \\ \text{---} \end{array} \stackrel{(K0)}{=} \begin{array}{c} k\pi \\ \text{---} \end{array} = \begin{array}{c} k\pi \\ \text{---} \end{array}$$

□

Lemma B.4.

$$\begin{array}{c} \text{---} \\ \diagup \quad \diagdown \\ \text{---} \end{array} \stackrel{(0)}{=} \begin{array}{c} \text{---} \\ \text{---} \end{array} \quad (\text{B.4})$$

Proof.

$$\begin{array}{c} \text{---} \\ \diagup \quad \diagdown \\ \text{---} \end{array} \stackrel{(0)}{=} \begin{array}{c} \text{---} \\ \text{---} \end{array} \stackrel{(Zer)}{=} \begin{array}{c} \text{---} \\ \text{---} \end{array} \stackrel{(K0)}{=} \begin{array}{c} \text{---} \\ \text{---} \end{array}$$

□

Lemma B.5.

$$\begin{array}{c} \blacktriangle \\ \diagup \quad \diagdown \\ a \quad b \\ \diagdown \quad \diagup \\ \blacktriangle \end{array} = \begin{array}{c} \text{---} \\ \text{---} \end{array} \quad (\text{B.5})$$

Proof.

$$\begin{array}{c} \blacktriangle \\ \diagup \quad \diagdown \\ a \quad b \\ \diagdown \quad \diagup \\ \blacktriangle \end{array} = \begin{array}{c} \text{---} \\ \text{---} \end{array} \stackrel{(BZW)}{=} \begin{array}{c} \text{---} \\ \text{---} \end{array} \stackrel{(ADD)}{=} \begin{array}{c} \text{---} \\ \text{---} \end{array}$$

□

Lemma B.6.

$$\begin{array}{c} \blacktriangle \\ \diagup \quad \diagdown \\ \pi \\ \diagdown \quad \diagup \\ \blacktriangle \end{array} = \begin{array}{c} \pi \\ \text{---} \end{array} \quad (\text{B.6})$$

Proof.

$$\text{Loop with } \pi = \boxed{1} \boxed{-1} \stackrel{(B.5)}{=} \boxed{0} \stackrel{(B.4)}{=} \text{Two red circles on a line}$$

□

Lemma B.7.

$$\text{Diamond with } a_1, a_2, \dots, a_n = \boxed{\sum_{i=1}^n a_i} \text{ (vertical line with green circle)}$$

(B.7)

Proof.

$$\text{Diamond } a_1, a_2, \dots, a_n \stackrel{(BZW)}{=} \dots \stackrel{(Aso)}{=} \dots \stackrel{(B.5)}{=} \boxed{\sum_{i=1}^n a_i} \text{ (vertical line with green circle)}$$

□

Lemma B.8.

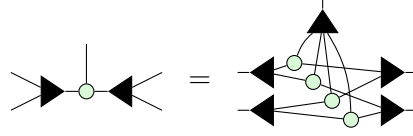
$$\text{Green circle over black triangle} = \text{Two red circles on a line}$$

(B.8)

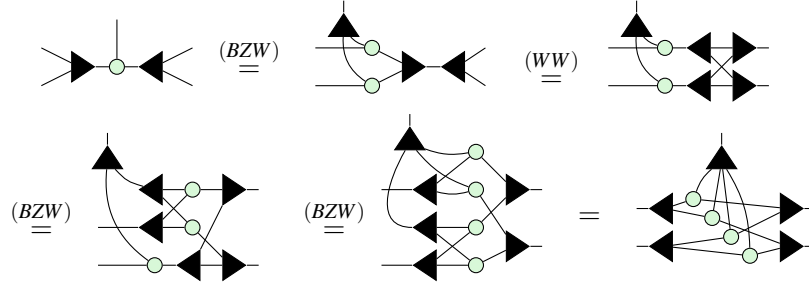
Proof.

$$\text{Green circle over black triangle} \stackrel{(TA)}{=} \dots \stackrel{(BZW)}{=} \dots \stackrel{(K0)}{=} \dots \stackrel{(Bs0)}{=} \dots \stackrel{(Ept)}{=} \text{Two red circles on a line}$$

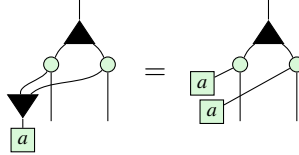
□

Lemma B.9.

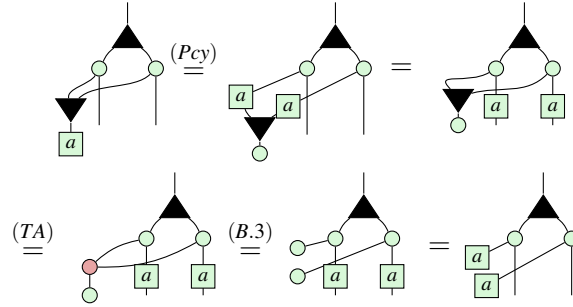
(B.9)

Proof.

□

Lemma B.10.

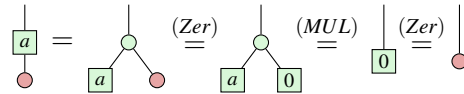
(B.10)

Proof.

□

Lemma B.11.

(B.11)

Proof.

□

Appendix C Main Proofs

Proof of Lemma ??

Proof. We can verify this computes the AND gate by computing on basis states.

$$\begin{array}{c}
 \begin{array}{c} \text{Diagram 1: } \pi \text{ and } \pi \text{ inputs to a green circle, which then feeds into a } \pi \text{ gate.} \\ \text{Diagram 2: } \pi \text{ and } \pi \text{ inputs to a green circle, which then feeds into a } \pi \text{ gate.} \end{array} \stackrel{(Bs0)}{=} \begin{array}{c} \text{Diagram 3: } \pi \text{ and } \pi \text{ inputs to a green circle, which then feeds into a } \pi \text{ gate.} \\ \text{Diagram 4: } \pi \text{ and } \pi \text{ inputs to a green circle, which then feeds into a } \pi \text{ gate.} \end{array} \stackrel{(K0)}{=} \begin{array}{c} \text{Diagram 5: } \pi \text{ and } \pi \text{ inputs to a green circle, which then feeds into a } \pi \text{ gate.} \\ \text{Diagram 6: } \pi \text{ and } \pi \text{ inputs to a green circle, which then feeds into a } \pi \text{ gate.} \end{array} \stackrel{(B.1)}{=} \text{Diagram 7: } \pi \text{ and } \pi \text{ inputs to a green circle, which then feeds into a } \pi \text{ gate.}
 \end{array} \tag{C.1}$$

Thus $AND(1, x) = x$. Since the diagram is clearly commutative, it remains to check $AND(0, x) = 0$.

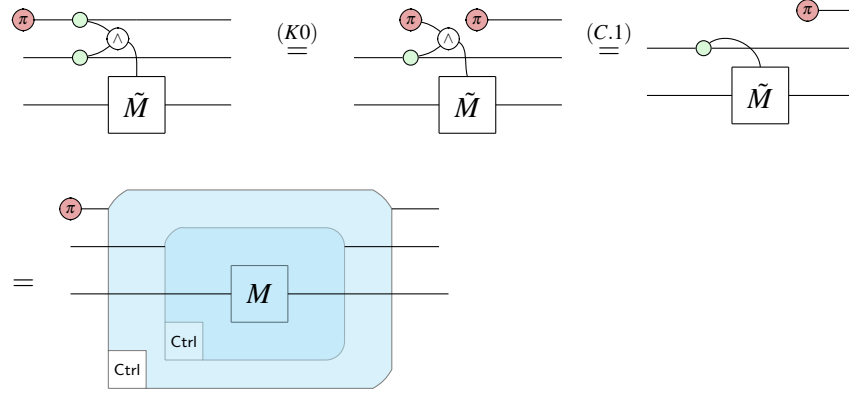
$$\begin{array}{c}
 \begin{array}{c} \text{Diagram 1: } \pi \text{ and } \pi \text{ inputs to a green circle, which then feeds into a } \pi \text{ gate.} \\ \text{Diagram 2: } \pi \text{ and } \pi \text{ inputs to a green circle, which then feeds into a } \pi \text{ gate.} \end{array} = \begin{array}{c} \text{Diagram 3: } \pi \text{ and } \pi \text{ inputs to a green circle, which then feeds into a } \pi \text{ gate.} \\ \text{Diagram 4: } \pi \text{ and } \pi \text{ inputs to a green circle, which then feeds into a } \pi \text{ gate.} \end{array} \stackrel{(BZW)}{=} \begin{array}{c} \text{Diagram 5: } \pi \text{ and } \pi \text{ inputs to a green circle, which then feeds into a } \pi \text{ gate.} \\ \text{Diagram 6: } \pi \text{ and } \pi \text{ inputs to a green circle, which then feeds into a } \pi \text{ gate.} \end{array} \stackrel{(K0)}{=} \begin{array}{c} \text{Diagram 7: } \pi \text{ and } \pi \text{ inputs to a green circle, which then feeds into a } \pi \text{ gate.} \\ \text{Diagram 8: } \pi \text{ and } \pi \text{ inputs to a green circle, which then feeds into a } \pi \text{ gate.} \end{array} \stackrel{(B.2)}{=} \begin{array}{c} \text{Diagram 9: } \pi \text{ and } \pi \text{ inputs to a green circle, which then feeds into a } \pi \text{ gate.} \\ \text{Diagram 10: } \pi \text{ and } \pi \text{ inputs to a green circle, which then feeds into a } \pi \text{ gate.} \end{array} \stackrel{(B.1)}{=} \begin{array}{c} \text{Diagram 11: } \pi \text{ and } \pi \text{ inputs to a green circle, which then feeds into a } \pi \text{ gate.} \\ \text{Diagram 12: } \pi \text{ and } \pi \text{ inputs to a green circle, which then feeds into a } \pi \text{ gate.} \end{array} = \begin{array}{c} \text{Diagram 13: } \pi \text{ and } \pi \text{ inputs to a green circle, which then feeds into a } \pi \text{ gate.} \\ \text{Diagram 14: } \pi \text{ and } \pi \text{ inputs to a green circle, which then feeds into a } \pi \text{ gate.} \end{array}
 \end{array} \tag{C.2}$$

□

Proof of Proposition 3

Proof. Plugging basis states:

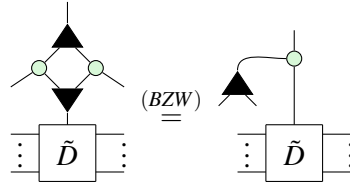
$$\begin{array}{c}
 \begin{array}{c} \text{Diagram 1: } \text{Control line with a green circle and } \pi \text{ gate, feeding into } \tilde{M}. \\ \text{Diagram 2: } \text{Control line with a green circle and } \pi \text{ gate, feeding into } \tilde{M}. \end{array} \stackrel{(K0)}{=} \begin{array}{c} \text{Diagram 3: } \text{Control line with a green circle and } \pi \text{ gate, feeding into } \tilde{M}. \\ \text{Diagram 4: } \text{Control line with a green circle and } \pi \text{ gate, feeding into } \tilde{M}. \end{array} \stackrel{(C.2)}{=} \begin{array}{c} \text{Diagram 5: } \text{Control line with a green circle and } \pi \text{ gate, feeding into } \tilde{M}. \\ \text{Diagram 6: } \text{Control line with a green circle and } \pi \text{ gate, feeding into } \tilde{M}. \end{array} \\
 = \begin{array}{c} \text{Diagram 7: } \text{Control line with a green circle and } \pi \text{ gate, feeding into } \tilde{M}. \\ \text{Diagram 8: } \text{Control line with a green circle and } \pi \text{ gate, feeding into } \tilde{M}. \end{array} = \begin{array}{c} \text{Diagram 9: } \text{Control line with a green circle and } \pi \text{ gate, feeding into } \tilde{M}. \\ \text{Diagram 10: } \text{Control line with a green circle and } \pi \text{ gate, feeding into } \tilde{M}. \end{array}
 \end{array}$$



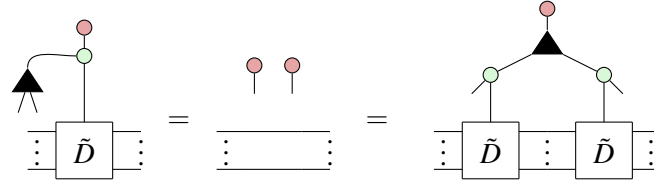
□

Proof of Lemma 4.1

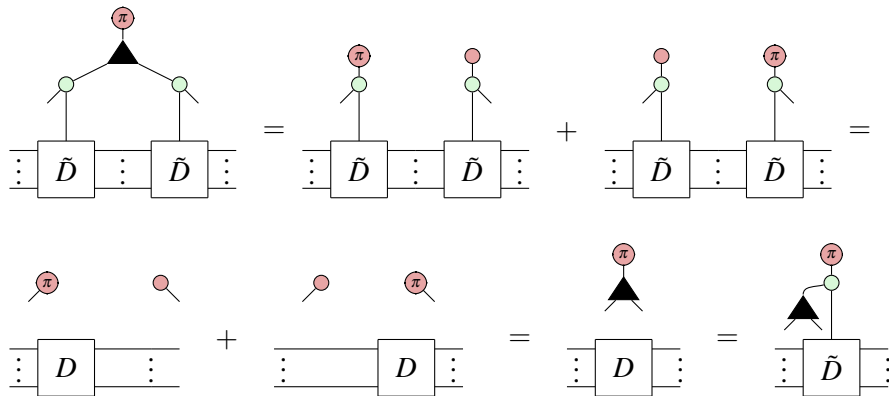
Proof. First of all, using (BZW) we can rewrite the LHS to



Then clearly



Meanwhile,



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

□

Proof of Lemma 4.2

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

$$\begin{aligned}
 & \text{Diagram with } \tilde{M}_1, \tilde{M}_2 \text{ and a red circle } \pi \text{ on top} = \text{Diagram with } M_1 + \text{Diagram with } M_2 \\
 & = \text{Diagram with } M_2 + \text{Diagram with } M_1 = \text{Diagram with } \tilde{M}_2, \tilde{M}_1 \text{ and a red circle } \pi \text{ on top}
 \end{aligned}$$

□

Proof of Lemma 4.4

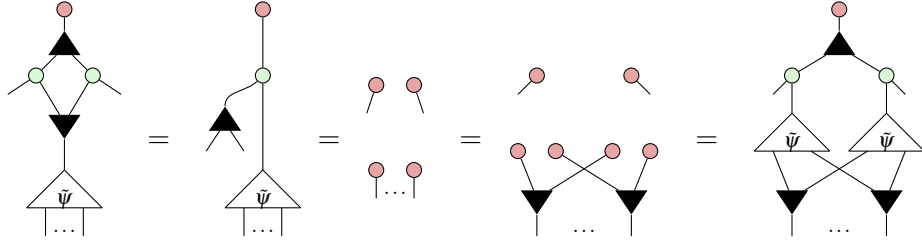
Proof.

$$\begin{aligned}
 & \text{Diagram 1} \stackrel{(BZW)}{=} \text{Diagram 2} \\
 & \stackrel{(4.1)}{=} \text{Diagram 3} \\
 & \stackrel{(4.2)}{=} \text{Diagram 4}
 \end{aligned}$$

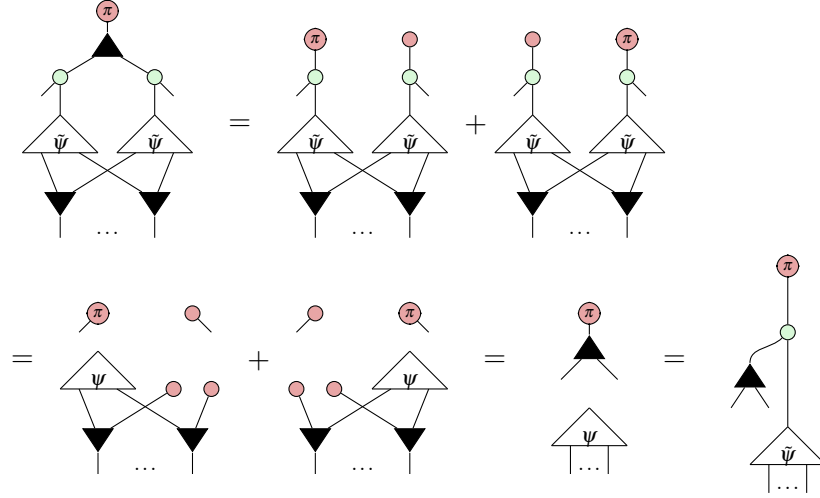
□

Proof of Lemma 4.5

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

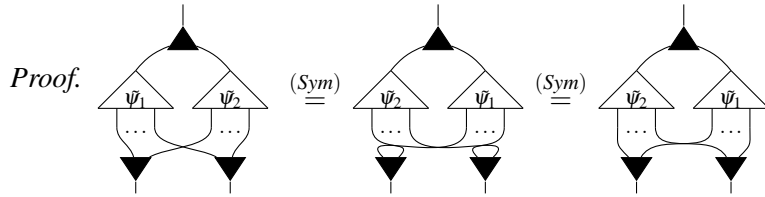


Meanwhile, plugging $|1\rangle$ gives



Completing the proof □

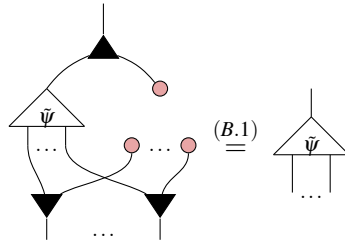
Proof of Lemma 4.6



Proof of Lemma 4.7

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

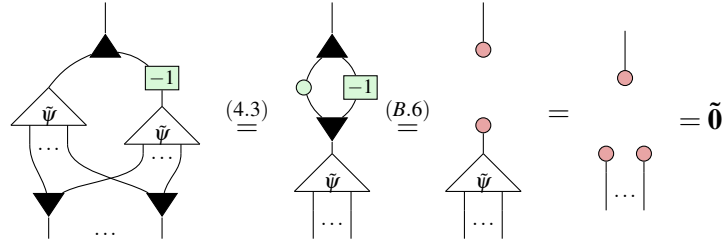
Then we have:



□

Proof of Lemma 4.8

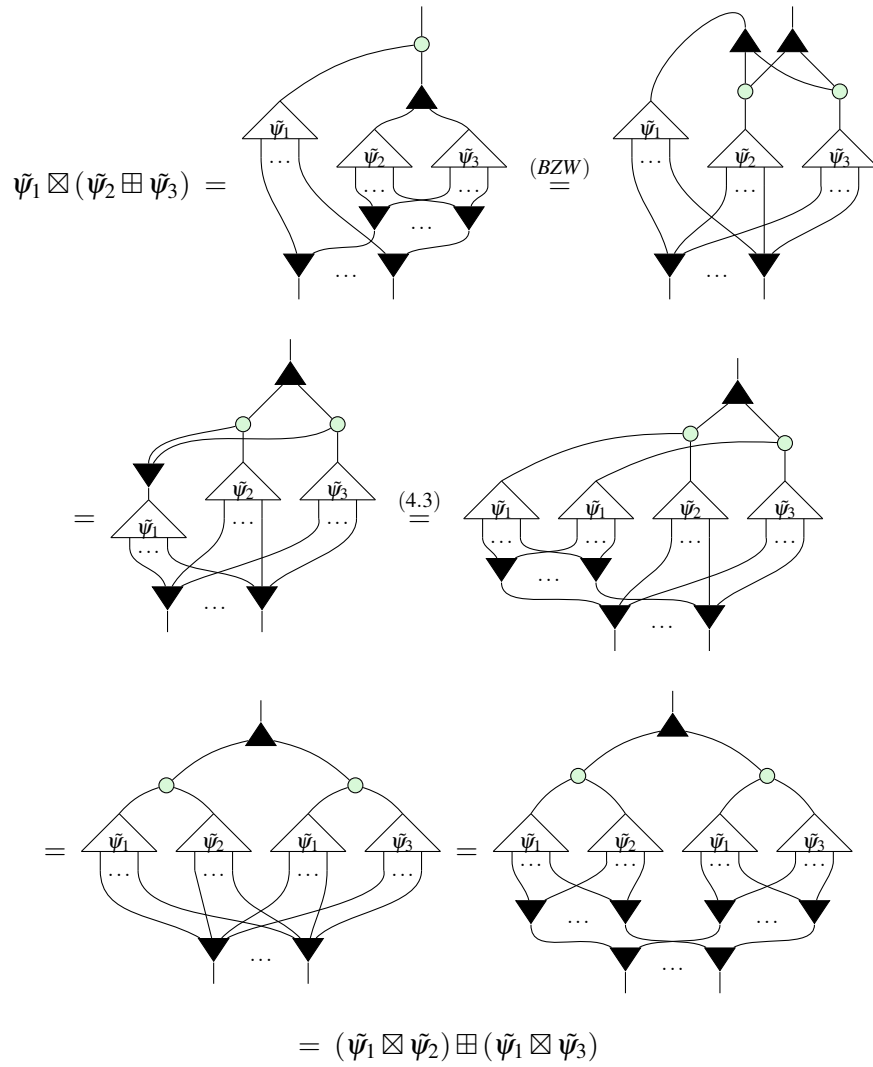
Proof. $\tilde{\psi} \circ \boxed{-1}$ is still a controlled state since $\boxed{-1}$ does nothing to \bullet . Then $\tilde{\psi} \circ \boxed{-1}$ inverts $\tilde{\psi}$ since:



□

Proof of Lemma 4.8

Proof.



□

Proof of Proposition 5

Proof. We prove by induction on n .

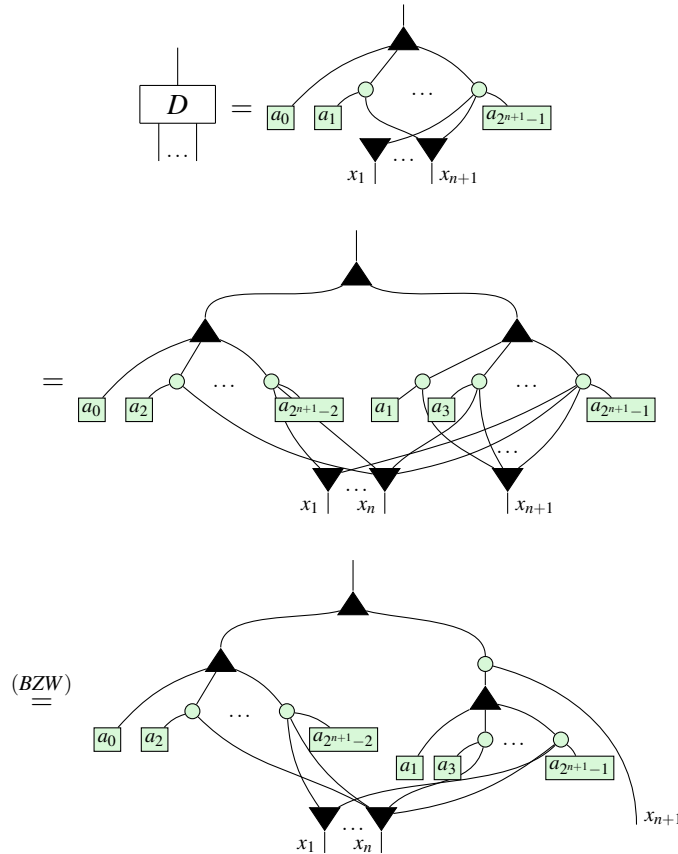
For the base case, $n = 0$. The only PNF with no outputs is a number so we have:

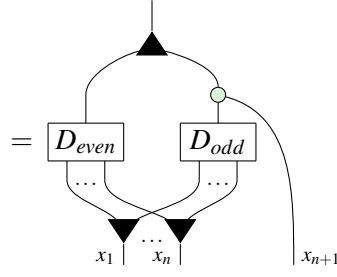
$$\boxed{a_0} = \begin{bmatrix} 1 & a_0 \end{bmatrix}$$

as desired.

For inductive hypothesis, we assume that (4.5) 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:

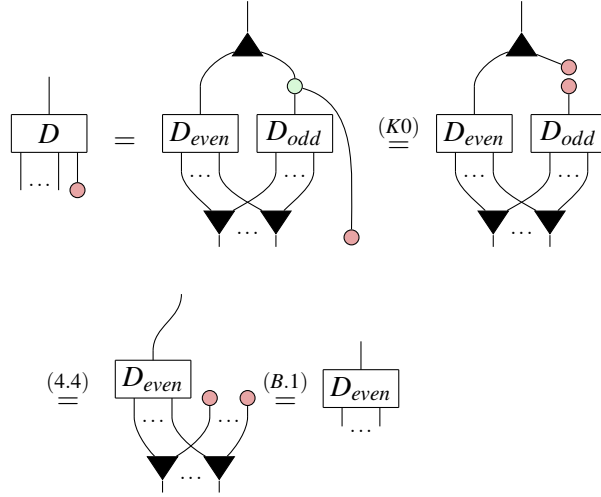




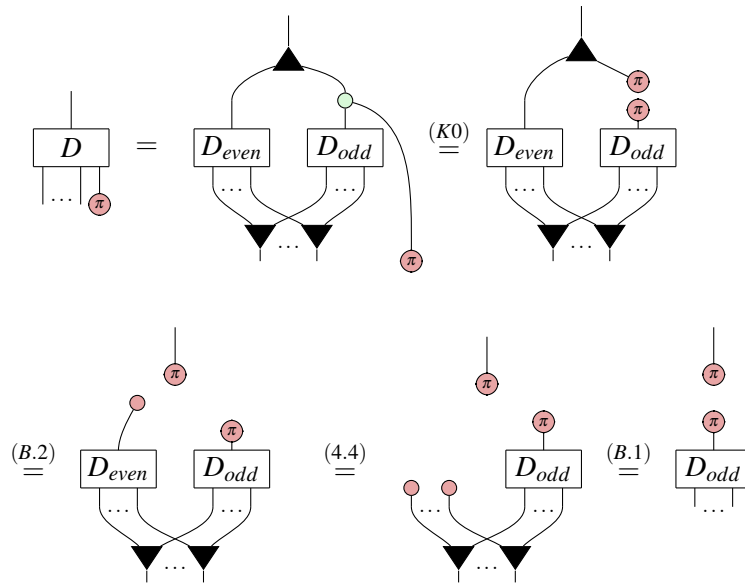
Where D_{even}, D_{odd} are PNF diagrams. Since they are over n variables, we can apply the inductive hypothesis and obtain:

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

Next, plugging red we observe:



Meanwhile,



Summing these together,

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

Completing the inductive step. □

Proof of Theorem 4.1

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

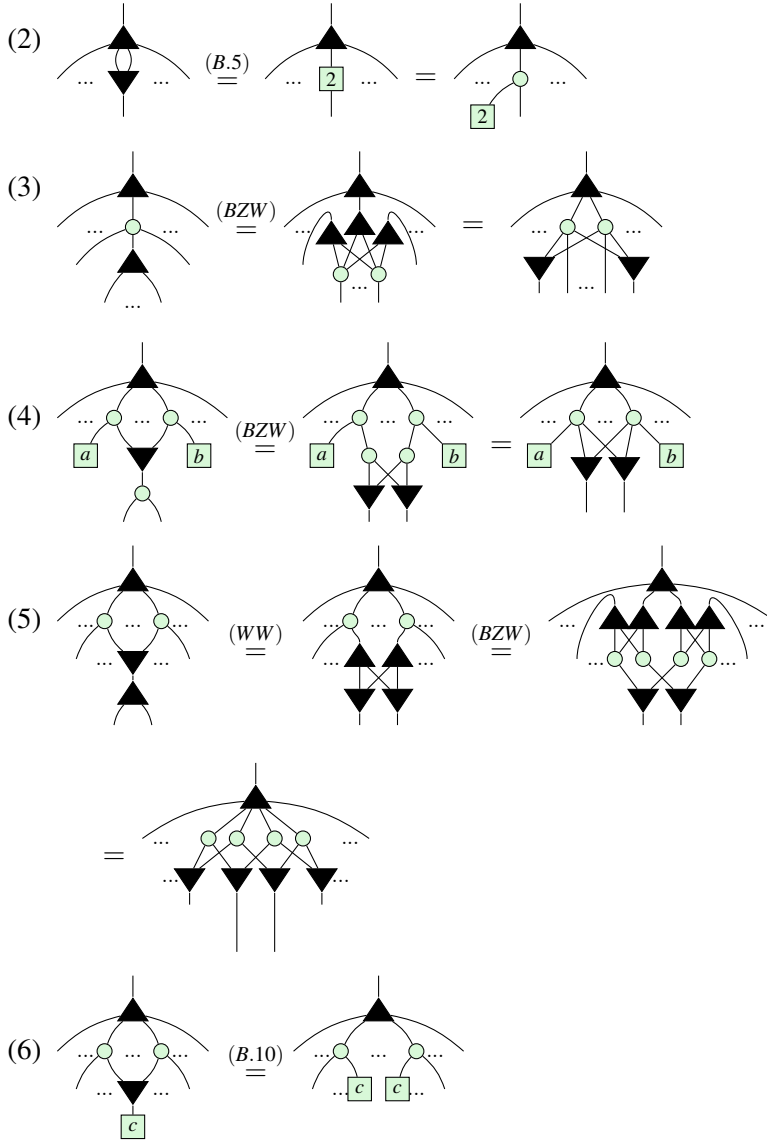
Otherwise, A has at least one output. First, we shall rewrite A into three layers, consisting of: (1) a single W at the top, (2) a layer of $\begin{array}{c} \circ \\ \diagup \quad \diagdown \\ \dots \end{array}$ and (3) a layer of \boxed{a} 's and $\begin{array}{c} \diagup \quad \diagdown \\ \dots \end{array}$'s.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

If the top of A is not already $\begin{array}{c} \blacktriangle \\ \diagup \quad \diagdown \\ \dots \end{array}$, it must be $\begin{array}{c} \circ \\ \diagup \quad \diagdown \\ \dots \end{array}$. It cannot be \boxed{a} since the remaining arithmetic diagram would then have no inputs which is impossible. It cannot be $\begin{array}{c} \blacktriangledown \\ \diagup \quad \diagdown \\ \dots \end{array}$ since there is only one input and arithmetic diagrams cannot contain \cap . Thus we can rewrite:

$$(1) \quad \begin{array}{c} \circ \\ \diagup \quad \diagdown \\ \dots \end{array} \stackrel{(B.1)}{=} \begin{array}{c} \blacktriangle \\ \diagup \quad \diagdown \\ \dots \end{array} \begin{array}{c} \circ \\ \diagup \quad \diagdown \\ \dots \end{array}$$


(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

(B.8) is applied and simplified with (K0) to remove $\begin{array}{c} \circ \\ \diagup \quad \diagdown \\ \dots \end{array}$ 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 ∇ '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 ∇ 's, use (B.7) to fuse them together.
- (8) Pad: use (B.4) 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  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. □

Proof of Theorem 4.2

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

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

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 $p+q = a_0+b_0 + (a_1+b_1)x_1$,

$$\begin{aligned} \phi_1(p) \boxplus \phi_1(q) &= \begin{array}{c} \text{Diagram 1: A green circle with two inputs and two outputs. The inputs are labeled } a_0 \text{ and } a_1 \text{ in green boxes. The outputs are labeled } b_0 \text{ and } b_1 \text{ in green boxes.} \end{array} = \begin{array}{c} \text{Diagram 2: A green circle with two inputs and two outputs. The inputs are labeled } a_0 \text{ and } a_1 \text{ in green boxes. The outputs are labeled } b_0 \text{ and } b_1 \text{ in green boxes.} \end{array} = \begin{array}{c} \text{Diagram 3: A green circle with two inputs and two outputs. The inputs are labeled } a_0+b_0 \text{ and } a_1 \text{ in green boxes. The outputs are labeled } a_1 \text{ and } b_1 \text{ in green boxes.} \end{array} \\ &\stackrel{(B.5)}{=} \begin{array}{c} \text{Diagram 4: A green circle with two inputs and two outputs. The inputs are labeled } a_0+b_0 \text{ and } a_1+b_1 \text{ in green boxes.} \end{array} = \phi_1(p+q) \end{aligned}$$

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

$$\begin{aligned} \phi_1(p) \boxtimes \phi_1(q) &= \begin{array}{c} \text{Diagram 1: A green circle with two inputs and two outputs. The inputs are labeled } a_0 \text{ and } a_1 \text{ in green boxes. The outputs are labeled } b_0 \text{ and } b_1 \text{ in green boxes.} \end{array} \stackrel{(B.9)}{=} \begin{array}{c} \text{Diagram 2: A green circle with two inputs and two outputs. The inputs are labeled } a_0 \text{ and } b_0 \text{ in green boxes. The outputs are labeled } a_1 \text{ and } b_1 \text{ in green boxes.} \end{array} \\ &\stackrel{(B.10)}{=} \begin{array}{c} \text{Diagram 3: A green circle with two inputs and two outputs. The inputs are labeled } a_0b_0 \text{ and } b_0 \text{ in green boxes. The outputs are labeled } a_1 \text{ and } b_1 \text{ in green boxes.} \end{array} \stackrel{(Pcy)}{=} \begin{array}{c} \text{Diagram 4: A green circle with two inputs and two outputs. The inputs are labeled } a_0b_0 \text{ and } a_1b_0 \text{ in green boxes. The outputs are labeled } a_0b_1 \text{ and } a_1b_1 \text{ in green boxes.} \end{array} \end{aligned}$$

$$\begin{aligned}
& \stackrel{(B.8)}{=} \text{Diagram 1} \stackrel{(B.11)}{=} \text{Diagram 2} \stackrel{(B.5)}{=} \text{Diagram 3} \\
& = \phi_1(p \times q)
\end{aligned}$$

Completing the base case.

Inductive step:

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

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:

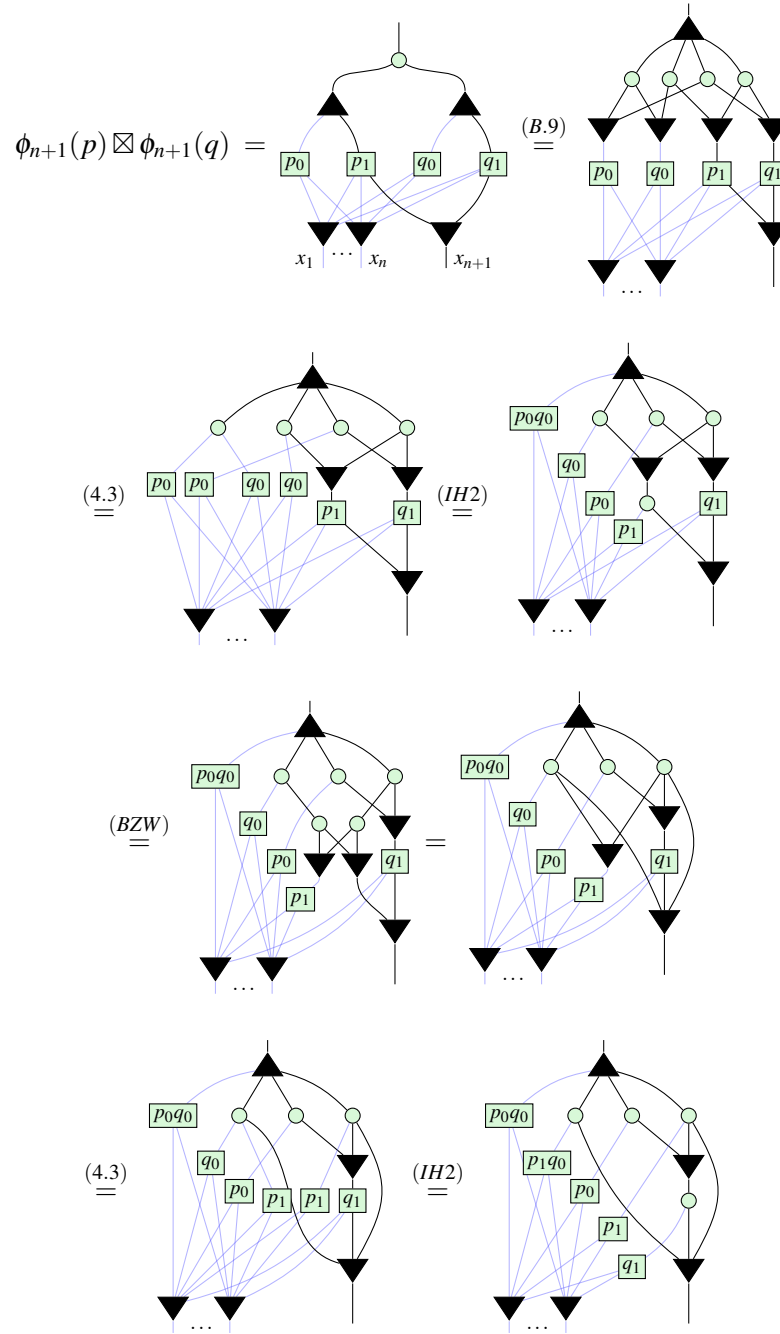
$$\begin{array}{c} \text{Diagram 1} \end{array} \stackrel{(IH1)}{=} \begin{array}{c} \text{Diagram 2} \end{array}$$

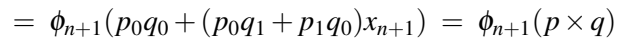
$$\begin{array}{c} \text{Diagram 3} \end{array} \stackrel{(IH2)}{=} \begin{array}{c} \text{Diagram 4} \end{array}$$

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:

$$\begin{aligned}
\phi_{n+1}(p) \boxplus \phi_{n+1}(q) &= \text{Diagram 5} \stackrel{(Aso)}{=} \text{Diagram 6} \\
&\stackrel{(IH1)}{=} \text{Diagram 7} \stackrel{(BZW)}{=} \text{Diagram 8} \stackrel{(IH1)}{=} \text{Diagram 9} \\
&= \phi_{n+1}(p_0 + q_0 + (p_1 + q_1)x_{n+1}) = \phi_{n+1}(p + q)
\end{aligned}$$

Similarly, for multiplication:

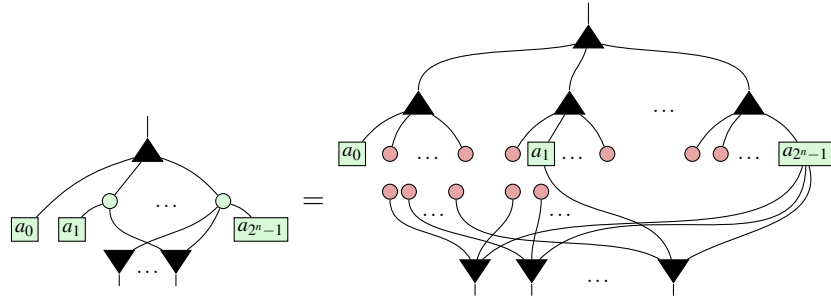




Finally, to see ϕ_n is an isomorphism, we use Theorem 4.1 to write an arbitrary controlled state in PNF:

$$\begin{bmatrix} 1 & a_0 \\ 0 & a_1 \\ \vdots & \ddots \\ 0 & a_{2^n-1} \end{bmatrix} = \text{Diagram of a butterfly network with } 2^n \text{ inputs and } 2^n \text{ outputs.}$$

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



$$= \phi_n(a_0) + \phi_n(a_1 x_n) + \dots + \phi_n(a_{2^n-1} x_1 x_2 \dots x_n)$$

$$= \phi_n(a_0 + a_1 x_n + \dots + a_{2^n-1} x_1 x_2 \dots x_n)$$

□